

Systemdokumentation

PR Software Engineering

Gruppe 3

259035 (2018S)

 Template

2017-05-03

**Über arc42**

arc42, das Template zur Dokumentation von Software- und Systemarchitekturen.

Erstellt von Dr. Gernot Starke, Dr. Peter Hruschka und Mitwirkenden.

Template Revision: 7.0 DE (asciidoc-based), January 2017

© We acknowledge that this document uses material from the arc 42 architecture template, <http://www.arc42.de>. Created by Dr. Peter Hruschka & Dr. Gernot Starke.

Inhaltsverzeichnis

[1. Einleitung 4](#_Toc518406301)

[1.1. Aufgabenstellung 4](#_Toc518406302)

[1.1.1. Inhalt 4](#_Toc518406303)

[1.1.2. Motivation 4](#_Toc518406304)

[1.2. Systemanforderungen 4](#_Toc518406305)

[1.3. Qualitätsziele 5](#_Toc518406306)

[1.4. Stakeholder 5](#_Toc518406307)

[2. Randbedingungen 6](#_Toc518406308)

[3. Kontextabgrenzung 7](#_Toc518406309)

[3.1. Fachlicher Kontext 7](#_Toc518406310)

[3.2. Technischer Kontext 8](#_Toc518406311)

[4. Lösungsstrategie 9](#_Toc518406312)

[5. Bausteinsicht 12](#_Toc518406313)

[5.1. Zerlegung des Gesamtsystems 13](#_Toc518406314)

[5.1.1. Ganzheitlicher Überblick 13](#_Toc518406315)

[5.1.2. Ebene 1 14](#_Toc518406316)

[5.1.3. Ebene 2 15](#_Toc518406317)

[6. Laufzeitsicht 16](#_Toc518406318)

[6.1. Szenario 16](#_Toc518406319)

[7. Verteilungssicht 17](#_Toc518406320)

[7.1. Konkrete Verteilung 17](#_Toc518406321)

[7.2. Begründung der Verteilung 18](#_Toc518406322)

[8. Querschnittliche Konzepte 19](#_Toc518406323)

[8.1. Fachliche Konzepte 19](#_Toc518406324)

[8.1.1. Personen 19](#_Toc518406325)

[8.1.2. Projekte 19](#_Toc518406326)

[8.1.3. Aufgabenbereiche 19](#_Toc518406327)

[8.1.4. Aufgaben 19](#_Toc518406328)

[8.1.5. Activities 19](#_Toc518406329)

[8.2. User Experience 20](#_Toc518406330)

[8.3. Architektur- und Entwurfsmuster 20](#_Toc518406331)

[8.4. Entwicklungskonzepte 20](#_Toc518406332)

[9. Risiken und technische Schulden 21](#_Toc518406333)

[9.1. Verbindungsaufbau zwischen App und Datenbank 21](#_Toc518406334)

[10. Glossar 22](#_Toc518406335)

[11. Quellenverzeichnis 22](#_Toc518406336)

[12. Abbildungsverzeichnis 23](#_Toc518406337)

# Einleitung

Diese, in deutscher Sprache verfasste, Systemdokumentation basiert auf jener von Gruppe 3, im weiteren Verlauf der Dokumentation als „Team“ bezeichnet, für „PR Software Engineering“, erstellten App. Dabei handelt es sich um ein „Task Management und Time Tracker“ Tool, im weiteren Verlauf der Dokumentation als „TimeTrackerApp“ bezeichnet.

## Aufgabenstellung

### Inhalt

Folgende Funktionalitäten (durch die LVA Leitung vorgegeben) sollen zur Verfügung gestellt werden:

* Erfassen von Projekten, Aufgabenbereichen und Aufgaben
* Aufgabenbereiche als Swimlanes mit Aufgaben darstellen
* Erfassen von Sollzeiten mit Fortschrittsanzeige
* Zuordnung von Aufgaben zu Personen
* Erfassen der Zeit zu bestimmten Aktivitäten, Zuordnung zu Aufgaben
* Definition eines Rundungsintervalls
* Erfassen strukturierter Notizen zu jedem erfassten Eintrag
* Nachträgliches Ändern von Datum, Zeit und Dauer
* Status zu Aufgaben anzeigen
* Reporting für ein gesamtes Projekt, für unterschiedliche Aufgabenbereiche, Personen, etc.
* Synchronisation über mehrere Rechner
* Export (.csv, JSON)

### Motivation

Konkret wird also eine App erzeugt, welche das Management von Projekten, speziell Aufgaben, erleichtern soll. Dazu stellt diese Funktionen zum Erstellen von Projekten, Aufgabenbereichen und Aufgaben bereit. Außerdem ist es möglich, Zeitmanagement für die einzelnen Aufgaben, unter Berücksichtigung beteiligter Personen, zu betreiben. Die App vereint all diese Funktionen in einer Software, welche einfach zu bedienen sein soll, und ein effizientes Arbeiten damit ermöglicht.

Wir entwickeln sie mit dem Hintergedanken, sie selbst alsbald für unser Projekt nutzen zu können, und somit die Zeiterfassung als Excel Dokument ablösen zu können.

## Systemanforderungen

Da das Programm mithilfe von Java entwickelt wurde, lässt sich diese auf einer Vielzahl an System ausführen, sofern Java von jenem unterstützt wird. Die minimalen Systemanforderungen setzen sich somit wie folgt zusammen:

Betriebssystem: Microsoft Windows 7 oder höher

Java 8 oder höher

Prozessor: Intel Core 2 Duo, 1GHz (oder vergleichbar)

Arbeitsspeicher: 1GB oder mehr

Festplatte: 500MB freier Festplattenspeicher

Internetverbindung: 5Mbit/s oder schneller

## Qualitätsziele

Nachfolgend werden die Qualitätsziele in tabellarischer Ansicht angeführt. Diese wurden zum Teil von der LVA Leitung, durch die Beurteilungskriterien, indirekt vorgegeben. Die Reihenfolge spiegelt die Relevanz der Ziele, welche durch das Team festgelegt wurde, wieder.

|  |  |
| --- | --- |
| **Qualitätsziel** | **Beschreibung** |
| Effizienz | Die App arbeitet effizient, in erster Linien betrifft dies den Programmcode. Zusätzlich soll eine effiziente Benutzung der App selbst möglich sein. |
| Stabilität | Die App arbeitet stabil, sprich sie liefert die Ergebnisse, welche der Benutzer, in Form von Erwartungen an sie stellt. Fehlerhafte Ausgaben sollten soweit eliminiert werden, sodass die Benutzung durch den Anwender reibungslos erfolgen kann. |
| Programmcode | Ist Teil des Qualitätsziels „Effizienz“. Der Code sollte schlank sein, lösungsorientiert und effizient arbeiten. Redundanzen im Code sollen weitestgehend, soweit möglich, entfernt werden. |
| Benutzeroberfläche | Die Benutzeroberfläche sollte selbsterklärend, und ohne Bedienungsanleitung möglich sein. Ebenso soll diese schlank, nicht mit Menüs überladen, und einfach strukturiert sein. |
| Sicherheit | Die App lässt sich ohne Sicherheitsrisiken verwenden. Verbindungen, welche zu Datenbanken hergestellt werden, basieren auf einer SSL-Verbindung, welche eine sichere Datenübertragung gewährleisten soll. |

## Stakeholder

Nachfolgend werden die Stakeholder, welche Einfluss auf das Projekt und die Entwicklung dessen haben, angeführt.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Kontakt** | **Funktion** | **Erwartungshaltung** |
| Rainer Weinreich | rainer.weinreich@gmail.com | LVA Leiter | * *siehe Vorbesprechung* |
| Benjamin Mayer | benjamin.mayer@jku.at | LVA Leiter | * *siehe Vorbesprechung* |
| Aistleithner Andrea | aistleithner.andrea97@gmx.at | Projektmitarbeiter | * sinnvolle Testung der App, um Fehler frühzeitig zu erkennen und um gegensteuern zu können * Erzeugung einer fehlerfreien App |
| Dusanic  Maja | majadusanic@gmail.com | Projektmitarbeiter | * sinnvolle Nutzung des SCR für die Entwicklung der Applikation und zum Management des Programmcodes |
| Huber  Christoph | christoph.huber91@gmail.com | Projektmitarbeiter | * effiziente Planung des Projekts, sowie der Entwicklung der App * effizientes Issuemanagement, welches die Fehler aus den Tests aufschlüsselt |
| Teuchtmann Alexander | a.teuchtmann@alteutech.at | Projektmitarbeiter | * sinnvolle Dokumentation des Entwicklungsfortschritts und der Anforderungen * verständliche Dokumentation des Systems (Systemdokumentation) |
| Tomic  Milos | tomicmilos94@yahoo.at | Projektmitarbeiter | * die Erzeugung von qualitativ hochwertigem Programmcode, welcher den Coderichtlinien entspricht |

# Randbedingungen

Grundsätzlich sind keine Randbedingungen, bis auf die geforderten Funktionalitäten, gegeben. Durch die Zusammensetzung des Teams ergeben sich jedoch bestimmte Einschränkungen. Ebenso hat das Team selbst Bedingungen festgelegt.

Organisatorische Einschränkung

|  |  |  |
| --- | --- | --- |
| **Einschränkung** | **Erklärung** | **Quelle** |
| Namenskonventionen | Die Namenskonventionen schränken den Gebrauch von Namen bei der Entwicklung ein. Sie stellen sicher, dass einheitliche Namen verwendet werden. | Code Richtlinien  Seite 15-16 |
| Coderichtlinien | Die Coderichtlinien schränken generell den Programmierstil ein. Es werden Richtlinien aufgestellt, welche die Qualität des Codes sicherstellen sollen. | Code Richtlinien |
| zeitliche Frist | Durch die LVA Leitung wurden zeitliche Fristen festgelegt (Meilensteine), welche einzuhalten sind. | Vorbesprechung |

Personelle Einschränkungen

|  |  |
| --- | --- |
| **Einschränkung** | **Erklärung** |
| Programmiersprache | Durch die Einschränkung, dass der Großteil des Teams nur die Programmiersprache Java beherrscht, wurde diese auch für das Projekt als Grundlage gewählt. Gleiches gilt für die Benutzung einer (my)SQL Datenbank für die Datenspeicherung. |

# Kontextabgrenzung

## Fachlicher Kontext

Nachfolgend sind alle Kommunikationspartner bezüglich der Benutzung der TimeTrackerApp aufgelistet. Ebenso finden sich hier die Ein- und Ausgabedaten, sowie die genutzten Schnittstellen und Kommunikationskanäle, wieder.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Partner** | **Eingabe** | **Ausgabe** | **Schnittstellen** | **Kanal** |
| Benutzer1 | über die Tastatur | - | Arbeitsplatz | direkt |
| Arbeitsplatz2 | vom Benutzer | an die Instanz | Instanz | direkt |
| Instanz3 | vom Benutzer | über die View | Instanz, Datenbank | direkt |
| Datenbank4 | über die Instanz | über die Instanz | Instanz | HTTP-Verbindung |

1. Benutzer:

Der Benutzer bedient eine Instanz der TimeTrackerApp über den Arbeitsplatz. Er ist sozusagen der Endbenutzer, welcher die App später verwendet. Mithilfe der Tastatur/Maus des Arbeitsplatzes werden Eingaben getätigt.

2. Arbeitsplatz:

Dieser stellt ein Endgerät des Benutzers dar, und kann ein Laptop oder ein Desktopcomputer sein. Über den Arbeitsplatz wird eine Instanz der TimeTrackerApp ausgeführt. Die Eingaben werden vom Benutzer erhalten, die Ausgabe erfolgt an die Instanz.

3. Instanz:

Als Instanz wird hier die Ausführung der TimeTrackerApp verstanden, sobald diese an einem Arbeitsplatz ausgeführt wird. Eingaben werden vom Benutzer, durch den Arbeitsplatz, erhalten. Die Ausgabe der Daten erfolgt dann über die View. Die Instanz kommuniziert desweiteren mit der Datenbank

4. Datenbank:

Sie stellt die letzte Stelle dar. Die Instanz kommuniziert über eine HTTP-Verbindung mit der Datenbank, liest und schreibt demnach in die Datenbank. Ein- und Ausgabe erfolgt über die laufende Instanz der TimeTrackerApp.

## C:\Users\ateuc\AppData\Local\Microsoft\Windows\INetCache\Content.Word\technische_Schnittstelle.pngTechnischer Kontext

Abbildung 1 Technische Schnittstellen des Projekts

|  |  |  |
| --- | --- | --- |
| Kanal | Eingabe | Ausgabe |
| TimeTrackerApp | SQL Statements | über die View dargestellte Daten |
| Datenbank | SQL Statements | Daten |

# Lösungsstrategie

Dieses Kapitel bietet Einsicht in die grundlegenden Entscheidungen und Lösungsansätze, welche die TimeTrackerApp prägen. Nachfolgend werden diese kurz erklärt, genauere Ausführungen sind in den Folgekapiteln zu finden.

**Technologieentscheidungen**

Aufgrund der zuvor bereits beschriebenen personellen Einschränkungen des Teams bezüglich der Fähigkeiten im Umgang mit Programmiersprachen, wurden folgende Sprachen für die Umsetzung des Projekts ausgewählt.

* Java

Aufgrund der weiten Verbreitung von Java in der Programmier-, sowie Anwendungswelt und der hohen Anzahl an unterstützten Geräten, wurde diese Programmiersprache als Grundbasis für das Projekt gewählt. Java kommt weltweit auf den unterschiedlichsten Endgeräten, sowohl im privaten, als auch im unternehmerischen Umfeld zum Einsatz. Außerdem wird mit Java eine Programmiersprache geboten, welche objektorientiert, robust, portabel und sehr leistungsfähig ist. Desweiteren verfügen alle Teammitglieder über Kenntnisse in Java, durch die Absolvierung der Lehrveranstaltungen „Einführung in die Softwareentwicklung“ und „Softwareentwicklung 2“, wodurch sich diese für das Projekt sehr eignet. [java.com, 2018]

* mySQL Datenbank

Im späteren Verlauf des Projekts wurde vom Team festgelegt, eine mySQL Datenbank für die persistente Datenspeicherung zu verwenden, da diese sowohl schlank, als auch effizient arbeitet und bei den weltweit größten Dienstleistungsanbietern (Facebook, Google, etc.) zum Einsatz kommt. Zudem waren durch die Absolvierung der Lehrveranstaltung „Datenmodellierung“ bei den Teammitgliedern bereits Kenntnisse im Umgang mit SQL vorhanden. [mysql.com, 2018]

**Entwurfs- und Architekturmuster**

Das für die TimeTrackerApp verwendete Architekturmuster kann als Client-Server-Architektur verstanden werden. Wie im späteren Verlauf der Dokumentation beschrieben, unterteilt sich das System grundsätzlich in einen Client (die Ausführungseinheit der TimeTrackerApp) und einen Server (Datenbank), welche miteinander kommunizieren.



Abbildung 2 Architektur der TimeTrackerApp

Der Entwurf der TimeTrackerApp kann wie folgt beschrieben werden. Zuerst besteht die „View“, welche für die Darstellung der TimeTrackerApp an sich, und der jeweiligen Daten verantwortlich ist. Diese wird von den „Models“, also den Objekten in ihrer jeweiligen Instanz verändert. Die Objekte erhalten und schreiben ihre Daten und in die „DB\_\*“ Methoden.



Abbildung 3 Entwurf der TimeTrackerApp

**Erreichung der Qualitätsanforderungen**

Nachfolgend werden die zuvor definierten Qualitätsanforderungen hinsichtlich der Erreichung analysiert.

* Effizienz

Einerseits soll die TimeTrackerApp effizient arbeiten, das heißt, dass man ohne Probleme durch die Menüs navigieren kann, und schnell und effizient seine Aufgaben mithilfe der App erledigen kann. Eine einfache und unkomplizierte Benutzung soll durch eine einfache und selbsterklärende Menüführung und durch einen generell sehr schlichten Programmaufbau garantiert werden.

* Stabilität

Durch ein effizientes Management sollen Fehler frühzeitig im Programm erkannt, und eliminiert werden, damit der spätere Benutzer einen reibungslosen Betrieb erwarten kann. Durch umfangreiche Tests an der App sollen diese Fehler gefunden, und später behoben werden.

* Programmcode

Der Programmcode der TimeTrackerApp soll mithilfe des Teams zuvor festgelegten Coderichtlinien sauber, effizient und natürlich den Richtlinien entsprechend aufgebaut werden. Zusätzlich wird der Programmcode durch Codereviews, welche durch eine Checkliste abgearbeitet werden, kontrolliert.

* Benutzeroberfläche

Wie bereits zuvor erwähnt ist eine einfache und selbsterklärende Benutzeroberfläche ein wichtiges Qualitätsziel. Der spätere Benutzer sollte in der Lage sein, die TimeTrackerApp ohne Bedienungsanleitung benutzen zu können. Dies soll durch eine gute Menüführung erreicht werden.

* Sicherheit

Dieses Ziel wurde im Verlauf des Projekts nicht mehr realisiert. Geplant wäre gewesen, die Datenbankverbindung, welche auf einer unverschlüsselten http-Verbindung basiert, durch eine SSL-Verbindung zu ersetzen.

**Organisatorische Entscheidungen**

Die Organisation des Projekts bzw. die Aufgabenverteilung innerhalb des Teams an die Projektmitarbeiter wurde wie folgt eingeteilt.

* Aistleithner Andrea

Testen der Anwendung

* Maja Dusanic

Verwaltung des Source Code Repository

* Christoph Huber

Requirements- und Issuemanagement, Planung

* Alexander Teuchtmann

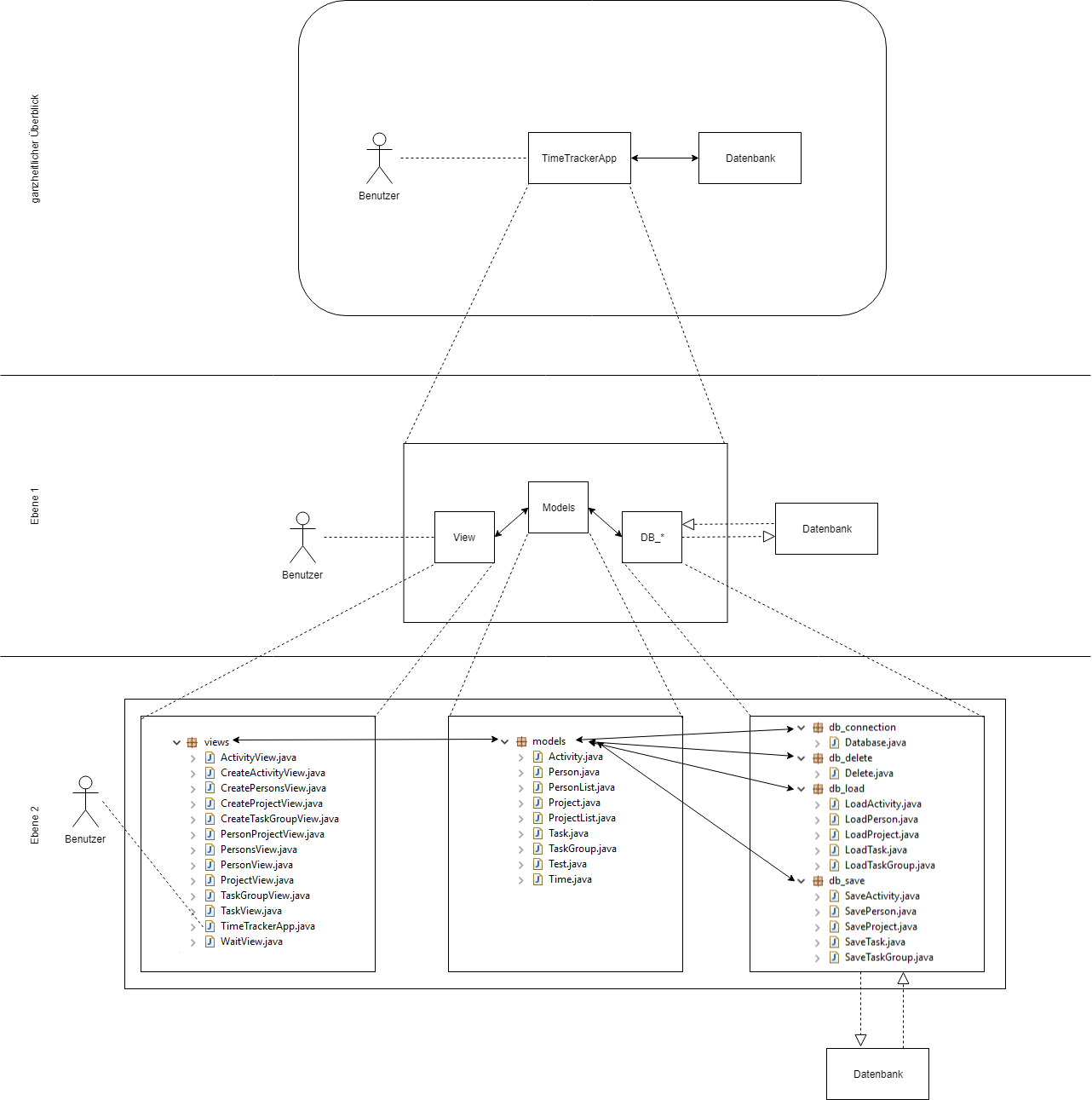
Architektur und Dokumentation

* Milos Tomic

Überprüfung Code Qualität

# Bausteinsicht

Die nachfolgende Grafik zeigt das System im ganzheitlichen Überblick (0) in der obersten Ebene. Die „Ebene 1“ zeigt den Aufbau der TimeTrackerApp und gliedert diese in die „View“, „Models“ und „DB\_\*“. Die „Ebene 2“ zeigt dann die Zerlegung der Bausteine in die einzelnen Packages, samt den dazugehörigen Klassen.

Abbildung 4 Systemzerlegung in Bausteine

## Zerlegung des Gesamtsystems

Nachfolgend wird das zuvor beschriebene Gesamtsystem heruntergebrochen und die einzelnen Ebenen erklärt.

### C:\Users\ateuc\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Bausteine_Ebene0.pngGanzheitlicher Überblick

Abbildung 5 Ganzheitlicher Überblick des Systems

Im Fokus steht bei dieser Sicht die Betrachtung des gesamten Systems. Hierbei wird zunächst nur die TimeTrackerApp als gesamtes System betrachtet. Diese kommuniziert über eine unverschlüsselte HTTP-Verbindung mit der Datenbank.

Der Benutzer selbst agiert mit der TimeTrackerApp, er merkt von der Kommunikation zur Datenbank selbst nichts und muss sich nicht darum kümmern.

### C:\Users\ateuc\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Bausteine_Ebene1.pngEbene 1

Abbildung 6 Ebene 1 der Bausteinsicht

Nun wird die TimeTrackerApp in den aktuellen Entwurf gegliedert dargestellt. Der äußere Rahmen stellt die TimeTrackerApp dar. Diese ist gegliedert in die „View“, die „Models“ und die „DB\_\*“, welche nach außen hin für die Kommunikation mit der Datenbank zuständig ist. Der Benutzer selbst agiert lediglich mit der View.

Die „View“ ist für die Darstellung der TimeTrackerApp verantwortlich. Sie stellt im Prinzip den Inhalt der „Models“ dar.

Die „Models“ stellen die konkreten Instanzen der Modelle dar. Hier werden die Objekte erstellt, mit Inhalt befüllt und an die „View“ zur Darstellung zurückgegeben.

Die „DB\_\*“ stellt die einzelnen Datenbankoperationen dar. Diese Operationen interagieren mit der Datenbank, sie lesen und schreiben die Daten daraus und geben sie an die „Models“ zurück.

### C:\Users\ateuc\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Bausteine_Ebene2.pngEbene 2

Abbildung 7 Ebene 2 der Bausteinsicht

In dieser Ebene werden die in Ebene 1 beschriebenen Bausteine nochmals näher betrachtet. Der äußere Rahmen stellt wieder die TimeTrackerApp dar. Diese ist wie in der vorherigen Ebene in die „views“, „models“ und die „db\_\*“ eigeteilt.

Die „views“ werden hier in das konkrete Package, mit den dazugehörigen Klassen unterteilt. Die Klassen sind für die Darstellung der Projekte (Projects), Personen (Persons), Aufgabenbereiche (TaskGroup) und der Zeiterfassung (Activities) zuständig. Der Benutzer selbst greift beim Start auf die Klasse „TimeTrackerApp.java“ zu.

Die „models“ werden hier ebenso in das konkrete Package, mit den dazugehörigen Klassen unterteilt. Die Klassen stellen dabei die einzelnen Instanzen (wie oben gezeigt) dar. Hier werden die späteren Objekte angelegt, verändert und wieder gelöscht.

Das „db\_connection“ Package ist für den Aufbau der Datenbankverbindung zuständig. Hier sind die Verbindungsdaten hinterlegt. „db\_load“ ist für das Laden von Informationen aus der Datenbank zuständig, „db\_save“ für das Speichern von Daten in die Datenbank. Mithilfe von „db\_delete“ werden Informationen aus der Datenbank gelöscht. Alle diese Packages sind an der Kommunikation mit der Datenbank selbst beteiligt.

# Laufzeitsicht

In diesem Kapitel wird anhand von einem Szenario die grundsätzliche Funktionalität der Bausteine beschrieben. Genauer gesagt wird anhand der Szenarien der Ablauf innerhalb der Bausteine erklärt.

## Szenario

Das erste Szenario handelt von der Erstellung eines Projekts, und der Hinzufügung einer Projektbeschreibung.

Zuerst wird vom Benutzer die „views/TimeTrackerApp“ aufgerufen. Diese erstellt dann eine neue Instanz der „views/CreateProjectView“. Sie ist für die Darstellung der Projekte zuständig.

Infolgedessen, wird während des Erstellens eines neuen Projekts ein neues „models/Project“ Objekt angelegt. Dieses wird dann mit Daten aus der Datenbank befüllt.

Zunächst wird „db\_save/SaveProject“ aufgerufen, welche wiederum mithilfe der „db\_connection/Database“ eine Verbindung zur Datenbank initialisiert und danach die Daten in die Datenbank speichert.

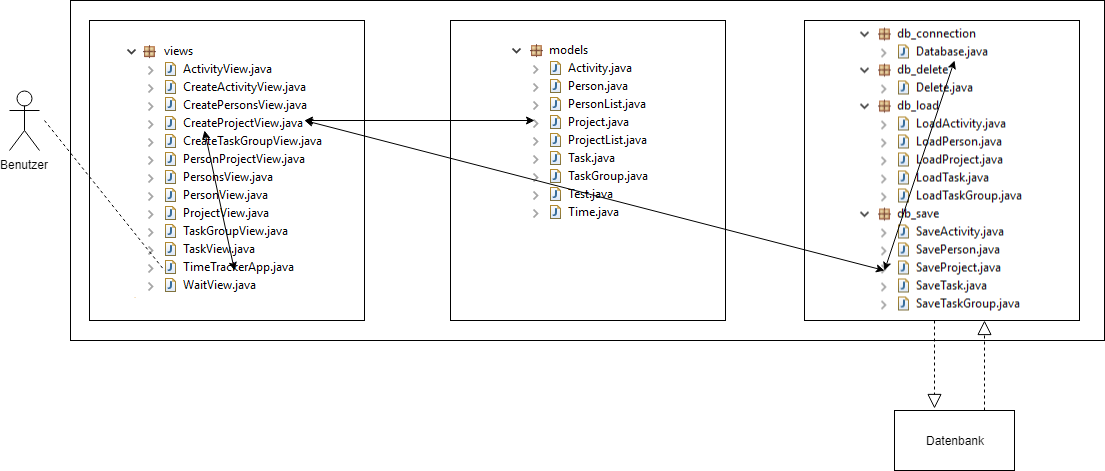


Abbildung 8 Szenario während der Laufzeitsicht

# Verteilungssicht

Die Verteilung der einzelnen Bausteine ist im Falle der TimeTrackerApp auf zwei unterschiedliche Systeme beschränkt. Die Ausführungseinheit der TimeTrackerApp befindet sich auf einem oder mehreren Arbeitsplätzen, welche unterschiedliche Endgeräte darstellen können (Laptop, Desktop). Das zweite System stellt die konsistente Datenspeicherung dar, sie erfolgt in einer extern gehosteten Datenbank auf einem Drittsystem, auf welches von den Arbeitsplätzen zugegriffen wird.

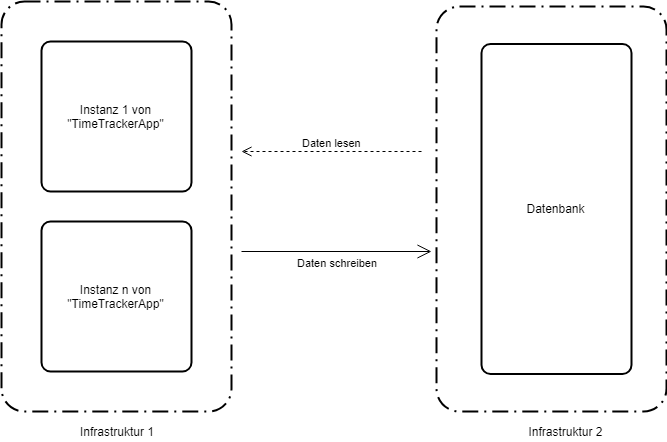


Abbildung 9 Verteilung der Infrastruktur

## Konkrete Verteilung

**Infrastruktur 1**

Im Verständnis dieser Arbeit haben wir die Infrastruktur 1 wie folgt definiert. Sie besteht im Grunde genommen aus der Ausführungseinheit der TimeTrackerApp. Dies kann ein mobiles Endgerät (zum Beispiel ein Laptop) oder auch ein Desktopcomputer sein. Da die Infrastruktur dieser Geräte im weitesten Sinne gleich oder ähnlich sind (logischer Aufbau), wurden diese aufgrund der Einfachheit in einer Infrastruktur zusammengefasst. Im eigentlichen Sinne wäre jedes Gerät, auf welchem eine Instanz der TimeTrackerApp läuft, eine eigene Infrastruktur.

**Infrastruktur 2**

Diese stellt das System, auf welchem die Datenbank gespeichert ist, dar. Ausgehend von der Infrastruktur 1, werden hier Daten konsistent gespeichert, und bei Bedarf abgerufen. Die Datenbank wird auf einem externen Server gehostet, welcher administrativ vom Dienstleistungsanbieter zur Gänze verwaltet wird. Die Installation, Verwaltung und Pflege der Datenbank selbst wird vom Team ausgeführt.

## Begründung der Verteilung

**Infrastruktur 1**

Um unabhängig von jeweiligen Endgerät zu sein, wurde auf ein universell einsetzbares Programmsystem (Java) gesetzt, welches ohne aufwändige Vorkehrungen auf fast jedem zuvor beschriebenen Benutzergerät zum Einsatz kommen kann, sofern eine aktuelle Version von Java auf jenem Gerät installiert ist. Somit ist die Benutzung der TimeTrackerApp nahezu ortsungebunden (sofern eine Internetverbindung vorhanden ist).

**Infrastruktur 2**

Um eine dauerhafte, und ebenso ortsunabhängige Datenspeicherung und -abrufung möglich zu machen, wurde dafür eine extern gehostete Datenbank verwendet. Diese ist mit 99%iger (laut Dienstleister) Wahrscheinlichkeit immer verfügbar und im Falle des Datenverlusts können Backups problemlos eingespielt werden.

# Querschnittliche Konzepte

In diesem Kapitel werden alle Konzepte angeführt und beschrieben, welche während des Projekts zur Erstellung der TimeTrackerApp notwendig waren, vom Team festgelegt wurden, oder als Vorgabe durch die LVA Leitung gegeben waren.

## Fachliche Konzepte

Die grundsätzliche Funktion der TimeTrackerApp ist das Management von Projekten, sowie deren Aufgaben und Zeiterfassung. Die genauere Aufführung der Funktionen findet sich in der „Benutzerdokumentation“ wieder.

### Personen

Sind unabhängig von allen weiteren Konstrukten. Sie können separat angelegt und auch wieder gelöscht werden. Sie bestehen aus einem Vor- und Nachnamen.

### Projekte

Es können Projekte erstellt werden. Diese umfassen alle weiterfolgenden Kategorien. Zu Projekten können Personen zugeordnet werden, Projekte können mehrere Aufgabenbereiche, Aufgaben und Activities beinhalten. Projekte können erstellt, bearbeitet (Name, Beschreibung) und gelöscht werden.

### Aufgabenbereiche

Sind einem Projekt untergeordnet und können mehrere Aufgaben beinhalten. Hier kann ein Aufgabenbereichsname vergeben werden. Auch können Aufgabenbereiche wieder gelöscht werden.

### Aufgaben

Unterliegen direkt einem Aufgabenbereich und können eine Aufgabenbeschreibung, sowie -namen beinhalten. Zusätzlich lässt sich hier eine Sollzeit eintragen. Ebenso ist ein Fortschrittsbalken ersichtlich, welcher die Differenz von Soll- und Istzeit anzeigt. Auch können Aufgaben wieder gelöscht werden.

### Activities

Stellen die konkrete Zeiterfassung dar. Hier kann für einen erfassten Eintrag eine Beschreibung, sowie ein Zeitfenster (Start und Ende) vergeben werden. Außerdem kann einer Aufgabe eine Person zugewiesen werden. Nachträglich können Einträge wieder gelöscht werden.

## User Experience

Die grafische Oberfläche der TimeTrackerApp wurden bewusst sehr einfach, und übersichtlich gehalten. Alle zuvor beschriebenen Konzepte (ausgenommen Aufgaben) werden in einem eigenen Fenster geöffnet, um die Übersichtlichkeit zu wahren.

Desweiteren werden Menüs zum Ändern bestimmter Eigenschaften (z.B.: Name) wiederum in einer eigenen Box dargestellt.

Als Designfarben wurden Grautöne gewählt, da diese als neutral angesehen werden. Das Team empfindet, dass damit am angenehmsten gearbeitet werden kann.

Die genauere Aufführung des Designs findet sich in der „Benutzerdokumentation“ wieder.

## Architektur- und Entwurfsmuster

Dieses Konzept wurde bereits in Kapitel 4 (Lösungsstrategie) diskutiert. Der Vollständigkeit halber wurde der Punkt erneut unter den querschnittlichen Konzepten aufgeführt, wird hier aber nicht nochmals näher beschrieben.

## Entwicklungskonzepte

Als Entwicklungskonzept wurde das in Kapitel 4 (Lösungsstrategie) dargestellte Konzept angewendet.

Um das Entwicklungskonzept nachhaltig zu überprüfen, wurden im Laufe der Entwicklung mehrere Codereviews durchgeführt. Die Vorlage für diese Reviews wurde zuvor im Team festgelegt, und als Checkliste ausgearbeitet.

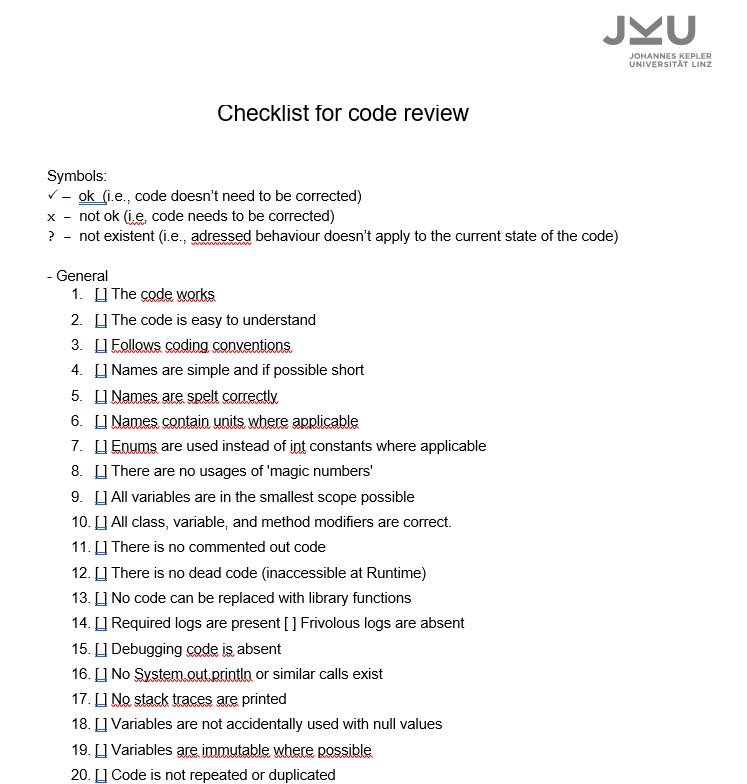


Abbildung 10 Auszug aus der Checkliste zum Codereview

Das Codedesign folgt den zuvor recherchierten Code Richtlinien. Diese Code Richtlinien sollen für einen einheitlichen Programmcode sorgen.

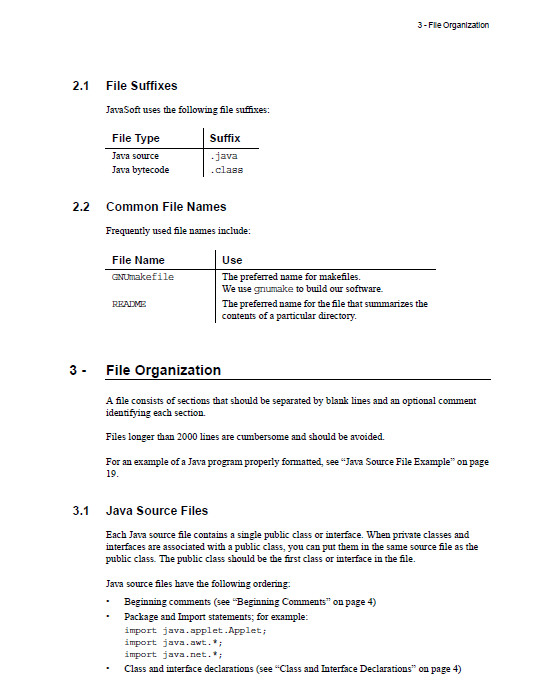


Abbildung 11 Auszug aus den Coderichtlinien

Außerdem wurde im gleichen Zyklus Unit Tests anhand des vorhandenen Codes durchgeführt. Diese garantieren für erwartete Ergebnisse. Es werden erwartete, mit resultierenden Ergebnissen vergleichen.

# Risiken und technische Schulden

Nachfolgend wurden die Risiken analysiert, welche im Gebrauch der TimeTrackerApp entstehen können. Wie bereits zuvor erwähnt, wurden der Sicherheitsaspekt der App in diesem Projekt nicht (mehr) realisiert.

## Verbindungsaufbau zwischen App und Datenbank

Bisher basiert der Verbindungsaufbau zwischen der TimeTrackerApp und der extern gehosteten Datenbank auf einer unverschlüsselten HTTP-Verbindung. Diese Verbindung könnte aufgrund der fehlenden Verschlüsselung abgehört, und Daten mitgelesen werden. Dies könnte durch die Verwendung einer SSL-Verbindung verhindert (erschwert) werden.

Grundsätzlich bietet die Library „mysql-connector“ (Connector/J) die Möglichkeit, diese Verbindung mithilfe von SSL herzustellen. Der Provider, welche die Datenbank zur Verfügung stellt, bietet jedoch offensichtlich noch keine Möglichkeit, eine derartige Verbindung herzustellen, da beim Testen der TimeTrackerApp immer Hinweise auf eine unsichere Verbindung gegeben wurden, und SSL daher nicht verwendet wird.

# Glossar

Dieses Glossar umfasst alle Begriffe und Abkürzungen, welche, im Zusammenhang mit in diesem Praktikum entstandener Systemdokumentation, verwendet wurden. Es soll ein einheitliches Verständnis für alle Begrifflichkeiten schaffen, und Redundanzen, in der Verwendung von mehreren Begriffen für ein Element, eliminieren.

Projektinterne Bezeichnungen

|  |  |
| --- | --- |
| **Begriff** | **Definition** |
| Projekt | Name für die in diesem Praktikum durchgeführten Aktionen |
| Team | Name für die „Gruppe 3“ des Praktikums |
| TimeTrackerApp | Name für die in diesem Praktikum erstellten Applikation |

Abkürzungsverzeichnis

|  |  |
| --- | --- |
| **Abkürzung** | **Definition** |
| .csv | Comma-separated values (Dateiformat) |
| App | Applikation |
| SCR | Source Code Repository |

Begriffserklärungen

|  |  |
| --- | --- |
| **Begriff** | **Definition** |
| Java | objektorientierte Programmiersprache |
| mySQL | Datenbankverwaltungssystem |
| SQL | Datenbanksprache für relationale Datenbanken |
| JSON | JavaScript Object Notation (Dateiformat) |

# Quellenverzeichnis

[java.com, 2018]

offizielle Webseite von Oracle Java; Zugriff: 25.06.2018

<https://java.com/de/about>

[mysql.com, 2018]  
 offizielle Webseite von MySQL; Zugriff: 25.06.2018

<https://www.mysql.com/de/why-mysql/#de-0-0>

# Abbildungsverzeichnis

Seite 8

Abbildung 1 Technische Schnittstellen des Projekts

Seite 9

Abbildung 2 Architektur der TimeTrackerApp

Seite 10

Abbildung 3 Entwurf der TimeTrackerApp

Seite 12

Abbildung 4 Systemzerlegung in Bausteine

Seite 13

Abbildung 5 Ganzheitlicher Überblick des Systems

Seite 14

Abbildung 6 Ebene 1 der Bausteinsicht

Seite 15

Abbildung 7 Ebene 2 der Bausteinsicht

Seite 16

Abbildung 8 Szenario während der Laufzeitsicht

Seite 17

Abbildung 9 Verteilung der Infrastruktur

Seite 20

Abbildung 10 Auszug aus der Checkliste zum Codereview

Seite 21

Abbildung 11 Auszug aus den Coderichtlinien