Architekturentwurf

Projektbezogene Zeiterfassung

Von Lukas Panni

Version 1.0 - 11.06.2020

Versionstabelle

Versionsnummer	Autor	Änderungsvermerk	Zu Anforderungsdokument
1.0 (11.06.2020)	Lukas Panni	Initiale Fassung	Version 1.0 (02.12.2019)

Inhalt

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Vielen Angestellten fällt es schwer einen Überblick über ihre Arbeitszeiten in verschiedenen Projekten zu bewahren. Häufig müssen diese Arbeitszeiten allerdings für jedes Projekt individuell erfasst und dokumentiert werden.

Die projektbezogene Zeiterfassung für Angestellte soll sehr einfach gestaltet werden, sodass die Angestellten sich besser auf ihre eigentlichen Aufgaben konzentrieren können.

Dazu soll im Rahmen dieses Projektes eine Anwendung entwickelt werden, die projektbezogene Zeiterfassungen, sowie den Überblick über geleistete Arbeitszeiten für Angestellte erleichtert.

Nachdem im Anforderungsdokument in Version 1.0.0 vom 02.12.2019 die Anforderungen an ein solches System erfasst und genauer spezifiziert wurden, soll in diesem Dokument die Architektur einer Webanwendung entworfen werden, die diese Anforderungen umsetzt.

Das Hauptziel dieses Architekturentwurfs ist es, die genannten Probleme möglichst effizient und einfach zu lösen. Das System soll so entworfen werden, dass Weiterentwicklung sowie spätere Änderungen aufgrund von neuen Anforderungen möglich sind. Dazu sollen die Komponenten des Systems untereinander möglichst lose gekoppelt und leicht austauschbar gestaltet sein. Dazu sollen einzelne Komponenten auch möglichst nur eine Aufgabe übernehmen um die Wartbarkeit des Systems und auch die Lesbarkeit, des in der Implementierungsphase zu schreibenden Quellcodes zu verbessern. Da die Einfachheit der Anwendung als eine der wichtigsten nichtfunktionalen Anforderungen identifiziert wurde soll im Entwurf aber vor Allem auch in der anschließenden Implementierung darauf geachtet werden, keine überflüssigen Funktionalitäten anzubieten. Dadurch würde die Komplexität des Systems nur weiter steigen und damit die Forderung nach Einfachheit und effizienter Bedienung nicht erfüllen. Außerdem ist das System so zu entwerfen, dass es vom Betriebssystem unabhängig funktionsfähig sein soll.

2 Statische Sicht – zu persistierende Daten

Als erste Sicht werden im Folgenden die zu persistierenden Daten beschrieben, da das Datenmodell auch bei möglichen Weiterentwicklungen seine Gültigkeit behalten soll. Mögliche Änderungen oder Weiterentwicklungen sollen auf das bestehende Datenmodell aufsetzen und eventuell zusätzliche Datenobjekte hinzufügen, aber das bestehende Modell nicht verändern um die Kompatibilität zu erhalten.

2.1 Entity-Relationship-Diagramm

Das nachfolgende ER-Modell zeigt alle von dem System zu persistierenden Daten und ihre Beziehungen.

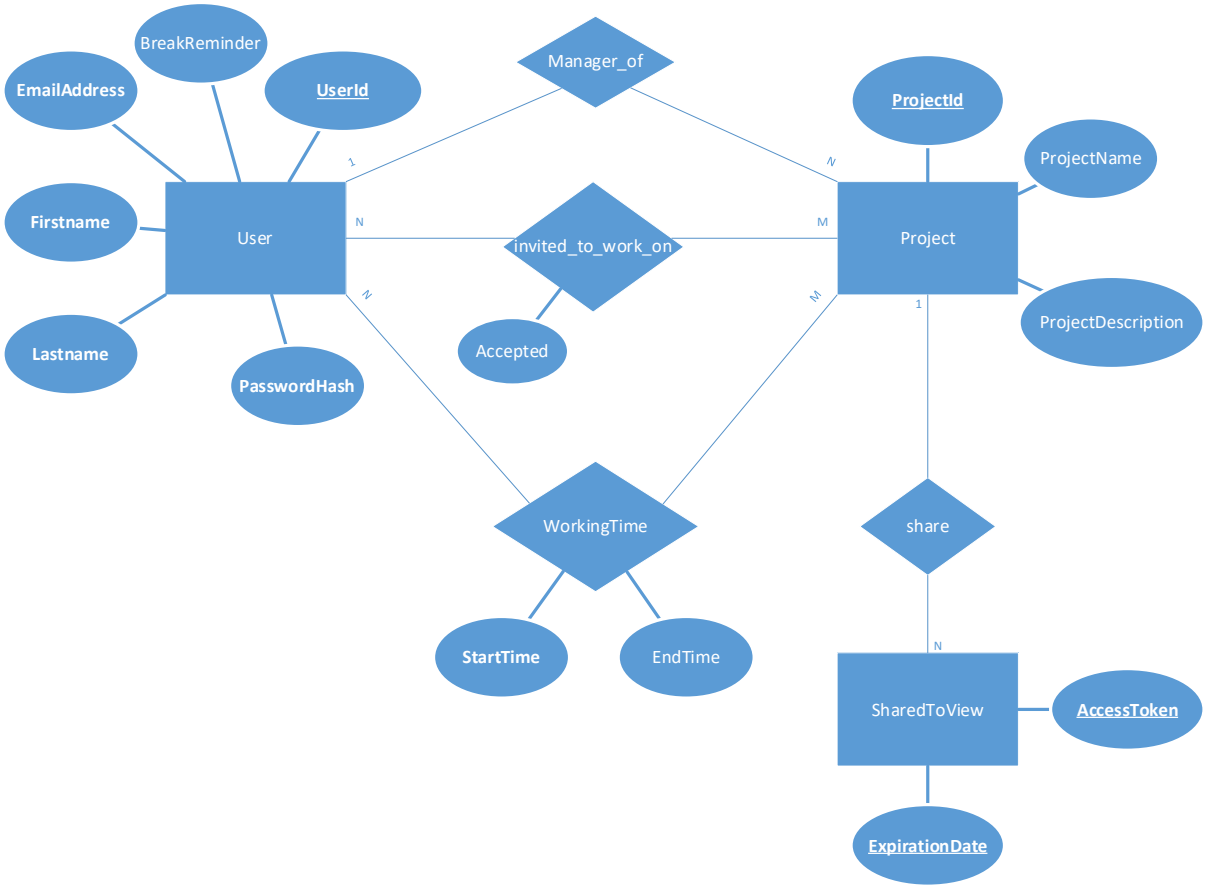

Abbildung 1: Entity-Relationship-Model

Für das System sind die Entitätstypen *User* und *Project* von zentraler Bedeutung. Der Entitätstyp *SharedToView* wird als Hilfstyp eingeführt, um die Anforderung der Zeitübersichtsfreigabe umzusetzen.

Die Beziehungen *invited_to_work_on* und *WorkingTime* sind ebenfalls von zentraler Bedeutung für das System, da diese beiden Beziehungen die Erfassten Arbeitszeiten für Projekte und die Angenommenen beziehungsweise Abgelehnten Projekteinladungen repräsentieren.

Dabei wurden folgende Entwurfsentscheidungen getroffen:

- Die User-Authentifizierung erfolgt über die E-Mail Adresse und ein Passwort, das als Hash gespeichert wird.
- Aufgezeichnete Arbeitszeiten werden als Kombination eines Start- und eines Endzeitpunkts dargestellt.
- Arbeitszeitfreigaben werden über ein AccessToken und ein Ablaufdatum identifiziert.

Für die Umsetzung wurde ein relationales Datenbankmodell gewählt, um eine strukturierte Speicherung der Daten zu gewährleisten.

Datentypen

Nach der Transformation des Entity-Relationship-Modells in ein relationales Datenbankmodell ergeben sich folgende Relationen mit Attributen, Datentypen und Integritätsbedingungen:

User		
Attribut	**Datentyp**	**Integritätsbedingungen**
UserId	INT	NOT NULL, PRIMARY KEY
EmailAddress	VARCHAR(255)	NOT NULL, UNIQUE
Firstname	VARCHAR(255)	NOT NULL
Lastname	VARCHAR(255)	NOT NULL
PasswordHash	VARCHAR(255)	NOT NULL
BreakReminder	INT	NOT NULL

Tabelle 1: Relation User mit Attributen, Datentypen und Integritätsbedingungen

Project		
Attribut	**Datentyp**	**Integritätsbedingungen**
ProjectId	INT	NOT NULL, PRIMARY KEY
ProjectName	VARCHAR(255	NOT NULL
ProjectDescription	VARCHAR(31)	
ProjectManager	INT	NOT NULL, FOREIGN KEY REFERENCES User.UserId

Tabelle 2: Relation Project mit Attributen, Datentypen und Integritätsbedingungen

WorkingTime		
Attribut	**Datentyp**	**Integritätsbedingungen**
StartTime	TIMESTAMP	NOT NULL
EndTime	TIMESTAMP	
ProjectId	INT	NOT NULL, FOREIGN KEY REFERENCES Project.ProjectId
UserId	INT	NOT NULL, FOREIGN KEY REFERENCES User.UserId

Tabelle 3: Relation WorkingTime mit Attributen, Datentypen und Integritätsbedingungen

Invited_to_work_on		
Attribut	**Datentyp**	**Integritätsbedingungen**
ProjectId	INT	NOT NULL, FOREIGN KEY REFERENCES Project.ProjectId
UserId	INT	NOT NULL, FOREIGN KEY REFERENCES User.UserId
Accepted	BOOLEAN	

Tabelle 4: Relation Invited_to_work_on mit Attributen, Datentypen und Integritätsbedingungen

SharedToView		
Attribut	**Datentyp**	**Integritätsbedingungen**
ProjectId	INT	NOT NULL, FOREIGN KEY REFERENCES Project.ProjectId
AccessToken	VARCHAR(255)	NOT NULL
ExpirationDate	TIMESTAMP	NOT NULL

Tabelle 5: Relation SharedToView mit Attributen, Datentypen und Integritätsbedingungen

Beispieldaten

Für das angegebene Relationenschema wurden folgende Beispieldaten erstellt:

User					
UserId	Firstname	Lastname	EmailAddress	PasswordHash	BreakReminder
1	Lukas	Panni	lukas@lukaspanni.de	$2y$10$Uhg	120
2	Test	User	testuser@test.de	$2y$10$Uh	60
3	Kurt	Frank	kurt.frank@web.de	$2y$10$UhG	180

Tabelle 6: Beispieldaten für die Relation User

Project			
ProjectId	ProjectDescription	ProjectName	ProjectManager
1	NULL	Softwareengineering	1
2	In diesem Projekt geht es darum …	OPC/UA-Server	3

Tabelle 7: Beispieldaten für die Relation Project

Invited_to_work_on		
ProjectId	UserId	Accepted
1	2	True
2	3	False
2	1	NULL

Tabelle 8: Beispieldaten für die Relation Invited_to_work_on

WorkingTime			
UserId	ProjectId	StartTime	EndTime
1	1	01.01.2020 16:30	01.01.2020 17:00
3	2	30.04.2020 08:00	30.04.2020 16:00
2	1	29.02.2020 23:33	01.03.2020 05:55

Tabelle 9: Beispieldaten für die Relation WorkingTime

SharedToView		
ProjectId	ExpirationDate	AccessToken
1	12.06.2020 15:00	2y10UhgGSz2tY5xVwgckCGRk.rmkMZtzzSqH1G

Tabelle 10: Beispieldaten für die Relation SharedToView

3 Statische Sicht – Objekttypen zur Laufzeit

Im Folgenden Abschnitt werden die Objekttypen zur Laufzeit dargestellt und genauer erklärt.

3.1 UML Klassendiagramm

Die folgende Abbildung zeigt ein UML Klassendiagramm der Objekttypen zur Laufzeit. Für die Darstellung wurde ein UML-Klassendiagramm gewählt, da solche Diagramme die am weitesten verbreitete Darstellung für einen solchen Zweck sind.

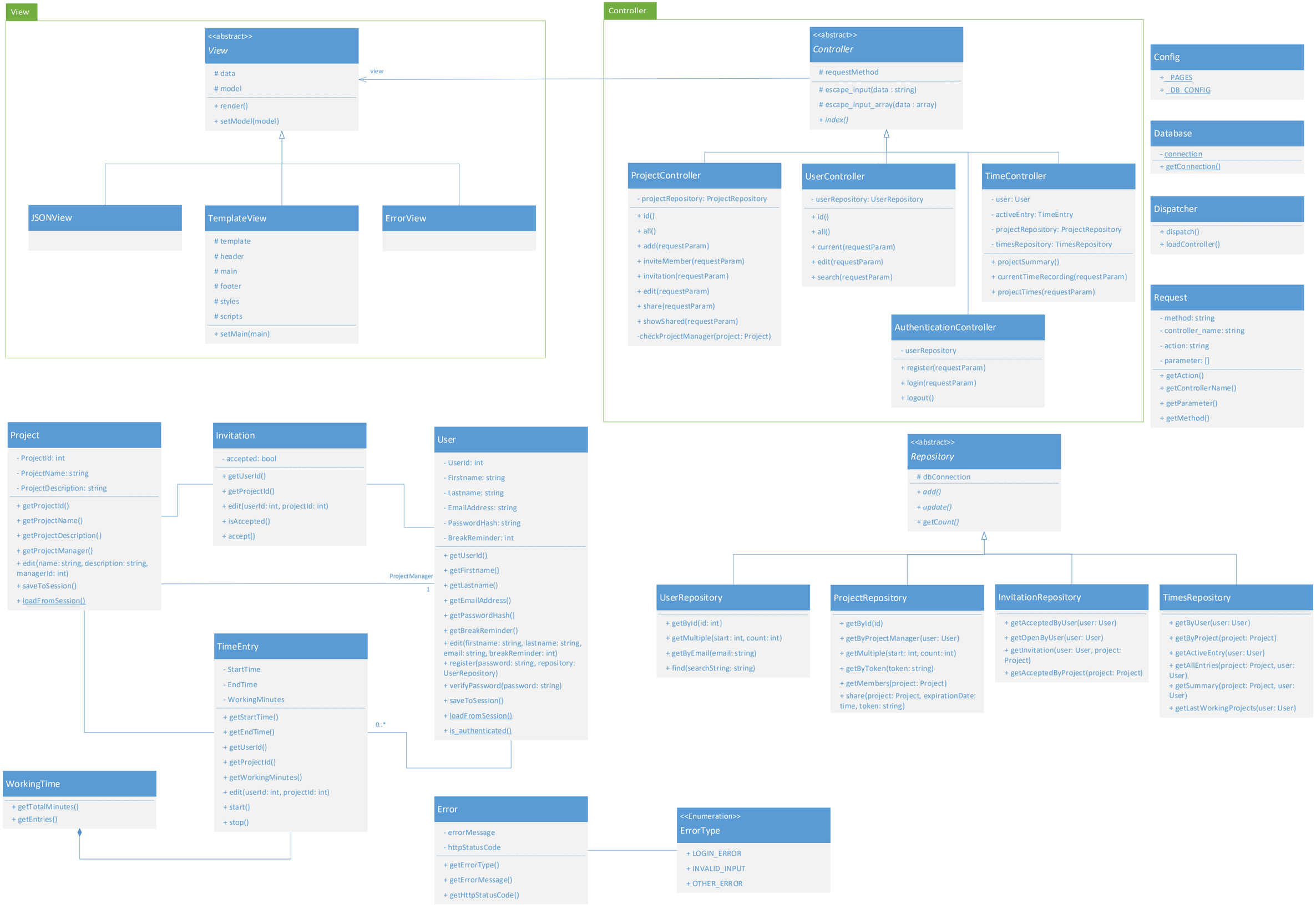

Abbildung 2: UML Klassendiagramm

Für das System wurde eine Model-View-Controller Architektur gewählt, weil damit eine einfache aber dennoch effektive Trennung von Datenhaltung, Anzeige und Steuerung möglich wird. Das Muster besteht im Wesentlichen aus den drei Rollen Model, View und Controller. Dadurch ergibt sich die folgende Trennung der Zuständigkeiten: Eine View sind nur für die Anzeige der Daten zuständig, ein Controller nimmt mögliche Benutzereingaben entgegen und übernimmt somit die Steuerung und das Model repräsentiert die Datenhaltung des Systems. [1]

Für jede Unterseite der geplanten Webanwendung wurde ein Controller entworfen, der alle HTTP-Requests, die sich auf diese Unterseite beziehen behandeln muss. Über Repositories können die Controller Objekte der Datenklassen laden und an eine View weitergeben. Für verschiedene Zwecke gibt es mehrere Implementierungen der View-Klasse. Die *JSONView* wird verwendet um Daten im JSON-Format auszugeben, die *ErrorView* um Fehlermeldungen anzuzeigen und die *TemplateView* wird genutzt, um Ausgaben basierend auf einem Template auszugeben.

Die Klassen *Dispatcher* und *Request* werden verwendet, um HTTP-Requests zum richtigen Controller zuzuordnen und die passende Methode dieses Controllers mit den angegebenen Parametern aufzurufen.

Der Zugriff auf persistierte Daten erfolgt über Repositories, um die eigentliche Logik besser von der Datenhaltung trennen zu können. Da der Zugriff auf Objekte, die in der Datenbank gespeichert sind so nur über die Repositories erfolgt trägt dies ebenfalls zu einer stärkeren Trennung der Zuständigkeiten bei. Außerdem wird so auch eine vergleichsweise einfache Möglichkeit geschaffen, zu einem späteren Zeitpunkt die Datenspeicherung anzupassen. [2] [3]

Objekte der Datenklassen *User*, *Project*, *Invitation*, *TimeEntry*, *WorkingTime* und *Error* können verwendet werden, um Aktionen, wie zum Beispiel das Starten einer Zeiterfassung lokal auszuführen. Solche Statusänderungen können dann über ein Repository-Objekt in der Datenbank persistiert werden.

Der gesamte Entwurf ist darauf ausgerichtet, die Zuständigkeiten einzelner Objekte sauber zu trennen und Komponenten leicht austauschbar zu machen. So sollen nachträgliche Änderungen, beispielsweise aufgrund einer Weiterentwicklung oder

aufgrund einer Anpassung an neu aufgekommen Anforderungen, einfach möglich sein.

4 Dynamische Sicht

4.1 Funktionseinheiten

Die Anforderungen des Anforderungsdokuments lassen sich zu folgenden Funktionseinheiten zusammenfassen:

ID	Funktionseinheiten	Anforderung	
F1	Zeitverwaltung	Zeiterfassung starten	R6
		Zeiterfassung unterbrechen	R7
F2	Zeitübersicht	Zeitübersicht anzeigen	R12
		Zeitübersicht filtern	R13
		Zeitübersicht Filter zurücksetzen	R14
		Zeitraum für Zeitübersicht	R15
		Zeitübersicht exportieren	R16
		Zeitübersicht für Project anzeigen	R17
		Projektbezogene Zeitübersicht als Diagramm	R19
		Diagramm anpassen	R20
F3	Zeitübersichtfreigabe	Zeitübersicht für Auftraggeber freigeben	R18
		Freigabe einschränken	R24
		Abrufen einer Freigabe	R25
F4	Projektverwaltung	Projekt erstellen	R1
		Projektdaten eintragen	R28
F5	Einladungsverwaltung	Zu Projekt einladen	R3
		Projekteinladung annehmen	R4
		Projekteinladung ablehnen	R5
		Projekt beitreten	R29
		Projekteinladung löschen	R30
F6	Benutzerverwaltung	Authentifizierung von Nutzer	R2
		Identifikation über E-Mail-Adresse	R26
		Hinterlegen einer E-Mail-Adresse	R27
F7	Pausenerinnerung	Grenzwert für Pausenerinnerung einstellen	R21
		Pausenerinnerung zustellen	R22

Tabelle 11: Funktionseinheiten

4.2 Prozessbeschreibung für Funktionseinheit Einladungsverwaltung

Die Funktionseinheit Einladungsverwaltung besteht aus dem Erstellen einer Einladung für ein Projekt, der Annahme beziehungsweise der Ablehnung und dem Löschen einer Einladung. Dabei wurde die Entscheidung getroffen, Einladungen nicht tatsächlich aus der Datenbank zu löschen, sondern diese als gelöscht interpretieren, sobald das Accepted-Feld auf einen Wert gesetzt wird. So wird die Zuordnung von Nutzern zu Projekten vereinfacht, da keine zusätzlichen Datenbankeinträge außer der Einladung benötigt werden und keine vermeidbaren Kopier- und Löschvorgänge in der Datenbank durchgeführt werden müssen.

Die Funktionseinheit wurde für die Darstellung in zwei Prozesse getrennt. Der erste dieser Prozesse ist der Prozess zum Erstellen einer Projekteinladung und ist als UML-Aktivitätsdiagramm in Abbildung 3 dargestellt. Bei diesem Vorgang wird zuerst geprüft ob der aktuell angemeldete Nutzer die Berechtigung hat, eine Einladung zu erstellen, und ob Eingabeparameter sinnrichtig gesetzt wurden. Werden alle Prüfungen ohne Fehler bestanden wird eine Einladung erstellt und in der Datenbank abgelegt.

Als zweiter Prozess wurde das Annehmen beziehungsweise das Ablehnen einer Einladung durch den Eingeladenen Nutzer modelliert Abbildung 4 zeigt diesen Prozess als UML-Aktivitätsdiagramm. Innerhalb dieses Prozesses werden alle Einladungen, die vom angemeldeten Nutzer noch nicht bearbeitet wurden aus der Datenbank geladen und können dann vom Nutzer angenommen oder abgelehnt werden. Das Ergebnis dieser Aktion wird schließlich in der Datenbank abgelegt.

Die beiden Teil-Prozesse werden als UML-Aktivitätsdiagramm dargestellt, da diese Art der Darstellung einfach, aber trotzdem effektiv ist. Insbesondere sind UML-Aktivitätsdiagramm subjektiv leichter verständlich als zum Beispiel BPMN-Diagramme, die eine mögliche Alternative Darstellung ermöglichen.

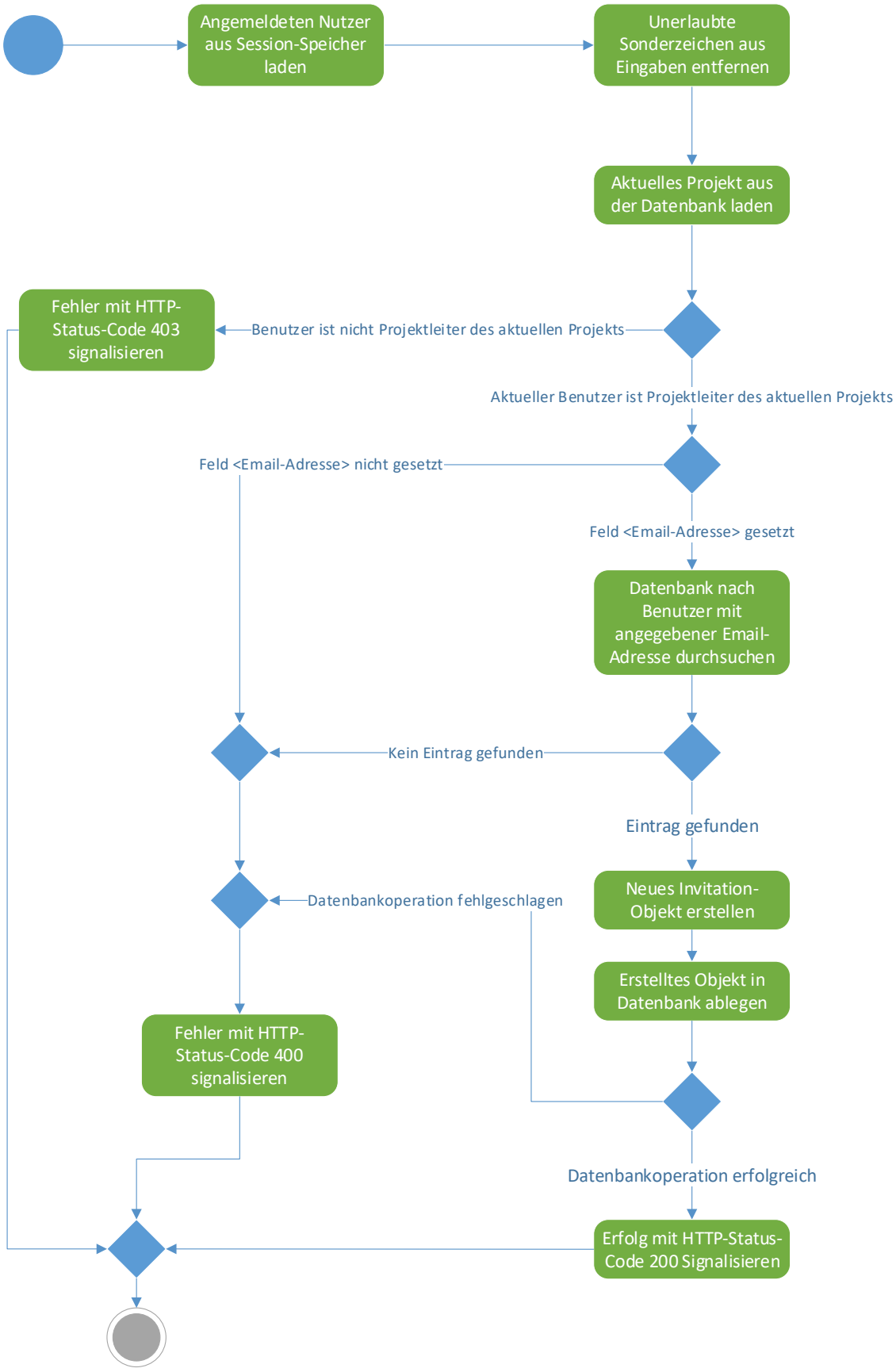

Abbildung 3: Prozess: Erstellen einer Einladung

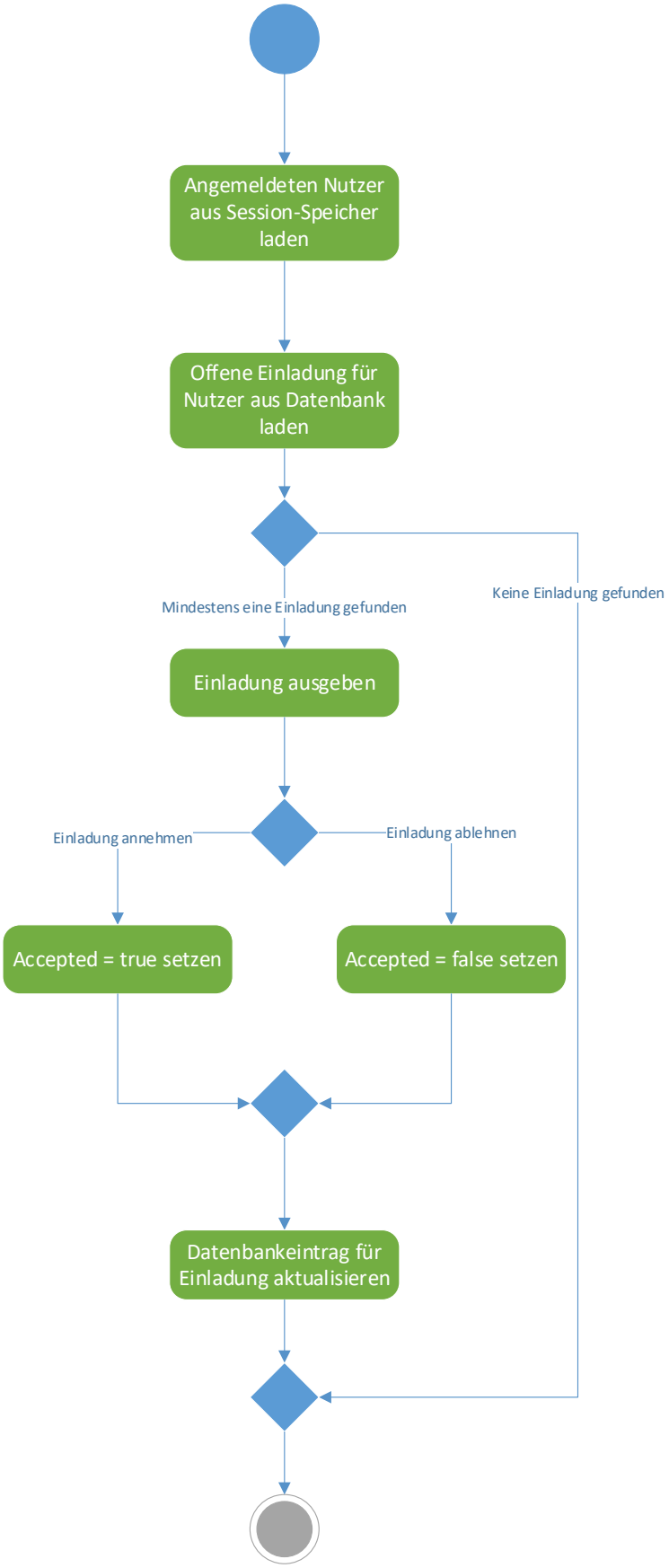

Abbildung 4: Prozess: Einladung annehmen/ablehnen

5 Auswahl der Backendtechnologie

In diesem Abschnitt soll die Entscheidung für die zu verwendende Backendtechnologie für die Implementierung des Systems getroffen werden. Das Backend der Webanwendung soll sich dabei um Datenhaltung und Geschäftslogik der Anwendung kümmern. In Kapitel 2 wurde bereits die Entscheidung getroffen ein relationales Datenmodell und damit auch eine Relationale Datenbank zu verwenden. Also ist im Folgenden noch die Entscheidung zu treffen, mit welcher Technologie die Geschäftslogik zu realisieren ist.

5.1 Kriterien für die Auswahl

K1	**Plattformunabhängigkeit**	**Bewertung**	**K.O.**
Die gewählte Backend-Technologie soll von konkreter Hardware und Betriebssystem unabhängig sein.		Skala: ja/nein	ja

K2	**Datenbankanbindung**	**Bewertung**	**K.O.**
Die gewählte Backend-Technologie soll die Anbindung einer relationalen Datenbank ermöglichen.		Skala: ja/nein	ja

K3	**Objektorientierung**	**Bewertung**	**K.O.**
Die gewählte Backend-Technologie soll objektorientierte Programmierung ermöglichen.		Skala: ja/nein	ja

K4	**Performance**	**Bewertung**	**K.O.**
Die gewählte Backend-Technologie soll möglichst performant sein.		Rang basierend auf: [4][1]	nein

K5	**Einarbeitungsaufwand**	**Bewertung**	**K.O.**
Die gewählte Backend-Technologie soll möglichst geringen Einarbeitungsaufwand erfordern.		Subjektive Skala: 0(kein)-10(viel)	nein

[1] Bei diesem Vergleich wird PHP in Verbindung mit den Frameworks Lumen und Laravel eingesetzt weshalb für die Bildung des Rangs der Mittelwert der beiden Framework-Ergebnisse genutzt wird.

K6	Einschränkung des Lösungsraums	Bewertung	K.O.
Der Lösungsraum soll durch die Wahl einer Alternative möglichst wenig eingeschränkt werden		Subjektive Skala: 0(keine)-10(starke)	nein

K7	Popularität	Bewertung	K.O.
Die gewählte Backendtechnologie soll möglichst populär sein, damit auf bestehende Ressourcen besser zurückgegriffen werden kann.		Rang basierend auf: [5]	nein

Die K.O.-Kriterien **K1**, **K2** und **K3** ergeben sich aus den Anforderungen und den bereits getroffenen Entwurfsentscheidungen. Das K.O.-Kriterium **K1** wird eingeführt, um zu gewährleisten, dass die Einführung des Systems unabhängig von konkret verfügbarer Hardware und vom eingesetzten Betriebssystem ist. Da nicht bekannt ist, welche Hardware und welches Betriebssystem von potentiellen Nutzern des Systems verwendet wird, wird so sichergestellt, dass möglichst viele potentielle Nutzer das System einsetzen können. Um das in 2 vorgestellte relationale Datenmodell einsetzen zu können, muss das K.O.-Kriterium **K2** eingeführt werden. Eine Alternative, die dieses Kriterium nicht erfüllt würde erfordern, ein neues Datenmodell zu erstellen, was zu diesem Zeitpunkt nicht gewünscht ist. Aufgrund der in 3.1 vorgestellten Objektorientierten-Architektur ist es notwendig, eine Technologie einzusetzen, die dieses Paradigma unterstützt. Ansonsten wäre auch hier eine komplette Überarbeitung der Architektur notwendig. Aus diesem Grund wird das Kriterium **K3** ebenfalls als K.O.-Kriterium festgelegt.

5.2 Gewichtung der Kriterien

Die K.O. Kriterien **K1**, **K2** und **K3** müssen von der ausgewählten Technologie zwingend erfüllt werden. Für die Kriterien **K4** - **K7** wurde die folgende Gewichtung festgelegt.

Kriterium	Gewichtung
K4	0,2
K5	0,3
K6	0,2
K7	0,3
Summe	1

Tabelle 12: Gewichtung der Kriterien

Die Gewichtung für **K7** und **K5** wurde höher festgelegt als die Gewichtung für die anderen Kriterien, da diese Kriterien maßgeblich den zusätzlichen Aufwand in der Implementierungsphase bestimmt und dieser möglichst gering gehalten werden sollte. Die Gewichtung für **K4** fällt relativ niedrig aus, da die Anforderungen an die Performance des Systems vergleichsweise niedrig ausfallen. Die Einschränkung des Lösungsraums wird ebenfalls niedrigerer gewichtet, da auf eine solche Einschränkung in dieser frühen Phase angemessen reagiert werden kann, sodass die Auswirkungen nicht so stark ausfallen.

5.3 Lösungsalternativen

Es wurde beschlossen mindestens vier alternativen zur Bewertung heranzuziehen. Die Lösungsalternativen wurden basierend auf Popularität und Zukunftstrends im Bereich der Webentwicklung ausgewählt. Die gewählten Alternativen sind:

ID	Bezeichnung	Beschreibung
A1	PHP	PHP-Basierte Web-Anwendung
A2	C#-ASP.NET	C#-Basierte Web-Anwendung in Verbindung mit dem ASP.NET Framework
A3	NodeJS	JavaScript-Basierte Web-Anwendung in Verbindung mit der NodeJS Javascript Laufzeitumgebung
A4	Java	Java-Basierte Web-Anwendung mit Java Server Pages und Java Servlets
A5	Python-Django	Python-Basierte Web-Anwendung in Verbindung mit dem Django Framework

Tabelle 13: Ausgewählte Lösungsalternativen

5.4 Bewertung der Lösungsalternativen

Die Alternativen **A1** [6] [7] [8], **A3** [6] [9] [10], **A4** [6] [11] [12] und **A5** [6] [13] [14] erfüllen alle K.O.-Kriterien.

Die Alternative **A2** erfüllt das K.O.-Kriterium **K1** nicht ohne weiteres [15] und wird deshalb für die folgende Bewertung der Nicht-K.O.-Kriterien nicht weiter in Betracht gezogen.

	K1	K2	K3	K4	K5	K6	K7	Bewertung
A1	Ja	Ja	Ja	4	2	1	1	1,9
A3	Ja	Ja	Ja	1	4	1	4	2,8
A4	Ja	Ja	Ja	2	4	0	2	2,2
A5	Ja	Ja	Ja	3	4	3	3	3,3
Gewichtung				0,2	0,3	0,2	0,3	1

Tabelle 14: Bewertung der Lösungsalternativen

Für **K4** wird der Rang basierend auf Testergebnissen von [4] gebildet. Dazu wird für jede Technologie der Mittelwert aller Testergebnisse gebildet. Da für Lösungsalternative **A1** bei diesem Vergleich zwei verschiedene Frameworks verglichen werden, ergibt sich der zur Rangbildung verwendete Wert aus dem Mittelwert der beiden Framework-Ergebnisse. Die Ergebnisse für **K5** basieren auf einer subjektiven Einschätzung. Für **K6** wurde die mögliche Einschränkung des Lösungsraums betrachtet. Die Alternativen **A1** und **A3** werden hier gleich bewertet, da beide die Einschränkung einer nicht vorhandenen Typsicherheit mit sich bringen. Die Alternative **A5** erhält die schlechteste Bewertung, da zusätzlich zu einer nicht vorhandenen Typsicherheit auch ein genaues Framework vorgegeben ist, was den Lösungsraum noch weiter einschränkt. Für die Lösungsalternative **A4** können keine derartigen Einschränkungen identifiziert werden. Die Ränge für das Kriterium **K7** ergeben sich nach [5].

Nach der Bewertung in Tabelle 14 fällt die Entscheidung auf Lösungsalternative **A1**. Da diese Alternative allerdings die schlechteste Performance im Vergleich aufweist, bleibt ein Restrisiko, dass diese Performance nicht ausreicht, die Anforderungen zu erfüllen. Da in den Anforderungen keine genauen Forderungen nach Performance bei bestimmter Hardware gestellt wurden kann die Eintrittswahrscheinlichkeit dafür nicht genauer angegeben werden. Sollte dieser Fall während der Implementierungsphase eintreten, ist es nur mit sehr hohem Zeitaufwand möglich die

Backendtechnologie zu wechseln. Außerdem bringt diese Alternative die Einschränkung einer dynamischen Typisierung und damit einer nicht gegebenen Typsicherheit mit sich. Daraus folgt, dass in der Implementierungsphase genauer auf Kompatibilität von Datentypen geachtet werden muss. Sonstige Auswirkungen auf die Architektur sind nicht ersichtlich.

6 Fazit

In diesem Dokument wurde der Entwurf einer Architektur für ein Zeiterfassungssystem vorgestellt. Dabei wurde versucht, den Entwurf möglichst einfach zu gestalten und dennoch die Forderungen nach Erweiterbarkeit und Anpassbarkeit des Systems zu erfüllen.

In den Abschnitten 2 wurden die zu persistierenden Daten modelliert und die Entscheidung für ein Datenbanksystem getroffen. Im folgenden Abschnitt wurden die zur Objekttypen zur Laufzeit anhand eines UML-Klassendiagramms vorgestellt. Abschnitt 4 beschreibt exemplarisch den Prozess des Erstellens einer Projekteinladung sowie den zugehörigen Prozess der Verarbeitung einer solchen Einladung durch einen Benutzer. Im Abschnitt 5 wurde schließlich der Entscheidungsprozess für eine bestimmte Backendtechnologie dargestellt und die getroffene Entscheidung dokumentiert.

Nachdem in diesem Dokument eine Architektur entworfen und verschiedene Entwurfsentscheidungen getroffen wurden kann mit der Implementierung des Systems begonnen werden. Dabei sollen die vorgestellte Architektur und getroffene Entwurfsentscheidungen berücksichtigt werden.

7 Literaturverzeichnis

[1] M. Geirhos, Entwurfsmuster Das umfassende Handbuch, Bonn: Rheinwerk Verlag, 2018, pp. 519-541.

[2] M. Fowler, Patterns of Enterprise Application Architecture, New Jersey: Pearson Education, 2003.

[3] N. Eder, „Das Repository Pattern anhand eines Beispiels inkl. Tests - Norbert Eder," 2011. [Online]. Available: https://www.norberteder.com/das-repository-pattern-anhand-eines-beispiels-inkl-tests/. [Zugriff am 28 Mai 2020].

[4] M. Cracan, „Web REST API Benchmark on a Real Life Application - Mihai Cracan - Medium," 2 August 2017. [Online]. Available: https://medium.com/@mihaigeorge.c/web-rest-api-benchmark-on-a-real-life-application-ebb743a5d7a3. [Zugriff am 1 Juni 2020].

[5] W3Techs, „Market Position Report of Top 10 Server-side Programming Languages," [Online]. Available: https://w3techs.com/technologies/market/programming_language/10. [Zugriff am 1 Juni 2020].

[6] „List of object-oriented programming languages - Wikipedia," 13 April 2020. [Online]. Available: https://en.wikipedia.org/wiki/List_of_object-oriented_programming_languages. [Zugriff am 1 Juni 2020].

[7] O'Reilly & Associates, „Relational Databases and SQL (Programming PHP)," 2003. [Online]. Available: https://docstore.mik.ua/orelly/webprog/php/ch08_02.htm. [Zugriff am 1 Juni 2020].

[8] PHP Group, „PHP: Installation und Konfiguration - Manual," [Online]. Available: https://www.php.net/manual/de/install.php. [Zugriff am 1 Juni 2020].

[9] OpenJS Foundation, „Download | Node.js," [Online]. Available:

https://nodejs.org/de/download/. [Zugriff am 1 Juni 2020].

[10] „Data Access,“ [Online]. Available: https://www.tutorialsteacher.com/nodejs/data-access-in-nodejs. [Zugriff am 1 Juni 2020].

[11] O'Reilly & Associates, „Database Connectivity (Java Servlet Programming),“ 2001. [Online]. Available: https://docstore.mik.ua/orelly/java-ent/servlet/ch09_01.htm. [Zugriff am 1 Juni 2020].

[12] The Apache Software Foundation, „Apache Tomcat 9 (9.0.35) - Tomcat Setup,“ 5 Mai 2020. [Online]. Available: https://tomcat.apache.org/tomcat-9.0-doc/setup.html. [Zugriff am 1 Juni 2020].

[13] Django Software Foundation, „How to install Django | Django documentation | Django,“ [Online]. Available: https://docs.djangoproject.com/en/3.0/topics/install/. [Zugriff am 1 Juni 2020].

[14] M. Makai, „Django ORM - Full Stack Python,“ [Online]. Available: https://www.fullstackpython.com/django-orm.html. [Zugriff am 1 Juni 2020].

[15] „Choose between ASP.NET 4.x and ASP.NET Core | Microsoft Docs,“ 12 Februar 2020. [Online]. Available: https://docs.microsoft.com/de-de/aspnet/core/fundamentals/choose-aspnet-framework?view=aspnetcore-3.1. [Zugriff am 30 Mai 2020].