# Wendy’s Reitstall

## Produktdokumentation

Eine Zusammenfassung von allen Anwendungsfällen, die während der Implementation dieses Projekts berücksichtigt wurden. In diesem Dokument befindet sich auch eine Zusammenfassung von allen geleisteten Stunden.

# Wendy's Reitstall

## *Anwendungsfälle*

### Anwendungsfall #1

#### *Zusammenfassung*

- *Titel:* Pferde anzeigen
- *Scope:* Suche
- *Level:* User Goal
- *Aktoren:* Benutzer
- *Kurzbeschreibung:* Benutzer möchte alle nicht gelöschten Pferde in der Datenbank ansehen.

#### *Szenarien*

- *Vorbedingungen*: Pferde sind bereits in der Datenbank abgespeichert.
- *Hauptszenario*: Der Benutzer gibt keine Suchkriterien an und das System gibt alle nicht gelöschten Pferde zurück.
- *Fehlerszenario*: Es gibt keine Pferde in der Datenbank, die angezeigt werden können, der Benutzer soll dementsprechend benachrichtigt. Wenn keine Datenbankverbindung aufgebaut werden kann, wird der Benutzer benachrichtigt.
- *Alternativszenario*: Keins
- *Nachbedingungen*: Der Benutzer kann ein Pferd bearbeiten oder löschen.

#### *Nonfunctional Constraints/Special Requirements*

- Diese Suchfunktion wird wahrscheinlich öfters ausgeführt.

# Anwendungsfall #2

## *Zusammenfassung*

- *Titel:* Pferd erstellen
- *Scope:* Datensatz einfügen
- *Level:* User Goal
- *Aktoren:* Benutzer
- *Kurzbeschreibung:* Benutzer möchte ein neues Pferd in die Datenbank einfügen.

## *Szenarien*

- *Vorbedingungen*: Das Pferd existiert nicht bereits in der Datenbank.
- *Hauptszenario*: Der Benutzer gibt die notwendigen Daten für das Pferd an.
- *Fehlerszenario*: Das Pferd befindet sich bereits in der Datenbank, der Benutzer soll dementsprechend benachrichtigt. Wenn keine Datenbankverbindung aufgebaut werden kann, wird der Benutzer benachrichtigt.
- *Alternativszenario*: Keins
- *Nachbedingungen*: Der Benutzer kann ein Pferd bearbeiten oder löschen.

## *Nonfunctional Constraints/Special Requirements*

- Diese Fsunktion wird wahrscheinlich öfters ausgeführt.

# Anwendungsfall #3

## Zusammenfassung

*Titel*: Pferde nach Attributen filtern bzw. suchen

*Scope*: Suche

*Level*: User Goal

*Aktoren*: Benutzer

*Kurzbeschreibung*: Benutzer möchte die Liste der angezeigten Pferde nach bestimmten Attributen filtern bzw. suchen (Attribute: Name, Größe, Gewicht).

## Szenarien

- *Vorbedingungen*: Pferde sind bereits in der Datenbank abgespeichert.
- *Hauptszenario*: Der Benutzer gibt spezifische Suchkriterien in der Suchmaske ein und es werden alle Pferde gezeigt, die diese Kriterien erfüllen.
- *Fehlerszenario*: Wenn keine Datenbankverbindung aufgebaut werden kann, wird der Benutzer benachrichtigt. Wenn die Eingaben fehlerhaft sind, wird der Benutzer benachrichtigt.
- *Alternativszenario*: Keine Pferde erfüllen die eingegebenen Suchkriterien und der Benutzer wird benachrichtigt.
- *Nachbedingungen*: Der Benutzer kann ein Pferd bearbeiten oder löschen.

## Nonfunctional Constraints/Special Requirements

- Diese Suchfunktion wird wahrscheinlich öfters ausgeführt.
- Um fehlerhafte Eingaben zu vermeiden sollen alle Eingaben sobald wie möglich geprüft werden.

# Anwendungsfall #4

## Zusammenfassung

*Titel*: Pferd bearbeiten

*Scope*: Datensatz ändern

*Level*: User Goal

*Aktoren*: Benutzer

*Kurzbeschreibung*: Benutzer möchte Attribute (Größe, Gewicht, Bild) eines bestimmtes Pferdes ändern.

## Szenarien

- *Vorbedingungen*: Ein Pferd wurde über die Suchmaske gefunden.
- *Hauptszenario*: Der Benutzer möchte bestimmte Attribute (Größe, Gewicht, Bild) eines Pferdes bearbeiten.
- *Fehlerszenario*: Es wird keinen gültigen Pfad zum Bild angegeben und das Bild lässt sich nicht richtig anzeigen. Der Benutzer gibt ungültige Daten in das jeweilige Feld, dieser Fehler soll eine Benachrichtigung erzeugen. Eine Aktualisierung der Daten kann auch fehlschlagen und der Benutzer soll benachrichtigt werden, wenn dies passiert.
- *Alternativszenario*: Der Benutzer möchte die Änderungen nicht speichern, also muss es möglich sein, den Vorgang abzubrechen.
- *Nachbedingungen*: Die Änderungen werden in der Datenbank erfolgreich gespeichert.

## Nonfunctional Constraints/Special Requirements

- Felder, die nicht geändert werden können (Geburtsdatum, Name des Pferdes) sollen auch als solche ersichtlich sein.

# Anwendungsfall #5

## *Zusammenfassung*

*Titel*: Pferd löschen

*Scope*: Datensatz löschen

*Level*: User Goal

*Aktoren*: Benutzer

*Kurzbeschreibung*: Benutzer möchte ein bestimmtes Pferd löschen.

## *Szenarien*

- *Vorbedingungen*: Ein Pferd wurde über die Suchmaske gefunden.
- *Hauptszenario*: Der Benutzer möchte ein bestimmtes Pferd aus der Datenbank löschen.
- *Fehlerszenario*: Eine Aktualisierung der Datenbank kann fehlschlagen und der Benutzer soll benachrichtigt werden, wenn dies passiert.
- *Alternativszenario*: Der Benutzer möchte das Pferd doch nicht löschen, als muss es möglich sein, den Vorgang abzubrechen.
- *Nachbedingungen*: Das Pferd wird als gelöscht markiert.

## *Nonfunctional Constraints/Special Requirements*

- Es soll kein Datensatz tatsächlich aus der Datenbank gelöscht, es wird nur eine entsprechende Flagge gesetzt.

# Anwendungsfall #6

## *Zusammenfassung*

*Titel*: Termin reservieren

*Scope*: Buchen

*Level*: User Goal

*Aktoren*: Benutzer

*Kurzbeschreibung*: Benutzer möchte ein Pferd für einen Termin reservieren.

## *Szenarien*

- *Vorbedingungen*: Der Benutzer hat die Maske für eine neue Buchung aufgemacht, gültige Daten für die Reservierung (Datum, Startzeit, Endzeit) eingegeben und ein verfügbares Pferd ausgewählt.
- *Hauptszenario*: Der Benutzer möchte ein Pferd für einen bestimmten Termin reservieren. Es können gleichzeitig mehrere Termine angelegt werden, es soll daher mehrere Reihen geben, die dies ermöglichen.
- *Fehlerszenario*: Der Benutzer wählt ein ungültiges Datum, ungültige Uhrzeiten oder ein nicht verfügbares Pferd aus und soll genau benachrichtigt, welcher Fehler aufgetreten ist. Wenn die Reservierung(en) aufgrund eines Fehlers der Datenbank nicht abgespeichert werden kann(können), muss es auch eine entsprechende Benachrichtigung geben.
- *Alternativszenario*: Der Benutzer will doch keinen Termin anlegen und Formulare, die nicht vollständig ausgefüllt sind, sollen nicht ausgewertet werden.
- *Nachbedingungen*: Der Termin wird in der Datenbank abgespeichert und einer Buchung zugeordnet.

## *Nonfunctional Constraints/Special Requirements*

- Jede Eingabe soll der Reihe nach überprüft werden. Wenn das Datum stimmt, dann sollten die Uhrzeiten geprüft werden, usw. Dies garantiert, dass der richtige Fehler erkannt wird und der Benutzer dementsprechend benachrichtigt wird.

# Anwendungsfall #7

## Zusammenfassung

*Titel*: Buchungen anzeigen

*Scope*: Suche

*Level*: User Goal

*Aktoren*: Benutzer

*Kurzbeschreibung*: Benutzer möchte alle Buchungen in der Datenbank ansehen.

## Szenarien

- *Vorbedingungen*: Es gibt Buchungen in der Datenbank.
- *Hauptszenario*: Der Benutzer möchte alle Buchungen im System anschauen.
- *Fehlerszenario*: Es gibt keine Buchungen in der Datenbank, die angezeigt werden können, der Benutzer soll dementsprechend benachrichtigt. Wenn keine Datenbankverbindung aufgebaut werden kann, wird der Benutzer benachrichtigt.
- *Alternativszenario*: Keins
- *Nachbedingungen*: Der Benutzer kann eine Buchungsbestätigung ausstellen oder einzelne Termine einer Buchung anzeigen und diese ändern bzw. löschen.

## Nonfunctional Constraints/Special Requirements

- Diese Suchfunktion wird wahrscheinlich öfters ausgeführt.

# Anwendungsfall #8

## *Zusammenfassung*

*Titel*: Buchungen nach Kundennamen suchen

*Scope*: Suche

*Level*: User Goal

*Aktoren*: Benutzer

*Kurzbeschreibung*: Benutzer möchte nach allen Buchungen von einem Kunden suchen.

## *Szenarien*

- *Vorbedingungen*: Es gibt Buchungen in der Datenbank.
- *Hauptszenario*: Der Benutzer möchte alle Buchungen in der Datenbank anschauen, die von einem bestimmten Kunden sind.
- *Fehlerszenario*: Es gibt keine Buchungen in der Datenbank, die angezeigt werden können, der Benutzer soll dementsprechend benachrichtigt. Wenn keine Datenbankverbindung aufgebaut werden kann, wird der Benutzer benachrichtigt.
- *Alternativszenario*: Keins
- *Nachbedingungen*: Der Benutzer kann eine Buchungsbestätigung ausstellen oder einzelne Termine einer Buchung anzeigen und diese ändern bzw. löschen.

## *Nonfunctional Constraints/Special Requirements*

- Diese Suchfunktion wird wahrscheinlich öfters ausgeführt.

# Anwendungsfall #9

## Zusammenfassung

*Titel*: Terminen nach Datum suchen

*Scope*: Suche

*Level*: User Goal

*Aktoren*: Benutzer

*Kurzbeschreibung*: Benutzer möchte nach allen Terminen zwischen zwei Daten suchen.

## Szenarien

- *Vorbedingungen*: Es gibt Terminen in der Datenbank.
- *Hauptszenario*: Der Benutzer möchte alle Termine in der Datenbank anschauen, die zwischen zwei Daten stattfinden.
- *Fehlerszenario*: Es gibt keine Termine in der Datenbank, die angezeigt werden können, der Benutzer soll dementsprechend benachrichtigt. Wenn keine Datenbankverbindung aufgebaut werden kann, wird der Benutzer benachrichtigt.
- *Alternativszenario*: Keins
- *Nachbedingungen*: Der Benutzer kann einen Termin ändern bzw. löschen.

## Nonfunctional Constraints/Special Requirements

- Diese Suchfunktion wird wahrscheinlich öfters ausgeführt.

# Anwendungsfall #10

## *Zusammenfassung*

*Titel*: Termin bearbeiten

*Scope*: Datensatz ändern

*Level*: User Goal

*Aktoren*: Benutzer

*Kurzbeschreibung*: Benutzer möchte die Attribute (Datum, Anfangszeit, Endzeit) eines bestimmten Termins ändern.

## *Szenarien*

- *Vorbedingungen*: Eine gültige Buchung wurde über die Suchmaske gefunden.
- *Hauptszenario*: Der Benutzer möchte einen bestimmten Termin innerhalb einer Buchung ändern.
- *Fehlerszenario*: Wenn ein Termin nicht mehr geändert werden kann, dann soll der Benutzer benachrichtigt werden. Wenn gültige Eingaben in ein Feld eingegeben wird, muss der Benutzer auf den richtigen Fehler benachrichtigt werden. Wenn die Änderungen aufgrund eines Fehlers der Datenbank nicht abgespeichert werden können, muss es auch eine entsprechende Benachrichtigung geben.
- *Alternativszenario*: Der Benutzer möchte einen Termin doch nicht ändern, es soll also möglich sein, den Vorgang abzubrechen.
- *Nachbedingungen*: Die Änderungen werden in der Datenbank abgespeichert.

## *Nonfunctional Constraints/Special Requirements*

- Nur Termine, die mehr als zwei Wochen in der Zukunft liegen, können geändert werden. Bevor eine einzelne Buchung samt allen Terminen angezeigt wird, muss das System auf diese Bedingung überprüfen, welche Termine nicht geändert werden können und diese dementsprechend markieren.

# Anwendungsfall #11

## Zusammenfassung

*Titel*: Termin löschen/stornieren

*Scope*: Datensatz löschen

*Level*: User Goal

*Aktoren*: Benutzer

*Kurzbeschreibung*: Benutzer möchte einen bestimmten Termin löschen.

## Szenarien

- *Vorbedingungen*: Eine gültige Buchung wurde über die Suchmaske gefunden.
- *Hauptszenario*: Der Benutzer möchte einen Termin innerhalb einer Buchung löschen/stornieren.
- *Fehlerszenario*: Wenn ein Termin nicht mehr gelöscht/storniert werden kann, dann soll der Benutzer benachrichtigt werden. Wenn die Änderungen aufgrund eines Fehlers der Datenbank nicht abgespeichert werden können, muss es auch eine entsprechende Benachrichtigung geben.
- *Alternativszenario*: Der Benutzer möchte einen Termin doch nicht löschen, es soll also möglich sein, den Vorgang abzubrechen.
- *Nachbedingungen*: Der Termin wird in der Datenbank als gelöscht/storniert markiert.

## Nonfunctional Constraints/Special Requirements

- Nur Termine, die mehr als zwei Wochen in der Zukunft liegen, können geändert werden. Bevor eine einzelne Buchung samt allen Terminen angezeigt wird, muss das System auf diese Bedingung überprüfen, welche Termine nicht geändert werden können und diese dementsprechend markieren.

## Anwendungsfall #12

### Zusammenfassung

*Titel*: Termine einer Buchung anzeigen

*Scope*: Suche

*Level*: User Goal

*Aktoren*: Benutzer

*Kurzbeschreibung*: Benutzer möchte alle Termine einer Buchung ansehen.

### Szenarien

- *Vorbedingungen*: Eine Buchung wurde ausgewählt.
- *Hauptszenario*: Der Benutzer möchte alle Termine einer einzelnen Buchung anschauen und eventuell welche löschen bzw. ändern oder eine Buchungsbestätigung ausstellen. Beim Laden der Liste der Termine soll die Termine markiert, die nicht mehr änderbar bzw. stornierbar sind.
- *Fehlerszenario*: Wenn keine Datenbankverbindung aufgebaut werden kann, wird der Benutzer benachrichtigt.
- *Alternativszenario*: Keins
- *Nachbedingungen*: Keine

### Nonfunctional Constraints/Special Requirements

- Diese Funktion wird wahrscheinlich öfters ausgeführt.

## Anwendungsfall #13

### Zusammenfassung

*Titel*: Buchungsbestätigung ausstellen

*Scope*: Suche

*Level*: User Goal

*Aktoren*: Benutzer

*Kurzbeschreibung*: Benutzer möchte eine Buchungsbestätigung ausstellen.

### Szenarien

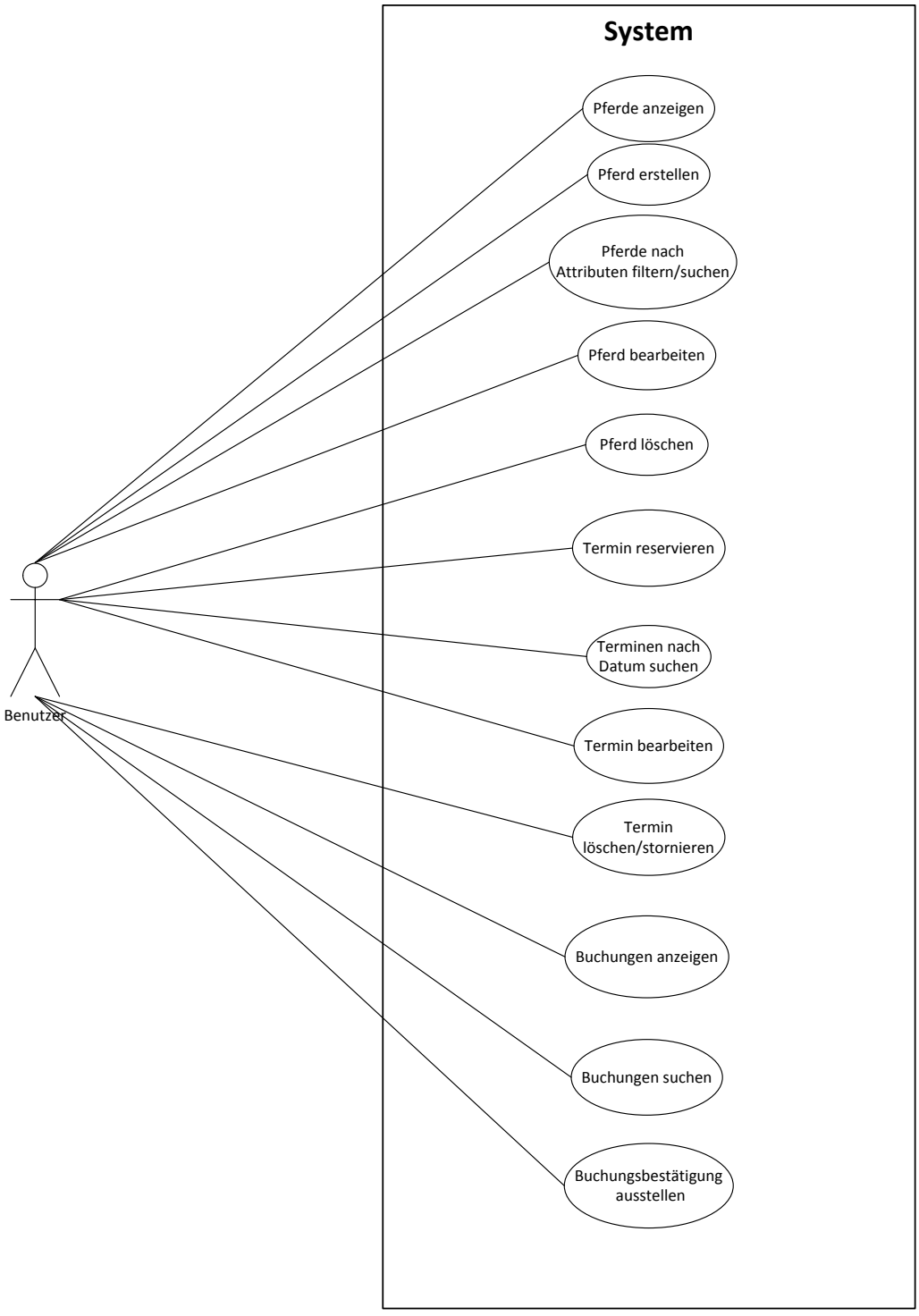
- *Vorbedingungen*: Eine Buchung wurde ausgewählt.
- *Hauptszenario*: Der Benutzer möchte eine Buchungsbestätigung aller Termine einer Buchung ausstellen.
- *Fehlerszenario*: Wenn keine Datenbankverbindung aufgebaut werden kann, wird der Benutzer benachrichtigt.
- *Alternativszenario*: Keins
- *Nachbedingungen*: Die Buchungsbestätigung wurde ausgestellt.

### Nonfunctional Constraints/Special Requirements

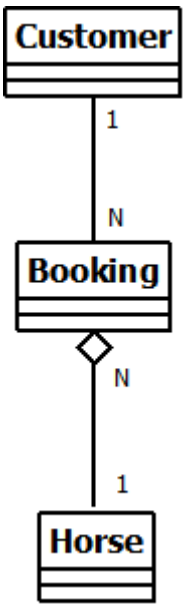
- Diese Funktion wird wahrscheinlich öfters ausgeführt.

# *Diagramme*

## Anwendungsfall Diagramm



## Domänenmodell



# *Stundenlisten*

## Zusammenfassung aller geleisteten Stunden

| Description | Total Hours |
|---|---|
| **DAO Implementation** | 0:46 |
| **Create classes/interfaces** | 1:38 |
| **DAO Implementation** | 1:45 |
| **Documentation** | 4:58 |
| **InfLab** | 0:55 |
| **Junit Tests** | 10:56 |
| **Miscelleaneous** | 1:30 |
| **Redo data structure** | 2:53 |
| **Refactoring** | 5:27 |
| **Service Implementation** | 4:02 |
| **Sketch models** | 2:42 |
| **SQL** | 0:42 |
| **Team building meeting** | 1:15 |
| **Technical preparation** | 3:53 |
| **UML** | 1:06 |
| **Use-cases** | 2:43 |
| **User interface design** | 0:56 |
| **User interface implementation** | 21:38 |
| | |
| **Total** | 69:45 |

## Stunden in Detail

| Date | Begin | End | Total Hours | Description |
|---|---|---|---|---|
| **07 Oct 14** | 08:45 | 09:45 | 01:00 | Technical preparation |
| **07 Oct 14** | 10:20 | 10:30 | 00:10 | Technical preparation |
| **07 Oct 14** | 12:40 | 13:40 | 01:00 | Technical preparation |
| **07 Oct 14** | 14:00 | 14:42 | 00:42 | SQL |
| **08 Oct 14** | 15:30 | 16:30 | 01:00 | Technical preparation |
| **08 Oct 14** | 18:15 | 19:30 | 01:15 | Team building meeting |
| **09 Oct 14** | 17:00 | 17:25 | 00:25 | Sketch models |
| **10 Oct 14** | 10:54 | 11:35 | 00:41 | Sketch models |
| **10 Oct 14** | 15:05 | 15:22 | 00:17 | Sketch models |
| **10 Oct 14** | 15:58 | 16:47 | 00:49 | Sketch models |
| **10 Oct 14** | 16:47 | 17:35 | 00:48 | Create classes/interfaces |
| **11 Oct 14** | 10:15 | 11:05 | 00:50 | Create classes/interfaces |
| **11 Oct 14** | 11:05 | 11:35 | 00:30 | Junit Tests |
| **11 Oct 14** | 11:39 | 12:56 | 01:17 | Junit Tests |
| **11 Oct 14** | 14:25 | 15:17 | 00:52 | DAO Implementation |
| **11 Oct 14** | 15:57 | 18:50 | 02:53 | Redo data structure |
| **12 Oct 14** | 11:00 | 11:30 | 00:30 | Sketch models |
| **12 Oct 14** | 12:40 | 14:10 | 01:30 | Junit Tests |
| **12 Oct 14** | 14:50 | 16:17 | 01:27 | Junit Tests |
| **12 Oct 14** | 20:00 | 20:41 | 00:41 | Junit Tests |
| **13 Oct 14** | 11:00 | 12:46 | 01:46 | Junit Tests |
| **13 Oct 14** | 14:22 | 15:08 | 00:46 | DAO Implementation |
| **13 Oct 14** | 15:08 | 15:23 | 00:15 | Documentation |
| **14 Oct 14** | 09:45 | 10:50 | 01:05 | Documentation |
| **14 Oct 14** | 13:05 | 14:00 | 00:55 | InfLab |
| **14 Oct 14** | 15:35 | 20:13 | 04:38 | Refactoring |
| **15 Oct 14** | 18:00 | 18:30 | 00:30 | UML |
| **16 Oct 14** | 13:48 | 13:57 | 00:09 | Refactoring |
| **16 Oct 14** | 14:47 | 17:30 | 02:43 | Use-cases |
| **16 Oct 14** | 19:32 | 20:12 | 00:40 | Refactoring |
| **17 Oct 14** | 10:00 | 10:53 | 00:53 | DAO Implementation |
| **17 Oct 14** | 10:53 | 11:20 | 00:27 | Service Implementation |
| **17 Oct 14** | 14:33 | 16:25 | 01:52 | Junit tests |
| **17 Oct 14** | 17:00 | 18:32 | 01:32 | Junit tests |
| **18 Oct 14** | 09:58 | 10:19 | 00:21 | Junit tests |
| **18 Oct 14** | 10:47 | 12:33 | 01:46 | Service Implementation |
| **18 Oct 14** | 13:00 | 14:08 | 01:08 | Service Implementation |
| **18 Oct 14** | 14:31 | 15:12 | 00:41 | Service Implementation |
| **18 Oct 14** | 15:49 | 16:25 | 00:36 | UML |
| **21 Oct 14** | 11:00 | 12:19 | 01:19 | User interface implementation |
| **21 Oct 14** | 12:19 | 13:20 | 01:01 | Documentation |

| | | | | |
|---|---|---|---|---|
| **21 Oct 14** | 13:56 | 14:36 | 00:40 | Documentation |
| **21 Oct 14** | 15:22 | 16:05 | 00:43 | Technical preparation |
| **21 Oct 14** | 16:52 | 17:48 | 00:56 | User interface design |
| **21 Oct 14** | 18:55 | 19:55 | 01:00 | User interface implementation |
| **21 Oct 14** | 20:25 | 21:10 | 00:45 | User interface implementation |
| **22 Oct 14** | 09:35 | 12:23 | 02:48 | User interface implementation |
| **22 Oct 14** | 14:55 | 17:24 | 02:29 | User interface implementation |
| **22 Oct 14** | 17:50 | 21:24 | 03:34 | User interface implementation |
| **22 Oct 14** | 21:45 | 22:04 | 00:19 | User interface implementation |
| **23 Oct 14** | 16:00 | 18:01 | 02:01 | User interface implementation |
| **23 Oct 14** | 19:00 | 21:52 | 02:52 | User interface implementation |
| **24 Oct 14** | 09:15 | 11:09 | 01:54 | User interface implementation |
| **25 Oct 14** | 12:23 | 13:53 | 01:30 | Miscelleaneous |
| **26 Oct 14** | 10:20 | 12:00 | 01:40 | Documentation |
| **27 Oct 14** | 09:30 | 10:54 | 01:24 | User interface implementation |
| **27 Oct 14** | 19:15 | 20:28 | 01:13 | User interface implementation |
| **27 Oct 14** | 20:28 | 20:45 | 00:17 | Documentation |
| | | | | |
| **Sum** | | | 69:45 | |

# Probleme

## Nicht fertig implementierte Funktionalität

- Suchfelder um nach Pferden zu filtern funktioniert nicht perfekt
- Anzeige vom Bild eines Pferdes
- Funktion, um das bequeme Ändern des Bildes eines Pferdes zu ermöglichen (momentan müsste man einen Pfad angeben)

## Nicht implementierte Funktionalität

- Funktionen, um nach einzelnen Buchungen bzw. Terminen zu filtern