HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Bachelorarbeit

Webbasiertes Ressourcen- und Terminplanungs-Tool

Moderne, responsive Webapplikation für Fahrschulen

Hochschule für Technik Rapperswil
Herbstsemester 2013

Autoren Moreno Feltscher
Peter Manser

Betreuer Prof. Dr. Andreas Rinkel
Experte Dr. Andreas Jarosch
Gegenleser Prof. Hansjörg Huser

Aufgabenstellung

Das Ziel dieser Arbeit besteht darin eine Webanwendung zu entwickeln, welche die Fahrlehrer in der Terminfindung für Fahrlektionen unterstützt. Zurzeit ist diese Aufgabe eine zeitraubende Angelegenheit, da für eine bestimmte Fahrzeugkategorie (z.B. Lastwagen) mehrere Fahrzeuge und Fahrlehrer potentiell zur Auswahl stehen und somit zur Findung eines passenden Termins die Agenden aller Komponenten abgeglichen werden müssen. Zudem gilt es je nach Ausbildung weitere Abhängigkeiten zu berücksichtigen, so muss z.B. bei der Anhängerausbildung neben dem eigentlichen Fahrzeug auch der dazu passende Anhänger zur Verfügung stehen. Dieser bisher manuell erfolgende Abgleich soll in Zukunft von der Webanwendung gemacht werden.

Da der Termin der nächsten Fahrlektion meistens noch im Fahrzeug mit dem Fahrschüler vereinbart wird, soll die Webanwendung für den Zugriff mit mobilen Geräten (vor allem iPhone und iPad) optimiert sein. Neben dem eigentlichen Finden/Verwalten von Terminen an welchen alle zu berücksichtigenden Kriterien erfüllt sind, soll die Anwendung auch die Verwaltung der zur Verfügung stehenden Ressourcen (Fahrzeuge, Fahrlehrer) inkl. deren Abhängigkeiten ermöglichen. Letztere Funktionalität muss dabei nicht zwingend für mobile Geräte optimiert sein. Der Zugriff auf die Anwendung soll rollenbasiert abgesichert sein. Zu diesem Zweck soll auch eine Benutzerverwaltung zur Verfügung gestellt werden. Des weiteren sollen die für eine bestimmten Ressource erstellten Termine über eine iCal Schnittstelle nach Aussen zugänglich gemacht werden, so dass die Termine z.B. auf dem iPhone als Kalender eingebunden werden können. Um nach Abschluss einer Ausbildung die von einem Fahrschüler benötigten Fahrlektionen ermitteln zu können, soll die Anwendung über eine minimale Fahrschülerdatenbank verfügen, so dass jeder Termin eindeutig einem Fahrschüler zugeordnet werden kann. Vor der eigentlichen Umsetzung der Anwendung gilt es ein für die Zielgeräte optimiertes Design sowie geeigneten Interaktionsablauf zu erarbeiten. Abhängig von der für die Arbeit zur Verfügung stehenden Zeit, kann die Anwendung um weitere Funktionen ergänzt werden. So z.B. um eine Statistikfunktion, die es ermöglicht aufgrund der im System festgehaltenen Terminen Auswertungen über die Anzahl unterrichteter Fahrlektionen zu erhalten. Oder die Funktion, Rechnungen zu generieren basierend auf der Anzahl unterrichteter Fahrlektionen.

Als Resultat wird eine Webanwendung erwartet, deren Handhabung sowie Look & Feel auf mobilen Geräten (insbesondere iPhone/iPad) möglichst analog zu den nativen Anwendungen ist. Qualitativ besteht der Anspruch, dass neben der korrekten Umset-

zung der „Allgemeinfälle“ auch mögliche Fehleingaben, oder unzulässige Operationen erkannt und korrekt behandelt werden. Dazu gehört beispielsweise der Fall, wenn zwei Fahrlehrer gleichzeitig versuchen einen Termin mit demselben Fahrzeug zu erstellen. Oder ein Termin bearbeitet werden soll, der in der Zwischenzeit gelöscht wurde. Nicht triviale technische Aspekte in der umgesetzten Lösung müssen für eine in der Informatik versierte Person verständlich dokumentiert sein.

Rapperswil, 19. Dezember 2013

Prof. Dr. Andreas Rinkel
Institut für Internet-Technologien und -Anwendungen
Hochschule für Technik Rapperswil

Erklärung der Eigenständigkeit

Wir erklären hiermit,

- dass wir die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt haben, ausser derjenigen, welche explizit in der Aufgabenstellung erwähnt ist oder mit dem Betreuer schriftlich vereinbart wurde,
- dass wir sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben haben.
- dass wir keine durch Copyright geschützten Materialien (z.B. Bilder) in dieser Arbeit in unerlaubter Weise genutzt haben.

Moreno Feltscher

Peter Manser

Abstract

Ausgangslage

Mittelgrosse Fahrschulen haben oftmals das Problem, eine effektive Ressourcen- und Terminplanung zu führen. Den Grund dafür stellen die verschiedenen involvierten Interessenten, wie Fahrlehrer und Fahrschüler, und deren Verfügbarkeiten dar. Zudem verfügen solche Fahrschulen über Fahrzeugparks mit mehreren Fahrzeugen und deren Zusätze, die wiederum Verfügbarkeiten und Kompatibilitäten aufweisen.

Vorgehen/Technologien

Zur Vereinfachung der angesprochenen Ressourcen- und Terminplanung wird eine rollenbasierte Webapplikation auf der Basis von Symfony2 entwickelt. Diese ermöglicht es der Fahrschule, die Ressourcen- und Fahrschülerdaten zentral zu pflegen. Für die persistente Datenhaltung kommt eine MySQL-Datenbank zum Einsatz. Die Vereinbarung einer neuen Fahrlektion geschieht in einer intuitiven Kalenderansicht, die alle vereinbarten Termine und Verfügbarkeiten von Ressourcen verständlich auflistet und mittels Filteroptionen entsprechend konfiguriert werden kann.

Da die Verwaltung der Applikation oftmals durch die Fahrlehrer vor Ort auf einem iPad erfolgt, wird auf eine responsive Darstellung, basierend auf dem Frontend-Framework Bootstrap und dem Toolkit Modernizr, gesetzt. Dies erlaubt es dem Benutzer, unabhängig von dem eingesetzten Gerät und Browser mit der Webapplikation interagieren zu können.

Durch das Testingframework PHPUnit wird mittels Unit- und Functional-Tests die Lauffähigkeit und Korrektheit des Codes sichergestellt.

Ergebnis

Entstanden ist eine Webapplikation, die den Verwaltungsaufwand der Fahrschulen durch effiziente Prozessabläufe bei der Termin- und Ressourcenplanung beschleunigt und vereinfacht. Ein Zugriff auf die Fahrschuldaten ist sowohl über die Weboberfläche, als auch über eine iCal- Schnittstelle zur Anbindung von Termindaten jederzeit möglich.

Management Summary

Ausgangslage

Die Verwaltung von Ressourcen im Bezug auf die Terminplanung gestaltet sich bei mittelgrossen Fahrschulen meist schwierig. Dies ist darauf zurückzuführen, dass solche Fahrschulen über einen Fahrzeugpark mit mehreren Fahrzeugen und Fahrzeugzusätzen, wie Anhänger oder Nummernschilder, verfügen. Zusätzlich werden mehrere Fahrlehrer, die auch im Teilzeitpensum angestellt sein können, beschäftigt. Die Problematik besteht darin, eine ideale Terminplanung unter Berücksichtigung aller Anforderungen dieser Ressourcen zu führen. Erschwerend hinzu kommt, dass eine externe Interessensgruppe in Form der Fahrschüler in die Terminplanung Einfluss nimmt.

Eine herkömmliche Lösung für dieses Problem stellen physische Agenden dar. Hierbei wird für jede Ressource (Fahrlehrer, Fahrzeug, Fahrzeugzusatz) eine eigene Agenda in Papierform geführt. Problematisch an dieser Lösung ist die Abstimmung der einzelnen Agenden untereinander. Um eine solche Terminsynchronisation vorzunehmen, muss jeweils eine Kontaktaufnahme zwischen den Fahrlehrern erfolgen. Dabei kann es vorkommen, dass nicht alle Agenden zugänglich sind.

Dies führt zu einem grossen zeitlichen Mehraufwand für die betroffenen Fahrlehrer und Fahrschulen. Es besteht zudem bei ungenügender Absprache die Möglichkeit von Terminkollisionen. Diese wirken sich negativ auf die Kundenzufriedenheit aus.

Zur Vereinfachung der Ressourcen- und Terminplanung von Fahrschulen wird deshalb eine Webapplikation auf der Basis von PHP entwickelt.

Vorgehen

Um die Anforderungen an die Softwarelösung zu eruieren, wird anhand einer mittelgrossen Fahrschule eine Anforderungsanalyse durchgeführt. Diese hat zum Ziel, in einem ersten Schritt die Probleme einer Lösung mittels physikalischen Agenden zu identifizieren. Ausgehend von dieser Ist-Analyse werden die Anforderungen an die Webapplikation in Form von Use Cases festgehalten.

Bei der Entwicklung der Lösung wird auf gängige Web-Standards gesetzt. Um die Wartbarkeit des Produkts sicherzustellen, werden solide und weitverbreitete Web Application Frameworks, wie Symfony2 und Doctrine, eingesetzt.

Bei der Entwicklung des Web-Frontends wird das Hauptaugenmerk auf eine responsive

Darstellung gelegt. Dies ermöglicht den Einsatz der Webapplikation auf verschiedenen Geräten wie Tablets, Smartphones oder gängigen Desktop-Computern. Zur Unterstützung der Frontend-Entwicklung wird das Framework Bootstrap verwendet.

Die Fahrschuldaten werden zentral in einer MySQL-Datenbank abgelegt.

Ergebnis

Die im Umfang der Bachelorarbeit erstellte rollenbasierte Webapplikation vereinfacht die Ressourcen- und Terminplanung für Fahrschulen nachhaltig. Damit können anhand einer intuitiven Weboberfläche in Form eines Kalenders neue Fahrlektionstermine unter Berücksichtigung der benötigten Ressourcen vereinbart werden. Der zugrundeliegende Prozess gestaltet sich durch die Eliminierung des Kommunikationsoverheads sehr effizient und zeitsparend.

Der standortunabhängige Zugriff auf Fahrschuldaten, wie Informationen zu den Fahrschülern, kann durch autorisierte Personen jederzeit vorgenommen werden. Dies wird durch die zentrale, redundanzfreie Datenhaltung sichergestellt.

Die iCal-Schnittstelle ermöglicht es dem Fahrlehrer zudem, seine Termine direkt in einen kompatiblen Kalender zu integrieren. Dies erlaubt unter anderem das Einrichten von Terminbenachrichtigungen auf dem Smartphone.

Das moderne, responsive Interface ermöglicht die Interaktion mit der Applikation unabhängig des eingesetzten Anzeigegeräts. Dieser moderne Auftritt der Fahrschule gegenüber dem Fahrschüler wirkt sich überdies positiv auf das Image des Unternehmens aus.

Ausblick

Im Rahmen der Anforderungsanalyse werden optionale Features definiert. Diese umfassen die Erstellung von Rechnungen, das Einsehen von Statistiken sowie die Verwaltung von Lernhinhalten und Lernfortschritten.

Genannte optionale Features können problemlos in einem Nachfolgeprojekt implementiert werden. Da bei der Entwicklung der Webapplikation viel Wert auf Wartbarkeit gelegt wird, stellt diese eine solide Basis dafür dar.

Danksagung

Wir bedanken uns bei folgenden Personen für Ihre Unterstützung während der Bachelorarbeit:

- Prof. Dr. Andreas Rinkel für die kompetente Betreuung unserer Bachelorarbeit.
- Der Ansprechperson des Industriepartners für die konstruktive Zusammenarbeit.
- Christoph Buchli, Christoph Bühler, Jonas Schwertfeger und Lorenz Bösch für das Korrekturlesen unseres Berichts.
- Manuel Alabor, Alexandre Joly und Michael Weibel für das Bereitstellen ihres LaTeX-Templates für die Dokumentation.
- Unseren Freunden, Verwandten und Partnerinnen für Geduld, Rat und motivierende Worte.

Inhaltsverzeichnis

Typografische Konventionen

Folgende typografischen Konventionen werden verwendet:

- Begriffe und Abkürzungen, die *kursiv und blau* geschrieben sind, werden im Glossar aufgeführt und erläutert.
- Literaturverweise werden in folgender Form gekennzeichnet: [AutorJahreszahl]
- Verweise auf Kapitel, Abbildungen oder Tabellen innerhalb des Berichts sind in blau gehalten.
- URLs werden entsprechend gekennzeichnet: *http://www.hsr.ch* (Beispiel mit Verweis auf HSR-Webseite)
- Code-Beispiele werden folgendermassen hervorgehoben:

```
1 <?php
2
3 class HelloWorld
4 {
5     public function sayHello()
6     {
7       echo 'Hello World!';
8     }
9 }
```

Kapitel 1 **Einführung**

1.1. Einleitung

Diese Arbeit hat gemäss Aufgabenstellung auf Seite 2 zum Ziel, eine webfähige Applikation für eine Fahrschule zu entwickeln.

Die Applikation soll bestehende Prozesse vereinheitlichen und zugleich optimieren. Dies umfasst zum einen die Verwaltung von Ressourcen innerhalb der Fahrschule. Eine weitere Anforderung besteht in einer vereinfachten Vorgehensweise zum Finden passender Termine unter Berücksichtigung der verschiedenen Ressourcen (Fahrlehrer, Fahrschüler, Fahrzeuge sowie Fahrzeugzusätze).

Die Applikation soll technisch gesehen auf aktuellen Technologien aus dem Webbereich aufbauen. So kann sichergestellt werden, dass die Lösung auch zukünftigen Anforderungen der Fahrschule gerecht werden kann.

Die Webapplikation muss für die Bedienung auf Tablets optimiert werden, soll aber auch auf Desktop-Computern und Smartphones bedienbar sein.

1.2. Ausgangslage: Planungs- und Verwaltungsabläufe der Fahrschule

Die mittelgrosse Fahrschule wird als Aktiengesellschaft geführt. Der Inhaber nimmt hierbei die Rolle des Firmenchefs sowie des Fahrzeugverantwortlichen ein, bildet aber zugleich als Fahrlehrer auch Fahrschüler aus. Die Firma beschäftigt zurzeit drei Fahrlehrer, wobei der Inhaber mit eingerechnet ist.

Die Fahrschule bildet in erster Linie Fahrschüler für die Kategorien C und D sowie deren Unterkategorien C1, CE und DE aus.

Folgende Grafik gibt den Aufbau und die Organisation der Fahrschule inklusive deren Mengengerüst wieder:

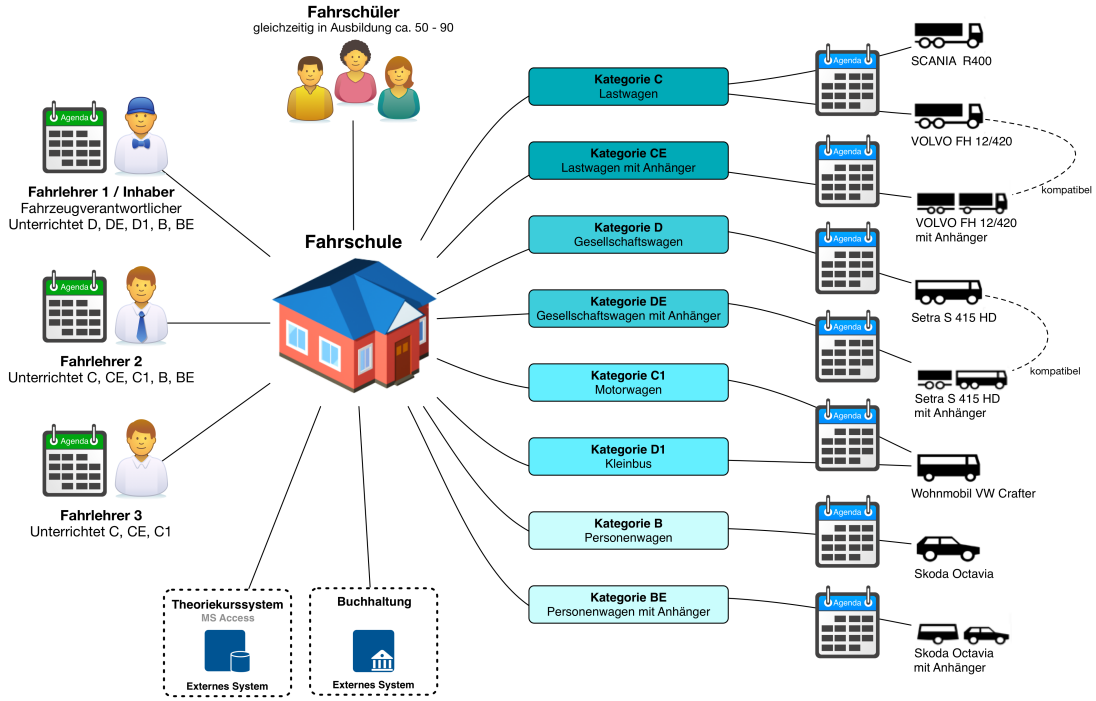


Abbildung 1.1.: Aufbau und Organisation der Fahrschule

Kategorien

Die von der Fahrschule ausgebildeten Kategorien sind durch die Schweizerischen Gesetzgebungen definiert. Eine Übersicht kann unter http://www.zh.ch/internet/sicherheitsdirektion/stva/de/StVAlf/LFkat.html eingesehen werden.

Fahrzeuge / Fahrzeugpark

Der Fahrzeugpark umfasst zwei Lastwagen, einen Car sowie einen Personenwagen. Der Personenwagen ist so ausgestattet, dass Ausbildungen für behinderte Fahrschüler möglich sind. Weiter besitzt die Fahrschule zwei Anhänger, einen für Personenwagen (entspricht Kategorie BE) sowie einen für Lastwagen (entspricht Kategorie CE).

Die Fahrzeuge müssen in regelmässigen Abständen gewartet werden. Hierfür verantwortlich zeigt sich die als Fahrzeugverantwortlicher definierte Person.

Fahrlehrer

Die Fahrlehrer, wovon es aktuell drei gibt, haben unterschiedliche Rollen innerhalb der Fahrschule zugeteilt. Hierbei unterscheidet sich *Fahrlehrer 1* von *Fahrlehrer 2* und *Fahrlehrer 3*, da er neben seiner Rolle als Ausbildender auch noch für den Unterhalt des Fahrzeugparks verantwortlich ist.

Fahrlehrer 1 und *Fahrlehrer 2* sind von der Fahrschule im Vollzeitpensum angestellt, wobei die Arbeitstage variieren können. Sie erteilen durchschnittlich an fünf Tagen die Woche zwischen Montag und Samstag Unterricht.
Fahrlehrer 3 ist im Teilzeitpensum angestellt. Das Pensum kann je nach Kundennachfrage variieren.
Alle Fahrlehrer führen ihre **eigene Agenda**, in der sie An- und Abwesenheiten eintragen. Die Agenden können jeweils nur vom Besitzer selbst eingesehen werden.

Die Fahrlehrer unterrichten verschiedene Kategorien. Dies ist teilweise auf den Umstand zurückzuführen, dass sie die gesetzlichen Befugnisse zur Ausbildung von Fahrschülern nur auf den entsprechenden Kategorien besitzen. Jedoch kann es vorkommen, dass der Fahrlehrer weitere Kategorien gesetzlich gesehen unterrichten dürfte, dies jedoch aufgrund des grossen administrativen Umfangs im Bezug auf die Terminplanung mittels physischen Agenden unterbunden wird.

Fahrschüler

Die Fahrschule bildet pro Jahr circa 120 bis 160 Fahrschüler aus. Dies resultiert in einer Anzahl von 50 bis 90 Fahrschülern, die gleichzeitig eine Ausbildung absolvieren.

Die Fahrschüler können jeweils eine Ausbildung einer Kategorie absolvieren, Parallelausbildungen auf mehreren Kategorien sind gesetzlich nicht möglich.

Agenden für Kategorien

Für jede angebotene Kategorie wird jeweils eine eigene physische Agenda geführt. Diese Agenden werden in den Fahrzeugen, welche für die auszubildenden Kategorien eingesetzt werden, aufbewahrt. Wie aus der Abbildung 1.1 hervorgeht, kann man bezüglich des Aufbewahrungsort zwei Fälle unterscheiden:

Kategorie umfasst ein Fahrzeug (Kategorien CE, D, DE, C1, D1, B, BE):
Die Agenda für die Kategorie wird im zugewiesenen Fahrzeug aufbewahrt.

Kategorie umfasst mehrere Fahrzeuge (Kategorie C):
Die Agenda für die Kategorie kann ihren Aufbewahrungsort ändern. Konkret heisst dies, dass die Agenda jeweils in dem Fahrzeug, auf welchem zuletzt gefahren wurde, aufbewahrt wird.

Theoriekurssystem

Das Theoriekurssytem wird dazu verwendet, die angebotenen Theoriekurse zu verwalten. Es werden sowohl Informationen zu den Kursen selbst, als auch Personeninformationen der Teilnehmenden erfasst.

Das Theoriekurssystem wird als *Microsoft Access*-Datenbank geführt. Eine Integration in die zu entwickelnde Lösung ist nicht vorgesehen.

Buchhaltung

Das Buchhaltungssystem wird wie das Theoriekurssystem als externe Applikation geführt. Hier werden alle für die Buchhaltung relevanten Daten zusammengetragen.

Eine Integration des Buchhaltungssystems in die zu entwickelnde Lösung ist nicht vorgesehen.

1.3. Ziel: Zentralisiertes Ressourcen- und Planungstool

Anhand der bestehenden Prozesse innerhalb der Fahrschule soll eine Webapplikation entwickelt werden, welche die Arbeit für die beteiligten Personen vereinfacht. Folgende Grafik verschafft einen Überblick über die zu implementierende Lösung:

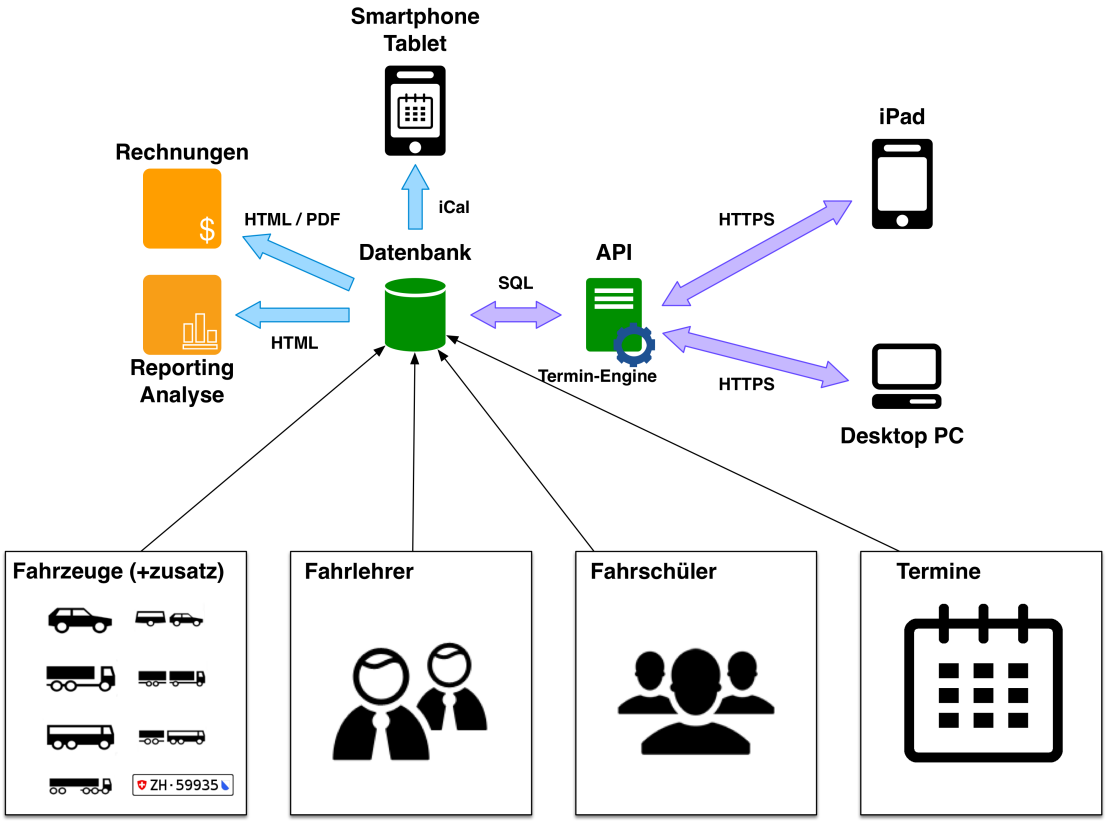


Abbildung 1.2.: Soll-Applikation

Die bestehenden Ressourcen sollen zentral verwaltet werden können. Weiter wird die Terminfindungslogik inklusive den Abhängigkeiten der verschiedenen Ressourcen in die Lösung implementiert. Ziel ist es, den Fahrlehrern ein Hilfsmittel zur Verfügung zu stellen, welches die Probleme bei der Terminfindung aus der Welt schafft.

Der Zugriff auf die Applikation muss überdies von mobilen Geräten wie Smartphones oder Tablets sowie von Desktop-Computern mit Zugang zum Internet möglich sein. Hierbei ist der Schwerpunkt auf die Entwicklung für Tablets (namentlich iPads) zu legen, da diese in der Fahrschule zum Einsatz kommen.

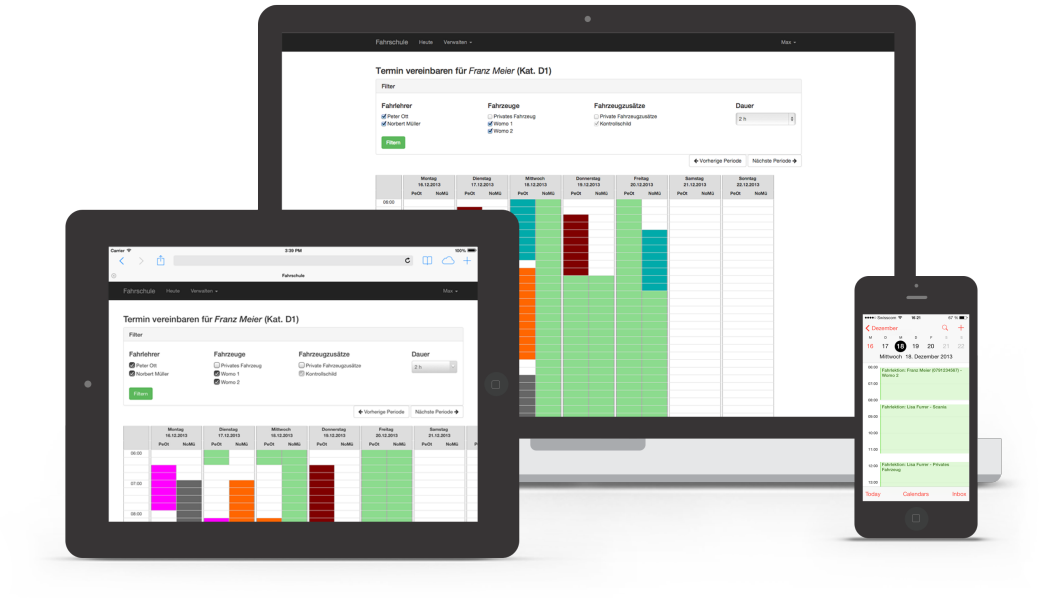


Abbildung 1.3.: Produkteübersicht

Als optionale Features sind das Erstellen von Rechnungen in elektronischer Form sowie eine Reporting- und Analysefunktion vorgesehen. Weiter wird vom Industriepartner ausgeführt, dass es nützlich wäre, wenn die Ausbildungsinhalte ebenfalls in der Applikation verwaltet werden können.
Optionale Features werden in Zusammenarbeit mit dem Industriepartner priorisiert und sind als Use Cases auf der Seite 30 einsehbar.

Kapitel 2 **Analyse der Fahrschule**

2.1. Ist Szenarien

Gemäss dem ausgearbeiteten Fragenkatalog anonymisiert sowie den gewonnen Erkenntnissen anlässlich des Kickoff-Meetings werden verschiedene „Ist Szenarien“ ausgearbeitet. Diese widerspiegeln die Prozesse, wie sie heute in der Fahrschule vorliegen.

Das den Szenarien zugrunde liegende Mengengerüst kann in Abbildung 1.1 eingesehen werden.

2.1.1. Eintritt neuer Fahrschüler

Akteure: Fahrschüler, Fahrlehrer, Sekretariat
Auslöser: Ein Fahrschüler möchte sich für Fahrkurse bei der Fahrschule anmelden.

Normalablauf

1. Der Fahrschüler kontaktiert einen Fahrlehrer.
2. Der Fahrlehrer erfasst den Namen und die Telefonnummer auf einem neuen Schülerblatt.
3. Ein Termin für die erste Fahrstunde wird vereinbart: → Szenario „Fahrlektionstermin vereinbaren“

Alternative Abläufe

1. a) Der Fahrschüler kontaktiert das Sekretariat der Fahrschule: Da das Sekretariat keinen Zugriff auf die Agenden der Resourcen verfügt, vermittelt es den Fahrschüler an einen Fahrlehrer.

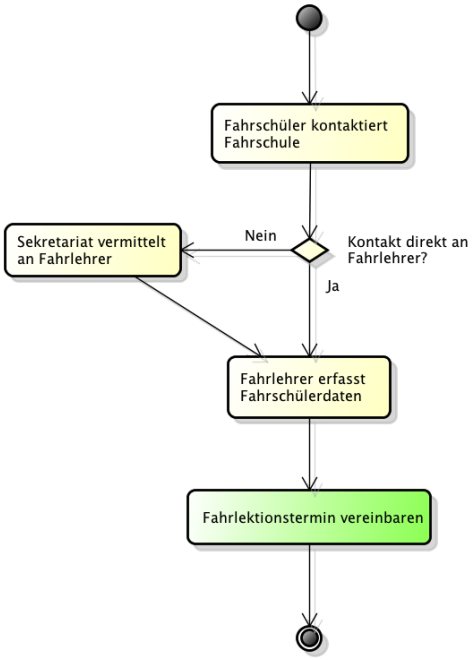


Abbildung 2.1.: Aktivitätsdiagramm Eintritt neuer Fahrschüler

2.1.2. Fahrstunde durchführen

Akteure: Fahrschüler, Fahrlehrer
Auslöser: Es wurde ein Termin für eine Fahrstunde vereinbart.

Normalablauf

1. Der Fahrschüler und der Fahrlehrer treffen sich an abgemachtem Ort und Zeit.
2. Die Ausbildungsinhalte für die gewünschte Lektion werden gemäss Logbuch durchgeführt.
3. Am Ende der Fahrstunde wird diese auf dem Schülerblatt notiert. Diese Angaben sind für die Verrechnung der Stunden relevant.
4. Ein Termin für die nächste Fahrstunde wird vereinbart: → Szenario „Fahrlektionstermin vereinbaren“

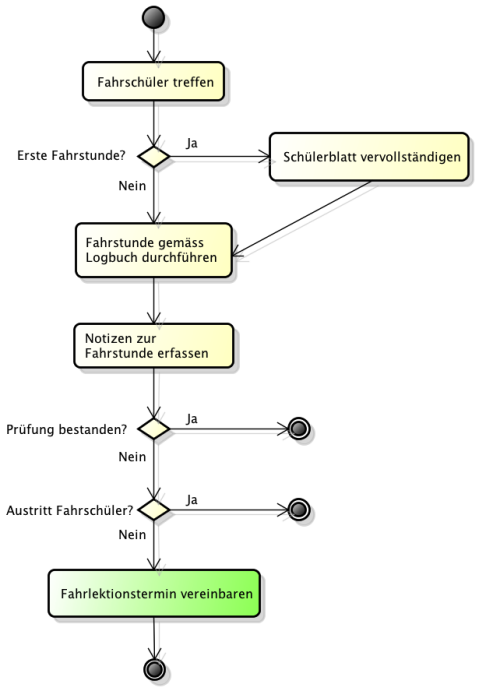


Abbildung 2.2.: Aktivitätsdiagramm Fahrstunde durchführen

Alternative Abläufe

1. a) Es handelt sich um die erste Fahrstunde: Die Angaben auf dem „Schülerblatt“ werden vervollständigt.

2. a) Bei der vorangehenden Fahrstunde hat man bereits gewisse Themen der jetzigen Fahrstunde behandelt (gekennzeichnet mit einer horizontalen Linie im Logbuch): Es wird an der Stelle weitergefahren, an welcher man beim letzten Mal aufgehört hat.

4. a) Der Fahrschüler besucht an demselben Tag die Fahrprüfung und besteht diese: Es wird kein nächster Termin vereinbart.

   b) Der Fahrschüler tritt aus der Fahrschule aus: Es wird kein nächster Termin vereinbart.

2.1.3. Fahrlektionstermin vereinbaren

Akteure: Fahrschüler, Fahrlehrer
Auslöser: Ein neu in die Fahrschule eingetretener Fahrschüler möchte eine erste Fahrstunde besuchen oder eine Fahrstunde wurde erfolgreich durchgeführt.

Normalablauf

1. Der Fahrlehrer sucht in der Agenda des Fahrzeugtyps einen freien, möglichen Termin. Dieser Termin hängt sowohl von der Verfügbarkeit des Fahrzeuges, wie auch von der Verfügbarkeit berechtigter Fahrlehrer ab. Weiter kann bei gewissen Kategorien auch noch die Verfügbarkeit von Anhängern oder Nummernschildern in die Terminfindung einfliessen.
2. Der Fahrlehrer schlägt dem Fahrschüler einen Termin vor.
3. Der Termin passt dem Fahrschüler.
4. Der passende Termin wird in der Agenda des Fahrzeugtyps eingeschrieben. Dabei werden folgende Informationen notiert: Datum und Zeit des Termin, Fahrlehrer, Fahrschüler, Fahrzeug.

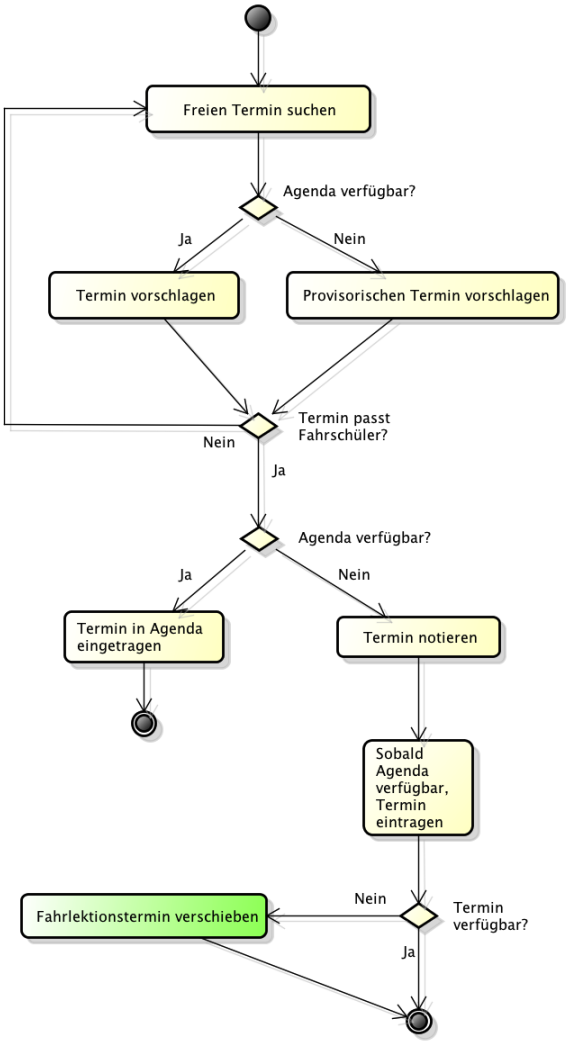


Abbildung 2.3.: Aktivitätsdiagramm Fahrlektionstermin vereinbaren

Alternative Abläufe

1. a) Die Agenda des Fahrzeugtyps ist für den Fahrlehrer nicht verfügbar (befindet sich in einem anderen Fahrzeug):
   i. Der Fahrlehrer schlägt dem Fahrschüler einen provisorischen Termin vor.
   ii. Weiter mit Schritt 3
3. a) Der Termin passt für den Fahrschüler nicht: Schritt 1 nochmals durchführen

4. a) Die Agenda des Fahrzeugtyps ist für den Fahrlehrer nicht verfügbar (befindet sich in einem anderen Fahrzeug):
   i. Der Fahrlehrer notiert sich den provisorischen Termin
   ii. Sobald die Agenda wieder zur Verfügung steht, wird der Termin eingetragen.
   ii. a) Der provisorische Termin ist bereits belegt: Der Termin wird verschoben → Szenario „Fahrlektionstermin verschieben“

2.1.4. Verfügbarkeit eines Fahrlehrers planen

Akteure: Fahrlehrer
Auslöser: Monatlich werden die Präsenzzeiten der Fahrlehrer geplant.

Normalablauf

1. Der Fahrlehrer trägt seine Präsenzzeiten für den kommenden Monat in die Agenden der Fahrzeugtypen, für die er Unterricht anbietet, ein.
2. Der Fahrlehrer trägt seine Abwesenheiten für den kommenden Monat in die Agenden der Fahrzeugtypen, für die er Unterricht anbietet, ein.

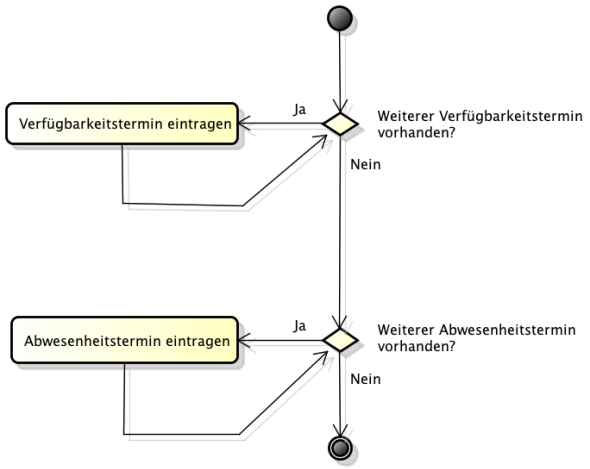


Abbildung 2.4.: Aktivitätsdiagramm Verfügbarkeit einer Ressource planen

2.1.5. Verfügbarkeit eines Fahrzeugs planen

Akteure: Fahrzeugverantwortlicher
Auslöser: Da gewisse Fahrzeuge nur teilweise zur Verfügung stehen (zum Beispiel infolge Ausleihe von/an anderen Fahrschulen), werden die Verfügbarkeiten monatlich neu geplant.

Normalablauf

→ Ablauf analog zu Normalablauf in Szenario „Verfügbarkeit eines Fahrlehrers planen“

2.1.6. Fahrzeug reparieren

Akteure: Fahrzeugverantwortlicher
Auslöser: Ein Fahrzeug wird regelmässig gewartet. / Ein Fahrzeug wird beschädigt und muss repariert werden.

Normalablauf

1. Der Fahrzeugverantwortliche kontaktiert den Garagisten.
2. Der Fahrzeugverantwortliche vereinbart einen Termin für die Wartung / Reparatur des Fahrzeugs.
3. Der Fahrzeugverantwortliche prüft anhand der Agenda des Fahrzeugs, ob in der Zeitspanne der Wartung / Reparatur Fahrstunden existieren.

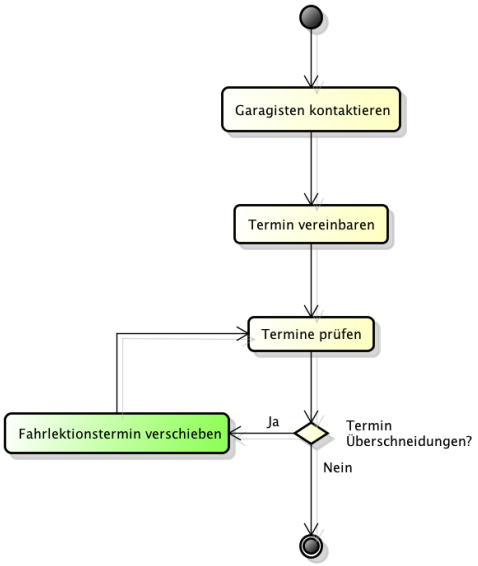


Abbildung 2.5.: Aktivitätsdiagramm Fahrzeug reparieren

Alternative Abläufe

3. a) In der Zeitspanne der Wartung / Reparatur sind Fahrstunden vorgesehen: Der Fahrzeugverantwortliche verschiebt die betroffenen Fahrlektionstermine → Szenario „Fahrlektionstermin verschieben“

2.1.7. Fahrlektionstermin verschieben

Akteure: Fahrlehrer / Fahrzeugverantwortlicher, Fahrschüler
Auslöser: Ein Fahrlehrer kann einen Fahrlektionstermin nicht wahrnehmen (infolge Krankheit, etc.). / Ein Fahrschüler kann einen Fahrlektionstermin nicht wahrnehmen. / Ein Fahrzeug fällt aus.

Normalablauf

1. Der betroffene Fahrschüler wird kontaktiert (Telefon, E-Mail, etc.).
2. Der bestehende, nicht mehr gültige Fahrlektionstermin wird aus der Agenda entfernt.
3. Ein neuer Fahrlektionstermin wird vereinbart → Szenario „Fahrlektionstermin vereinbaren“

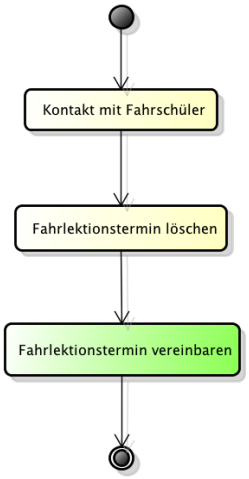


Abbildung 2.6.: Aktivitätsdiagramm Fahrlektionstermin verschieben

Alternative Abläufe

1. a) Der Fahrschüler selbst möchte den Termin verschieben: Der Fahrschüler kontaktiert den für seinen Fahrlektionstermin zuständigen Fahrlehrer.

2.1.8. Anstellung eines neuen Fahrlehrer

Akteure: Fahrlehrer
Auslöser: Ein neuer Fahrlehrer hat den Bewerbungsprozess durchlaufen und wird angestellt.

Normalablauf

1. Der neue Fahrlehrer erscheint am ersten Arbeitstag.
2. Der neue Fahrlehrer plant seine Verfügbarkeit → Szenario „Verfügbarkeit eines Fahrlehrers planen“

2.1.9. Kauf eines neuen Fahrzeugs

Akteure: Fahrzeugverantwortlicher
Auslöser: Die Fahrschule entscheidet sich, ein neues Fahrzeug anzuschaffen.

Normalablauf

1. Das Fahrzeug wird durch den Fahrzeugverantwortlichen beschafft.
2. Sobald das Fahrzeug geliefert wurde, plant der Fahrzeugverantwortliche dessen Verfügbarkeiten → Szenario „Verfügbarkeit eines Fahrzeugs planen“

2.1.10. Rechnungen ausstellen

Akteure: Fahrlehrer, Fahrschüler
Auslöser: Der Fahrschüler als Privatkunde hat 10 - 15 unbezahlte Fahrlektionstermine abgeschlossen. / Die offerierte Anzahl Fahrlektionstermine eines Fahrschülers, der einer Firma angehört, ist erreicht.

Normalablauf

1. Der Fahrlehrer ermittelt am Ende einer Fahrlektion die Anzahl unbezahlter Fahrlektionen. Dies tut er anhand des Schülerblatts.
2. Der Fahrlehrer überträgt die Anzahl unbezahlter Fahrlektionen auf das Rechnungsblatt.
3. Der Fahrlehrer berechnet anhand der Anzahl unbezahlter Fahrlektionen und dem Preis pro Fahrlektion das Rechnungstotal.
4. Der Fahrlehrer überträgt die Adressdaten des Fahrschülers vom Schülerblatt auf das Rechnungsblatt.
5. Der Fahrlehrer übergibt das Rechnungsblatt dem Fahrschüler.

Alternative Abläufe

4. a) Die Fahrstunden des Fahrschülers werden durch seinen Arbeitgeber bezahlt: Der Fahrlehrer überträgt die Adressdaten der Firma auf das Rechnungsblatt.
   b) Die Fahrstunden des Fahrschülers werden durch eine Sozialbehörde (zum Beispiel Arbeitslosenversicherung) bezahlt: Der Fahrlehrer überträgt die Adressdaten der Sozialbehörde auf das Rechnungsblatt.

5. a) Der Rechnungsempfänger wünscht einen Postversand der Rechnung: Der Fahrlehrer verschickt die Rechnung nach Abschluss der Fahrlektion per Post an den entsprechenden Adressanten.

   b) Der Rechnungsempfänger wünscht einen Versand der Rechnung per E-Mail: Der Fahrlehrer verschickt die Rechnung nach Abschluss der Fahrlektion per E-Mail an den entsprechenden Adressanten.

2.2. Problemstellung der heutigen Lösung mit physischen Agenden

Anhand der Ist Szenarien ist ersichtlich, dass die heutige Vorgehensweise einige Probleme mit sich bringt.

2.2.1. Mehrere Agenden

Ein Hauptproblem stellen die verschiedenen Agenden, wovon es je eine pro Fahrzeugtyp gibt, dar. Darin werden die verschiedenen dem Fahrzeugtypen zugeteilten Fahrzeuge und Fahrzeugzusätze, wie Anhänger oder Nummernschilder, verwaltet. Die Fahrlehrer wiederum tragen ihre Verfügbarkeiten und Termine in diesen Agenden ein. Dies führt dazu, dass ein Fahrlehrer nur für den Unterricht einer Kategorie eingesetzt werden kann.

2.2.2. Agenden nicht für alle zugänglich

Da es sich bei den Agenden um Kalender auf Papier handelt, kann es vorkommen, dass ein physischer Zugriff darauf nicht möglich ist. Die wichtigsten dieser Fälle umfassen folgende Szenarien:

Fahrlehrer kann keine fixen Termine vereinbaren

Gibt es pro Fahrzeugtyp mehrere Fahrzeuge, kann dies dazu führen, dass ein Fahrlehrer lediglich einen provisorischen Termin vereinbaren kann. Dieser Umstand rührt daher, dass zwei Fahrlehrer mit zwei verschiedenen Fahrzeugen gleichzeitig Unterricht erteilen können. Da hierbei lediglich eine Agenda für beide Fahrzeuge geführt wird, kann nur einer der beiden Fahrlehrer diese Agenda auf sich tragen. Der andere Fahrlehrer muss im Anschluss an die Fahrstunde einen provisorischen Termin mit dem Fahrschüler vereinbaren. Dieser provisorische Termin muss sobald der Fahrlehrer physischen Zugriff auf die Agenda erhält, verifiziert werden. Sollte eine Überschneidung vorliegen, muss der betroffene Fahrschüler nochmals kontaktiert und ein neuer Termin vereinbart werden. Dies führt zu einem unnötigen Mehraufwand und bietet dem Kunden einen schlechten Service.
→ Siehe hierzu auch „Ist Szenario Fahrlektionstermin vereinbaren“

Sekretariat kann keine Termine vereinbaren

Da die verschiedenen Agenden jeweils in den Fahrzeugen aufbewahrt werden und sich während den Fahrlektionen bei einem Fahrlehrer befinden, ist es dem Sekretariat nicht möglich, Fahrlektionstermine zu vereinbaren.
Dies wirkt sich in erster Linie dann aus, wenn ein neuer Fahrschüler in die Fahrschule eintritt (entspricht „Ist Szenario Eintritt neuer Fahrschüler“). Der Fahrschüler ruft dabei auf dem Sekretariat an und muss von diesem zugleich für die Vereinbarung eines ersten Fahrlektionstermin an einen der Fahrlehrer weitervermittelt werden. Hierbei kann es vorkommen, dass der vermittelte Fahrlehrer keine Ressourcen für einen weiteren Fahrschüler hat und dieser an einen weiteren Fahrlehrer weiterverwiesen werden muss. Dies stellt einen unnötigen Mehraufwand sowohl für

das Sekretariat, als auch für den Fahrlehrer und Fahrschüler dar.
Weiter kommt es vor, dass Fahrschüler einen Fahrlektionstermin zum Beispiel infolge Krankheit nicht wahrnehmen können. Die betroffenen Fahrschüler rufen nicht selten direkt auf der Sekretariatsnummer an, wobei ihnen nicht sofort bei der weiteren Terminfindung weitergeholfen werden kann. Stattdessen vermittelt das Sekretariat den Schüler an den zuständigen Fahrlehrer weiter.

Fahrlehrer kann keine Termine für Fahrlehrer einer anderen Kategorie erfassen
Da die Agenden jeweils pro Fahrzeugkategorie geführt werden ist es unmöglich, dass ein Fahrlehrer einen Termin für eine Fahrzeugkategorie, die er selbst nicht unterrichtet, vereinbaren kann. Hierzu muss er zuerst einen Fahrlehrer der gewünschten Kategorie verständigen, der dann die Terminvereinbarung vornimmt. Dies wirkt sich in erster Linie bei den Inhabern der Fahrschule aus, da die beiden die Fahrschule teilweise gemeinsam führen. In dieser Hinsicht wirken sich Absenzen von einem Fahrlehrer, der eine ganze Fahrzeugkategorie alleine führt, gravierend aus. Möchte ein Fahrschüler während einer solchen Absenz einen Termin für genannte Fahrzeugkategorie vereinbaren ist dies faktisch unmöglich und der betroffene Schüler muss auf einen Terminfindungstermin nach der Absenz des Fahrlehrers vertröstet werden.

2.2.3. Informationen zum Fahrschüler nicht jederzeit verfügbar

Da die Schülerblätter jeweils in den Fahrzeugen aufbewahrt werden, ist ein ständiger Zugriff auf die Fahrschülerinformationen nicht gewährleistet. Das ist dann problematisch, wenn ein Fahrlehrer einen seiner Fahrschüler ausserhalb seiner Präsenszeit kontaktieren möchte. Ein Grund hierfür stellt zum Beispiel das Verschieben eines Fahrlektionstermin dar. Da ein Fahrlehrer zeitweise bis zu zwanzig Fahrschüler unterrichtet, gestaltet sich das ständige Mitführen aller Schülerinformationen unpraktikabel. Tritt ein solcher Fall ein, bleibt dem Fahrlehrer nichts anderes übrig, als nochmals in die Fahrschule zu fahren, um physischen Zugriff auf das entsprechende Schülerblatt zu erhalten.

2.2.4. Das Ausstellen von Rechnungen gestaltet sich umständlich

Die heutige Situation ist im Bezug auf das Ausstellen von Rechnungen nicht zufriedenstellend. Dies wirkt sich vor allem in folgenden zwei Szenarien aus:

Rechnungen müssen von Hand erfasst werden
Da momentan die Information über die Anzahl absolvierter Fahrstunden lediglich in Papierform auf dem Schülerblatt vorliegt, muss eine Rechnung von Hand ausgestellt werden. Hierbei müssen jeweils alle für die Rechnung relevanten Informationen wie Rechnungsadresse oder Preis pro Fahrstunde mühsam von Hand vom Schülerblatt auf das Rechnungsblatt übertragen werden. Weiter muss die Anzahl der verrechneten Fahrstunden von Hand kalkuliert und anschliessend jede verrechnete Fahrstunde auf dem Schülerblatt einzeln als „bezahlt“deklariert werden.

Rechnungen können nicht durch eine administrative Hilfskraft ausgestellt werden

Eine einheitliche Stelle, die für das Erstellen von Rechnungen zuständig ist, kann zum jetzigen Zeitpunkt nicht geschaffen werden. Grund dafür ist die fehlende Einsicht in die Schülerblätter, welche die Informationen über die unbezahlten Fahrstunden beinhalten. Der physische Zugriff auf diese Informationen ist lediglich den Fahrlehrern vorbehalten.

Kapitel 3 Anforderungen an das Planungs-Tool

3.1. Funktionale Anforderungen

3.1.1. Actors & Stakeholders

Actors sind reale Personen oder Drittsysteme, welche die Applikation bedienen oder Einfluss auf die darin abgewickelten Prozesse nehmen. Stakeholders bezeichnen Ansprechsgruppen und deren Erwartungen bezüglich der Software-Lösung.

Im Anschluss sind die Actors und Stakeholders des Termin Planungs-Tools aufgeführt.

Fahrlehrer

Ein Fahrlehrer ist ein Mitarbeiter der Fahrschule, welcher die Aufgabe hat, Fahrschüler auszubilden. Der Fahrlehrer erfasst den Lernfortschritt der Fahrschüler, vereinbart Fahrlektionstermine und pflegt Fahrschülerdaten. Da ein Fahrlehrer im Teilzeitpensum eingestellt werden kann, muss es ihm überdies möglich sein, seine An- und Abwesenheit im System zu hinterlegen.

Administrator

Ein Administrator ist für den administrativen Teil der Fahrschule zuständig. Er ist zuständig für die Verwaltung aller Basisdaten, Ressourcen und Fahrschüler. Zudem unterstützt er die Fahrlehrer bei der Terminverwaltung.

Mitarbeiter

Der Mitarbeiter bezeichnet einen Administrator oder einen Fahrlehrer.

Fahrschüler

Der Fahrschüler ist der Kunde der Fahrschule und ist daran interessiert an einer gewissen Anzahl Fahrlektion-Terminen, auf die Prüfung der entsprechenden Fahrkategorie vorbereitet zu werden.

3.1.2. Use Case Diagramm

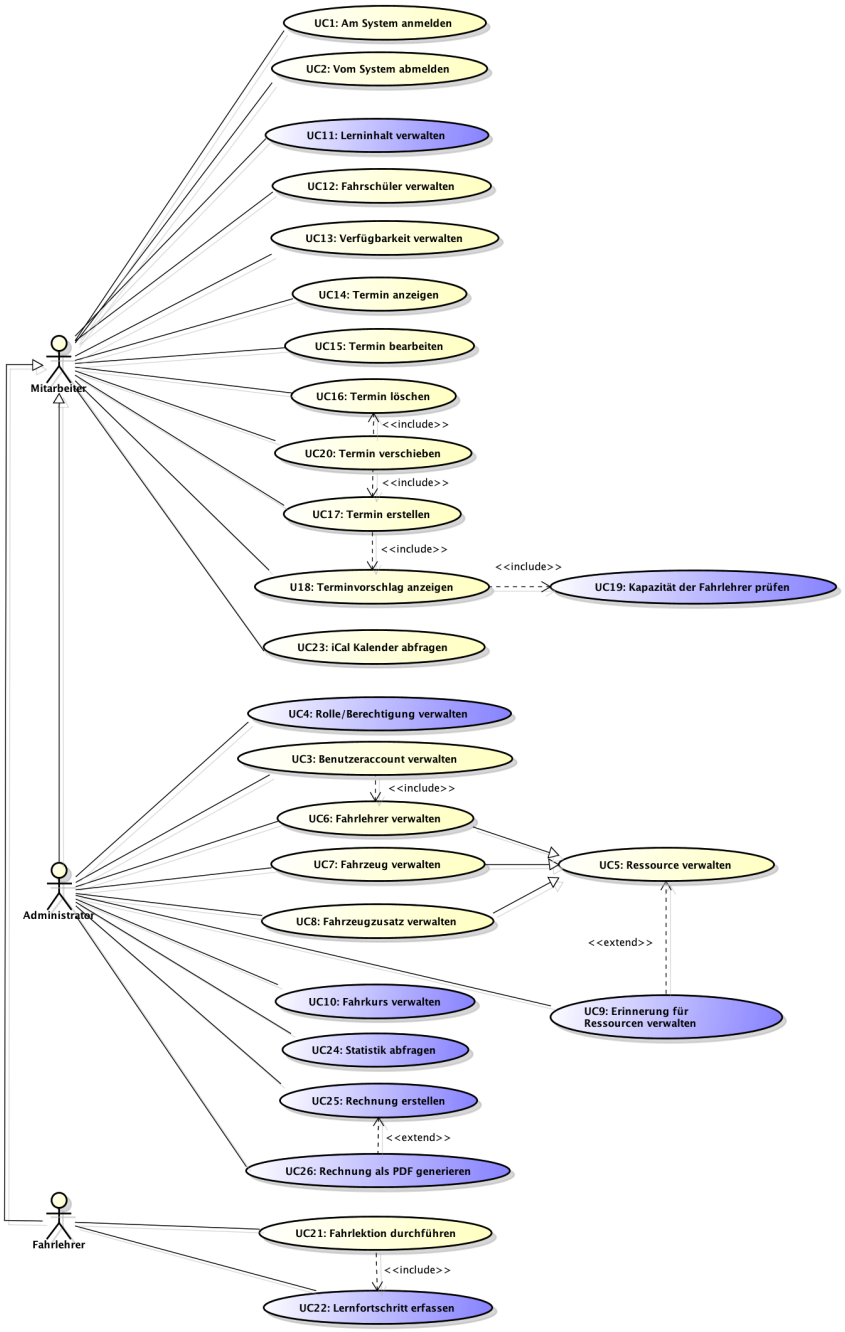


Abbildung 3.1.: Use Case Diagramm

Farbcodierungen: Gelb: Mandatory / Violett: Optional

3.1.3. Use Cases: Übersicht

Folgende Tabelle soll einen Überblick auf die erarbeiteten Use Cases, aufgeteilt in Mandatory und Optional Anforderungen, geben. Die optionalen Use Cases werden durch den Industriepartner priorisiert. Die Priorisierung ist in der Spalte „Priorität“ ersichtlich.

Alle Use Cases sind ebenfalls im Projektverwaltungstool (*http://git.smartive.ch/peter/bachelorarbeit/issues?label_name=usecase&state=all*) erfasst.

Use Case	Ticket in Gitlab	Aufwand [d]	Priorität
UC1: Am System anmelden	#13	0.5	M
UC2: Vom System abmelden	#14	0.5	M
UC3: Benutzeraccount verwalten	#20	1	M
UC5: Ressource verwalten	#15	0.5	M
UC6: Fahrlehrer verwalten	#18	1	M
UC7: Fahrzeug verwalten	#16	1	M
UC8: Fahrzeugzusatz verwalten	#17	1	M
UC12: Fahrschüler verwalten	#24	1	M
UC13: Verfügbarkeit verwalten	#25	2	M
UC14: Termin anzeigen	#26	0.5	M
UC15: Termin bearbeiten	#27	0.5	M
UC16: Termin löschen	#28	0.5	M
UC17: Termin erstellen	#29	3	M
UC18: Terminvorschlag anzeigen	#30	6	M
UC20: Termin verschieben	#32	0.5	M
UC21: Fahrlektion durchführen	#33	0.5	M
UC23: iCal Kalender abfragen	#35	2	M
UC11: Lerninhalt verwalten	#23	1	O Prio 1
UC22: Lernfortschritt verwalten	#34	1	O Prio 2
UC25: Rechnung erstellen	#37	2	O Prio 3
UC26: Rechnung als PDF generieren	#38	1	O Prio 4
UC9: Erinnerung für Ressource verwalten	#19	2	O
UC10: Fahrkurs verwalten	#22	1	O
UC4: Rollen/Berechtigungen verwalten	#21	2	O
UC19: Kapazität der Fahrlehrer prüfen	#31	0.5	O
UC24: Statistik abfragen	#36	3	O

Legende „Priorität“: M = Mandatory, O = Optional

Tabelle 3.1.: Use Cases: Übersicht

Dies ergibt zusammengefasst folgende Aufwandschätzung:

Priorität	Geschätzter Gesamtaufwand [d]
Mandatory	22
Optional	13.5

Tabelle 3.2.: Use Cases: Übersicht

3.1.4. Use Cases

Die Use Cases werden gemäss den Vorgaben aus dem Buch „Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development“ von C. Larmann [Lar] verfasst. Auf die Beschreibung der „Special Requirements“ und „Technology and Data Variations List“ wird verzichtet, da diese im Bezug auf die Arbeit keinen Mehrwert bringen.

UC1: Am System anmelden

UC1	**Am System anmelden**
Primary Actor	Mitarbeiter
Stakeholders and Interests	• Mitarbeiter: Kann sich am System anmelden
Preconditions	• Mitarbeiter ist nicht am System angemeldet • Mitarbeiter ist im System registriert
Postconditions	• Mitarbeiter ist am System angemeldet
Basic Flow	1. Mitarbeiter gibt seine Anmeldedaten ein. 2. Anmeldedaten sind korrekt. 3. Mitarbeiter wird auf die Startseite weitergeleitet.
Alternative Flows	2.a Anmeldedaten sind nicht korrekt: Mitarbeiter wird erneut aufgefordert, die Anmeldedaten einzugeben. → Schritt 1
Frequency of Occurrence	Täglich

Tabelle 3.3.: UC1: Am System anmelden

UC2: Vom System abmelden

UC2	**Vom System abmelden**
Primary Actor	Mitarbeiter
Stakeholders and Interests	• Mitarbeiter: Kann sich vom System anmelden
Preconditions	• Mitarbeiter ist am System angemeldet
Postconditions	• Mitarbeiter ist nicht mehr am System angemeldet
Basic Flow 1. Mitarbeiter klickt auf den „Abmelden“Button. 2. Mitarbeiter wird vom System abgemeldet.	
Alternative Flows	
Frequency of Occurrence	

Tabelle 3.4.: UC2: Vom System abmelden

UC3: Benutzeraccount verwalten

UC3	**Benutzeraccount verwalten (*CRUD*)**
Primary Actor	Administrator
Stakeholders and Interests	• Administrator: Kann einen Benutzeraccount des Systems erstellen, einsehen, ändern, oder löschen
Preconditions	• Administrator ist am System angemeldet **Read, Update, Delete** • Es ist mindestens ein Benutzeraccount im System vorhanden
Postconditions	**Create** • Neuer Benutzeraccount ist erstellt **Update** • Änderung am Benutzeraccount ist persistent gespeichert **Delete** • Benutzeraccount ist gelöscht

Basic Flow

Create

1. Administrator wählt „Benutzeraccount erstellen“
2. Administrator erfasst Daten zu Benutzeraccount
3. System speichert neuen Benutzeraccount

Read

1. Administrator wählt Benutzeraccount aus Benutzeraccount-Liste
2. System zeigt dem Administrator die Informationen zum Benutzeraccount an

Update

1. Administrator wählt Benutzeraccount aus Benutzeraccount-Liste
2. Administrator wählt „Benutzeraccount bearbeiten“
3. Administrator erfasst neue Daten zu Benutzeraccount
4. System speichert Änderungen an Benutzeraccount

Delete

1. Administrator wählt Benutzeraccount aus Benutzeraccount-Liste
2. Administrator wählt „Benutzeraccount löschen“
3. System löscht Benutzeraccount

Alternative Flows

Create

2.a Administrator hat nicht alle erforderlichen Angaben zu Benutzeraccount erfasst: Administrator wird aufgefordert, die fehlenden Angaben zu erfassen → Schritt 2

2.b Benutzername ist bereits im System registriert: Administrator wird aufgefordert, einen anderen Benutzernamen zu wählen → Schritt 2

Update

3.a Administrator hat nicht alle erforderlichen Angaben zu Benutzeraccount erfasst: Administrator wird aufgefordert, die fehlenden Angaben zu erfassen → Schritt 3

3.b Neuer Benutzername ist bereits im System registriert: Administrator wird aufgefordert, einen anderen Benutzernamen zu wählen → Schritt 3

Frequency of Occurrence	Halbjährlich

Tabelle 3.5.: UC3: Benutzeraccount verwalten

UC4: Rolle/Berechtigung verwalten

UC4	**Rolle/Berechtigung verwalten (*CRUD*)**
Primary Actor	Administrator
Stakeholders and Interests	• Administrator: Kann eine Rolle und deren Berechtigungen erstellen, einsehen, ändern, oder löschen • Mitarbeiter: Eine Änderung der Berechtigung für eine Rolle betrifft alle Mitarbeiter, welcher dieser Rolle angehören
Preconditions	• Administrator ist am System angemeldet **Read, Update, Delete** • Es ist mindestens eine Rolle im System vorhanden
Postconditions	**Create** • Neue Rolle ist erstellt • Berechtigungen der Rolle sind aktiv **Update** • Änderung an Rolle ist persistent gespeichert • Berechtigungen der Rolle sind aktiv **Delete** • Rolle ist gelöscht • Berechtigungen der Rolle sind inaktiv

Basic Flow

Create

1. Administrator wählt „Rolle erstellen“
2. Administrator erfasst Informationen zu Rolle
3. Administrator selektiert Funktionalitäten, für welche Rolle berechtigt ist
4. System speichert neue Rolle und deren Berechtigungen

Read

1. Administrator wählt Rolle aus Rollen-Liste
2. System zeigt dem Administrator die Informationen zur Rolle an

Update

1. Administrator wählt Rolle aus Rollen-Liste
2. Administrator wählt „Rolle bearbeiten“
3. Administrator erfasst neue Informationen zu Rolle
4. Administrator selektiert Funktionalitäten, für welche Rolle neu berechtigt ist
5. Administrator deselektiert Funktionalitäten, für welche Rolle nicht mehr berechtigt ist
6. System speichert Änderungen an Rolle und Berechtigungen

Delete

1. Administrator wählt Rolle aus Rollen-Liste
2. Administrator wählt „Rolle löschen“
3. System löscht Rolle und damit verknüpfte Berechtigungen

Alternative Flows

Create

2.a Administrator hat nicht alle erforderlichen Angaben zu Rolle erfasst: Administrator wird aufgefordert, die fehlenden Angaben zu erfassen → Schritt 2

2.b Name der Rolle ist bereits im System registriert: Administrator wird aufgefordert, einen anderen Namen für die Rolle zu wählen → Schritt 2

Update

3.a Administrator hat nicht alle erforderlichen Angaben zu Rolle erfasst: Administrator wird aufgefordert, die fehlenden Angaben zu erfassen → Schritt 3

3.b Neuer Benutzername ist bereits im System registriert: Administrator wird aufgefordert, einen anderen Namen für die Rolle zu wählen → Schritt 3

Frequency of Occurrence	Weniger als jährlich

Tabelle 3.6.: UC4: Rolle/Berechtigung verwalten

UC5: Ressource verwalten

UC5	**Ressource verwalten (*CRUD*)**
Primary Actor	Administrator
Stakeholders and Interests	• Administrator: Kann eine Ressource erstellen, einsehen, ändern, oder löschen • Mitarbeiter: Will Fahrlektionsterminverschiebungen infolge Änderungen von Ressourcen einsehen können
Preconditions	• Administrator ist am System angemeldet **Read, Update, Delete** • Es ist mindestens eine Ressource im System vorhanden

Postconditions	**Create** • Neue Ressource ist erstellt **Update** • Änderung an Ressource ist persistent gespeichert • Falls Änderung an Ressource Termine tangieren sind betroffene Termine der Ressource verschoben worden **Delete** • Ressource ist gelöscht • Termine der Ressource sind verschoben worden

Basic Flow

Create

1. Administrator wählt „Ressource erstellen“
2. Administrator erfasst Informationen zu Ressource
3. System speichert neue Ressource

Read

1. Administrator wählt eine Ressource aus Ressourcen-Liste
2. System zeigt dem Administrator die Informationen zur Ressource an

Update

1. Administrator wählt Ressource aus Ressourcen-Liste
2. Administrator wählt „Ressource bearbeiten“
3. Administrator erfasst neue Informationen zu Ressource
4. System speichert Änderungen an Ressource

Delete

1. Administrator wählt Ressource aus Ressourcen-Liste
2. Administrator wählt „Ressource löschen“
3. System löscht Ressource

Alternative Flows

Create

2.a Administrator hat nicht alle erforderlichen Angaben zur Ressource erfasst: Administrator wird aufgefordert, die fehlenden Angaben zu erfassen → Schritt 2

2.b Identifier der Ressource ist bereits im System registriert: Administrator wird aufgefordert, einen anderen Identifier für die Ressource zu wählen → Schritt 2

2.c Bei Ressource handelt es sich um Fahrzeug mit Fahrzeugzusätzen: Administrator erfasst die Fahrzeugzusätze

Update

3.a Administrator hat nicht alle erforderlichen Angaben zur Ressource erfasst: Administrator wird aufgefordert, die fehlenden Angaben zu erfassen → Schritt 3

3.b Identifier der Ressource ist bereits im System registriert: Administrator wird aufgefordert, einen anderen Identifier für die Ressource zu wählen → Schritt 3

4.a Vorgenommene Änderungen tangiert bestehenden Termin:

1. Termin wird verschoben → UC20: Termin verschieben

   Schritt 1 für jeden betroffenen Termin wiederholen

Delete

3.a Die Ressource besitzt einen Termin:

1. Termin wird verschoben → UC20: Termin verschieben

   Schritt 1 für jeden betroffenen Termin wiederholen

3.b Zur Ressource gibt es Erinnerungen: Die Erinnerungen werden gelöscht → UC9: Erinnerung für Ressource verwalten

Frequency of Occurrence	Jährlich

Tabelle 3.7.: UC5: Ressource verwalten

UC6: Fahrlehrer verwalten

Die Verwaltung eines Fahrlehrers in Form einer Ressource wird in „UC5: Ressource verwalten“ abgehandelt.
→ „UC5: Ressource verwalten“

UC7: Fahrzeug verwalten

Die Verwaltung eines Fahrzeugs in Form einer Ressource wird in „UC5: Ressource verwalten“ abgehandelt.
→ „UC5: Ressource verwalten“

UC8: Fahrzeugzusatz verwalten

Die Verwaltung eines Fahrzeugzusatz in Form einer Ressource wird in „UC5: Ressource verwalten“ abgehandelt.
→ „UC5: Ressource verwalten“

UC9: Erinnerung für Ressource verwalten

UC9	**Erinnerung für Ressource verwalten (*CRUD*)**
Primary Actor	Administrator
Stakeholders and Interests	• Administrator: – Kann eine Erinnerung für eine Ressource erstellen, einsehen, ändern, oder löschen – Wird zu festgelegtem Zeitpunkt über ein Ereignisse im Zusammenhang mit der Ressource erinnert
Preconditions	• Administrator ist am System angemeldet • Es sind Ressourcen im System vorhanden **Read, Update, Delete** • Es gibt mindestens eine Erinnerung für die Ressource
Postconditions	**Create** • Neue Erinnerung zu einer Ressource ist erstellt • Die Erinnerungen werden zu festgelegtem Zeitpunkt verschickt **Update** • Änderung an Erinnerung zu einer Ressource ist persistent gespeichert • Die Erinnerungen werden zu festgelegtem Zeitpunkt verschickt **Delete** • Erinnerung zu einer Ressource ist gelöscht • Es werden keine Erinnerungen zur Ressource mehr verschickt

Basic Flow

Create

1. Administrator wählt Ressource aus Ressourcen-Liste
2. Administrator wählt „Erinnerung erstellen“
3. Administrator erfasst Informationen zu Erinnerung
4. System speichert neue Erinnerung zu Ressource

Read

1. Administrator wählt eine Ressource aus Ressourcen-Liste
2. Administrator wählt „Erinnerungen anzeigen“
3. Administrator wählt Erinnerung aus Erinnerungs-Liste
4. System zeigt dem Administrator die Informationen zur Erinnerung an

Update

1. Administrator wählt Ressource aus Ressourcen-Liste
2. Administrator wählt „Erinnerungen anzeigen“
3. Administrator wählt Erinnerung aus Erinnerungs-Liste
4. Administrator wählt „Erinnerungen bearbeiten“
5. Administrator erfasst neue Informationen zu Erinnerung
6. System speichert Änderungen an Erinnerung zu Ressource

Delete

1. Administrator wählt Ressource aus Ressourcen-Liste
2. Administrator wählt „Erinnerungen anzeigen“
3. Administrator wählt Erinnerung aus Erinnerungs-Liste
4. Administrator wählt „Erinnerungen löschen“
5. System löscht Erinnerung zu Ressource

Alternative Flows

Create

3.a Administrator hat nicht alle erforderlichen Angaben zur Ressource erfasst: Administrator wird aufgefordert, die fehlenden Angaben zu erfassen → Schritt 3

Update

5.a Administrator hat nicht alle erforderlichen Angaben zur Ressource erfasst: Administrator wird aufgefordert, die fehlenden Angaben zu erfassen → Schritt 5

Frequency of Occurrence	Mehrmals jährlich

Tabelle 3.8.: UC9: Erinnerung für Ressource verwalten

UC10: Fahrkurs verwalten

UC10	**Fahrkurs verwalten (*CRUD*)**
Primary Actor	Administrator
Stakeholders and Interests	• Administrator: Kann einen Fahrkurs erstellen, einsehen, ändern, oder löschen • Mitarbeiter: Will über gelöschte Fahrlektionstermine informiert werden
Preconditions	• Administrator ist am System angemeldet **Read, Update, Delete** • Es ist mindestens ein Fahrkurs im System vorhanden
Postconditions	**Create** • Neuer Fahrkurs ist erstellt **Update** • Änderung an Fahrkurs ist persistent gespeichert **Delete** • Fahrkurs ist gelöscht • Termine des Fahrkurses sind gelöscht worden

Basic Flow

Create

1. Administrator wählt „Fahrkurs erstellen“
2. Administrator erfasst Informationen zu Fahrkurs
3. System speichert neuen Fahrkurs

Read

1. Administrator wählt einen Fahrkurs aus Fahrkurs-Liste
2. System zeigt dem Administrator die Informationen zum Fahrkurs an

Update

1. Administrator wählt einen Fahrkurs aus Fahrkurs-Liste
2. Administrator wählt „Fahrkurs bearbeiten“
3. Administrator erfasst neue Informationen zu Fahrkurs
4. System speichert Änderungen an Fahrkurs

Delete

1. Administrator wählt einen Fahrkurs aus Fahrkurs-Liste
2. Administrator wählt „Fahrkurs löschen“
3. System löscht Fahrkurs

Alternative Flows

Create

2.a Administrator hat nicht alle erforderlichen Angaben zum Fahrkurs erfasst: Administrator wird aufgefordert, die fehlenden Angaben zu erfassen → Schritt 2

2.b Name des Fahrkurses ist bereits im System registriert: Administrator wird aufgefordert, einen anderen Namen für den Fahrkurs zu wählen → Schritt 2

Update

3.a Administrator hat nicht alle erforderlichen Angaben zum Fahrkurs erfasst: Administrator wird aufgefordert, die fehlenden Angaben zu erfassen → Schritt 3

3.b Name des Fahrkurses ist bereits im System registriert: Administrator wird aufgefordert, einen anderen Namen für den Fahrkurs zu wählen → Schritt 3

Delete

3.a Es besteht ein Termin für den zu löschenden Fahrkurs:

a) Termin wird gelöscht

Schritt 1 für jeden betroffenen Termin wiederholen

Frequency of Occurrence	Bei Anpassung der Strassengesetze

Tabelle 3.9.: UC10: Fahrkurs verwalten

UC11: Lerninhalt verwalten

UC11	**Lerninhalt verwalten (*CRUD*)**
Primary Actor	Mitarbeiter
Stakeholders and Interests	• Mitarbeiter: Kann einen Lerninhalt erstellen, einsehen, ändern oder löschen • Fahrlehrer: Kann anhand erfasster Lerninhalte Fahrlektionen durchführen
Preconditions	• Mitarbeiter ist am System angemeldet • Es ist mindestens ein Fahrkurs im System vorhanden **Read, Update, Delete** • Es wurde mindestens ein Lerninhalt erfasst
Postconditions	**Create** • Neuer Lerninhalt ist erstellt **Update** • Änderung an Lerninhalt ist persistent gespeichert • Abgeschlossene Lerninhalte bei Fahrlektionen werden durch Änderung nicht verändert **Delete** • Lerninhalt ist gelöscht • Abgeschlossene Lerninhalte bei Fahrlektionen bleiben bestehen

Basic Flow

Create

1. Mitarbeiter wählt Fahrkurs aus Fahrkurs-Liste
2. Mitarbeiter wählt „Lerninhalt erstellen“
3. Mitarbeiter erfasst Informationen zu Lerninhalt
4. System speichert neuen Lerninhalt zu Fahrkurs

Read

1. Mitarbeiter wählt Fahrkurs aus Fahrkurs-Liste
2. Mitarbeiter wählt „Lerninhalt anzeigen“
3. Mitarbeiter wählt Lerninhalt aus Lerninhalts-Liste
4. System zeigt dem Mitarbeiter die Informationen zum Lerninhalt an

Update

1. Mitarbeiter wählt Fahrkurs aus Fahrkurs-Liste
2. Mitarbeiter wählt „Lerninhalt anzeigen“
3. Mitarbeiter wählt Lerninhalt aus Lerninhalts-Liste
4. Mitarbeiter wählt „Lerninhalt bearbeiten“
5. Mitarbeiter erfasst neue Informationen zu Lerninhalt
6. System setzt alten Lerninhalt auf inaktiv
7. System speichert neuen Lerninhalt und setzt diesen auf aktiv

Delete

1. Mitarbeiter wählt Fahrkurs aus Fahrkurs-Liste
2. Mitarbeiter wählt „Lerninhalt anzeigen“
3. Mitarbeiter wählt Lerninhalt aus Lerninhalts-Liste
4. Mitarbeiter wählt „Lerninhalt löschen“
5. System setzt Lerninhalt auf inaktiv

Alternative Flows

Create

3.a Mitarbeiter hat nicht alle erforderlichen Angaben zum Lerninhalt erfasst: Mitarbeiter wird aufgefordert, die fehlenden Angaben zu erfassen → Schritt 3

3.b Titel des Lerninhalts ist für ausgewählten Fahrkurs bereits im System registriert: Mitarbeiter wird aufgefordert, einen anderen Titel für den Lerninhalt zu wählen → Schritt 3

Update

5.a Mitarbeiter hat nicht alle erforderlichen Angaben zum Lerninhalt erfasst: Mitarbeiter wird aufgefordert, die fehlenden Angaben zu erfassen → Schritt 5

5.b Titel des Lerninhalts ist für ausgewählten Fahrkurs bereits im System registriert: Mitarbeiter wird aufgefordert, einen anderen Titel für den Lerninhalt zu wählen → Schritt 5

Frequency of Occurrence Jährlich

Tabelle 3.10.: UC11: Lerninhalt verwalten

UC12: Fahrschüler verwalten

UC12	**Fahrschüler verwalten (*CRUD*)**
Primary Actor	Mitarbeiter
Stakeholders and Interests	• Mitarbeiter: Kann einen Fahrschüler erstellen, einsehen, ändern oder löschen • Fahrschüler: Möchte Fahrlektionen besuchen können
Preconditions	• Mitarbeiter ist am System angemeldet **Read, Update, Delete** • Es ist mindestens ein Fahrschüler im System vorhanden
Postconditions	**Create** • Neuer Fahrschüler ist erstellt • Fahrschüler kann Fahrlektionen besuchen **Update** • Änderung an Fahrschüler ist persistent gespeichert • Sofern Anpassung an Fahrkurszuweisung vorgenommen wird und Fahrschüler bereits Termine für alten Fahrkurs besitzt, sind diese gelöscht **Delete** • Fahrschüler ist gelöscht • Sofern vorhanden, sind Termine des Fahrschülers gelöscht • Fahrschüler kann keine Fahrlektionen mehr besuchen

Basic Flow

Create

1. Mitarbeiter wählt „Fahrschüler erstellen“
2. Mitarbeiter erfasst Informationen zu Fahrschüler
3. System speichert neuen Fahrschüler

Read

1. Mitarbeiter wählt einen Fahrschüler aus Fahrschüler-Liste
2. System zeigt dem Mitarbeiter die Informationen zum Fahrschüler an

Update

1. Mitarbeiter wählt Fahrschüler aus Fahrschüler-Liste
2. Mitarbeiter wählt „Fahrschüler bearbeiten“
3. Mitarbeiter erfasst neue Informationen zu Fahrschüler
4. System speichert Änderungen an Fahrschüler

Delete

1. Mitarbeiter wählt Fahrschüler aus Fahrschüler-Liste
2. Mitarbeiter wählt „Fahrschüler löschen“
3. System löscht Fahrschüler

Alternative Flows

Create

2.a Mitarbeiter hat nicht alle erforderlichen Angaben zu Fahrschüler erfasst: Mitarbeiter wird aufgefordert, die fehlenden Angaben zu erfassen → Schritt 2

Update

3.a Mitarbeiter hat nicht alle erforderlichen Angaben zu Fahrschüler erfasst: Mitarbeiter wird aufgefordert, die fehlenden Angaben zu erfassen → Schritt 3

4.a Vorgenommene Änderungen an Fahrkurszuweisung tangiert bestehenden Termin:

a) Termin wird gelöscht → UC16: Termin löschen

Schritt 1 für jeden betroffenen Termin wiederholen

Delete

3.a Der Fahrschüler besitzt einen Termin:

a) Termin wird gelöscht → UC16: Termin löschen

Schritt 1 für jeden betroffenen Termin wiederholen

Frequency of Occurrence	Wöchentlich

Tabelle 3.11.: UC12: Fahrschüler verwalten

UC13: Verfügbarkeit verwalten

UC13	**Verfügbarkeit verwalten (*CRUD*)**
Primary Actor	Mitarbeiter
Stakeholders and Interests	• Mitarbeiter: Kann eine Verfügbarkeit erstellen, einsehen, ändern oder löschen • Fahrschüler: Möchte über Terminverschiebungen informiert werden
Preconditions	• Mitarbeiter ist am System angemeldet **Read, Update, Delete** • Es ist mindestens eine Verfügbarkeit für den Mitarbeiter im System vorhanden
Postconditions	**Create** • Neue Verfügbarkeit ist erstellt **Update** • Änderung an Verfügbarkeit ist persistent gespeichert • Sofern Änderung bestehenden Termin tangiert, ist dieser verschoben worden **Delete** • Verfügbarkeit ist gelöscht • Sofern Löschen der Verfügbarkeit bestehenden Termin tangiert, ist dieser verschoben worden

Basic Flow

Create

1. Mitarbeiter wählt „Verfügbarkeit erstellen“
2. Mitarbeiter wählt Typ der Verfügbarkeit: Anwesenheit / Abwesenheit
3. Mitarbeiter wählt Startdatum und Startzeit für neue Verfügbarkeit
4. Mitarbeiter wählt Enddatum und Endzeit für neue Verfügbarkeit
5. System speichert neue Verfügbarkeit

Read

1. Mitarbeiter wählt Verfügbarkeit aus der Verfügbarkeits-Liste / Kalender aus
2. System zeigt dem Mitarbeiter die Informationen zur Verfügbarkeit an

Update

1. Mitarbeiter wählt Verfügbarkeit aus der Verfügbarkeits-Liste / Kalender aus
2. Mitarbeiter wählt „Verfügbarkeit bearbeiten“
3. Mitarbeiter erfasst neue Informationen (Datum, Zeit, Notiz) zur Verfügbarkeit
4. System speichert Änderungen an Verfügbarkeit

Delete

1. Mitarbeiter wählt Verfügbarkeit aus der Verfügbarkeits-Liste / Kalender aus
2. Mitarbeiter wählt „Verfügbarkeit löschen“
3. System löscht Verfügbarkeit

Alternative Flows

Create

4.a Mitarbeiter möchte Grund für Verfügbarkeit angeben: Mitarbeiter erfasst Grund für Verfügbarkeit

Update

3.a Mitarbeiter hat Angaben zu Datum oder Zeit nicht ausgefüllt: Mitarbeiter wird aufgefordert, die fehlenden Daten zu erfassen → Schritt 3

4.a Vorgenommene Änderungen an Datum oder Zeit tangiert bestehenden Termin:

a) Termin wird verschoben → UC20: Termin verschieben

Schritt 1 für jeden betroffenen Termin wiederholen

Delete

3.a Verfügbarkeit beinhaltet Termin:

a) Termin wird verschoben → UC20: Termin verschieben

Schritt 1 für jeden betroffenen Termin wiederholen

Frequency of Occurrence	Wöchentlich

Tabelle 3.12.: UC13: Verfügbarkeit verwalten

UC14: Termin anzeigen

UC14	**Termin anzeigen**
Primary Actor	Mitarbeiter
Stakeholders and Interests	• Mitarbeiter: Kann Termine einsehen
Preconditions	• Mitarbeiter ist am System angemeldet
Postconditions	
Basic Flow	1. Mitarbeiter wählt Termin aus Termin-Liste / Kalender-Ansicht 2. System zeigt dem Mitarbeiter die Informationen zum Termin an
Alternative Flows	
Frequency of Occurrence	Ständig

Tabelle 3.13.: UC14: Termin anzeigen

UC15: Termin bearbeiten

UC15	**Termin bearbeiten**
Primary Actor	Mitarbeiter
Stakeholders and Interests	• Mitarbeiter: Kann Termine ändern
Preconditions	• Mitarbeiter ist am System angemeldet
Postconditions	
Basic Flow	1. Mitarbeiter wählt Termin aus Termin-Liste / Kalender-Ansicht 2. Mitarbeiter wählt „Termin bearbeiten“ 3. Mitarbeiter erfasst neue Daten zu Termin 4. System speichert Änderungen an Termin
Alternative Flows	3.a Mitarbeiter hat nicht alle erforderlichen Angaben zum Termin erfasst: Mitarbeiter wird aufgefordert, die fehlenden Angaben zu erfassen → Schritt 3 3.b Mitarbeiter will Termin, Dauer oder Fahrlehrer anpassen: Nicht möglich. → UC20: Termin verschieben
Frequency of Occurrence	Täglich

Tabelle 3.14.: UC15: Termin bearbeiten

UC16: Termin löschen

UC16	**Termin löschen**
Primary Actor	Mitarbeiter
Stakeholders and Interests	• Mitarbeiter: Kann Termine löschen
Preconditions	• Mitarbeiter ist am System angemeldet
Postconditions	• Termin ist gelöscht
Basic Flow	1. Mitarbeiter wählt Termin aus Termin-Liste / Kalender-Ansicht 2. Mitarbeiter wählt „Termin löschen“ 3. System löscht Termin
Alternative Flows	
Frequency of Occurrence	Täglich

Tabelle 3.15.: UC16: Termin löschen

UC17: Termin erstellen

UC17	**Termin erstellen**
Primary Actor	Mitarbeiter
Stakeholders and Interests	• Mitarbeiter: Kann Termin erstellen • Fahrschüler: Kann Termin vereinbaren
Preconditions	• Mitarbeiter ist am System angemeldet • Fahrschüler ist in direktem Kontakt mit Mitarbeiter
Postconditions	• Termin ist erstellt • Benötigte Ressourcen (Fahrzeug, Fahrzeugzusätze und Fahrlehrer) sind gebucht

Basic Flow

1. System zeigt alle Terminvorschläge an → UC18: Terminvorschlag anzeigen
2. Mitarbeiter wählt „Termin erstellen“
3. Mitarbeiter wählt nächsten möglichen Termin und schlägt diesen dem Fahrschüler vor
4. Mitarbeiter wählt „Termin buchen“
5. System erstellt Termin

Alternative Flows

3.a Termin passt dem Fahrschüler nicht → Schritt 3

5.a Terminkollision: System kann Termin nicht erstellen, da eine der benötigten Ressourcen für diesen Zeitraum bereits für einen anderen Termin gebucht wurde. → Schritt 2

Frequency of Occurrence	Stündlich (Alternative Flow 5: Eine Terminkollision kommt sehr selten vor, da selten 2 Fahrlehrer zum gleichen Zeitpunkt, für die gleiche Fahrkategorie, Termine vereinbaren)

Tabelle 3.16.: UC17: Termin erstellen

UC18: Terminvorschlag anzeigen

UC18	**Terminvorschlag anzeigen**
Primary Actor	Mitarbeiter
Stakeholders and Interests	• Mitarbeiter: Will einen neuen Termin mit bestimmten Ressourcen für einen Fahrschüler vorgeschlagen bekommen
Preconditions	• Mitarbeiter ist angemeldet und besitzt die entsprechenden Berechtigungen
Postconditions	• Mitarbeiter erhält einen Terminvorschlag

Basic Flow

1. Mitarbeiter wählt „Termin finden“
2. Mitarbeiter wählt Fahrschüler, für welchen Termin vereinbart werden soll aus Fahrschüler-Liste aus
3. System selektioniert alle für diesen Fahrschüler möglichen Filterattribute (Fahrlektionsdauer, Fahrlehrer, Fahrzeuge, Wochentage)
4. System zeigt gemäss Filter mögliche Termine (unter Berücksichtigung von → UC19: Kapazität der Fahrlehrer prüfen) in einer Kalenderansicht an

Alternative Flows

2.a Fahrschüler ist bereits vorselektiert: Weiter mit Punkt 3

3.a Mitarbeiter setzt eigene Filterkriterien (Fahrlektionsdauer, Fahrlehrer, Fahrzeuge, Wochentage)

3.b Fahrschüler stellt Fahrzeug zur Verfügung: Mitarbeiter deselektiert alle Fahrzeuge in den Filterkriterien

4.a Fahrschüler stellt Fahrzeug zur Verfügung: System zeigt Termine lediglich unter Einbezug des Fahrlehrers (unter Berücksichtigung von → UC19: Kapazität der Fahrlehrer prüfen) in einer Kalenderansicht an

Frequency of Occurrence	Stündlich

Tabelle 3.17.: UC18: Terminvorschlag anzeigen

UC19: Kapazität der Fahrlehrer prüfen

UC19	**Kapazität der Fahrlehrer prüfen**
Primary Actor	Mitarbeiter
Stakeholders and Interests	• Mitarbeiter: Berücksichtigung der rechtlichen Regelung bezüglich Arbeitsstunden eines Fahrlehrers
Preconditions	
Postconditions	• Es werden nur Fahrlehrer in Terminen berücksichtigt, welche noch Kapazitäten haben
Basic Flow	1. System filtert Termine unter Berücksichtigung der Kapazitäten der Fahrlehrer
Alternative Flows	
Frequency of Occurrence	Stündlich

Tabelle 3.18.: UC19: Kapazität der Fahrlehrer prüfen

UC20: Termin verschieben

UC20	**Termin verschieben**
Primary Actor	Mitarbeiter
Stakeholders and Interests	• Fahrschüler: Will einen neuen Termin erhalten
Preconditions	• Mitarbeiter ist angemeldet und besitzt die entsprechenden Berechtigungen • Es ist ein Termin vorhanden
Postconditions	• Alter Termin ist gelöscht • Neuer Termin ist vereinbart
Basic Flow	1. Bestehenden Termin löschen → UC16: Termin löschen 2. Neuen Termin vereinbaren → UC17: Termin erstellen
Alternative Flows	
Frequency of Occurrence	Täglich

Tabelle 3.19.: UC20: Termin verschieben

UC21: Fahrlektion durchführen

UC21	**Fahrlektion durchführen**
Primary Actor	Fahrlehrer
Stakeholders and Interests	• Fahrlehrer: – Möchte eine Fahrlektion durchführen – Möchte wissen, wo der Fahrschüler im Bezug auf die Lerninhalte steht • Fahrschüler: Möchte eine Fahrlektion durchführen • Administrator: Möchte aufgrund der Anzahl durchgeführter Fahrlektionen Rechnungen ausstellen können
Preconditions	• Fahrlehrer ist angemeldet und besitzt die entsprechenden Berechtigungen • Es ist ein Termin vereinbart worden
Postconditions	• Fahrlektion ist durchgeführt worden

Basic Flow

1. Fahrschüler und Fahrlehrer treffen sich gemäss vorher abgemachtem Termin
2. Fahrlehrer ruft Fahrschülerdaten ab
3. Fahrlehrer fragt bereits abgeschlossene Lerninhalte beim System ab, um über die Kenntnisse des Fahrschülers Bescheid zu wissen
4. Fahrlehrer fragt nächsten zu behandelnden Lerninhalt beim System ab
5. Lerninhalt wird behandelt
6. Fahrlehrer erfasst behandelten Lerninhalt im System → UC22: Lernfortschritt erfassen

   Schritte 3 und 5 werden solange wiederholt, bis die Fahrlektion zu Ende ist
7. Am Ende der Fahrstunde wird ein neuer Termin vereinbart → UC17: Termin erstellen

---

Alternative Flows

2.a Es handelt sich um die erste Fahrstunde und der Fahrschüler ist noch nicht im System erfasst: Fahrlehrer erfasst Fahrschüler im System → UC12: Fahrschüler verwalten , weiter mit Punkt 4

3.a Fahrschüler hat noch keine abgeschlossenen Lerninhalte: Weiter mit Punkt 4

7.a Fahrschüler besucht an demselben Tag die Fahrprüfung und besteht diese: Abbruch

7.b Fahrschüler tritt aus Fahrschule aus: Abbruch

Frequency of Occurrence Mehrmals täglich

Tabelle 3.20.: UC21: Fahrlektion durchführen

UC22: Lernfortschritt erfassen

UC22	**Lernfortschritt erfassen**
Primary Actor	Fahrlehrer
Stakeholders and Interests	• Fahrlehrer: Möchte jeweils einsehen können, wo ein Fahrschüler im Bezug auf die Lerninhalte steht • Fahrschüler: Möchte Fortschritte bei der Ausbildung machen, ohne bereits erlernte Manöver zu oft wiederholen zu müssen
Preconditions	• Fahrlehrer ist angemeldet und besitzt die entsprechenden Berechtigungen • Fahrlektion hat begonnen • Lerninhalt ist behandelt worden • Fahrlehrer hat Schülerinformationen aufgerufen
Postconditions	• Behandelter Lerninhalt ist in System gespeichert

Basic Flow

1. Fahrlehrer wählt behandelten Lerninhalt aus Lerninhalt-Liste
2. Fahrlehrer markiert Lerninhalt als „behandelt“

Alternative Flows

1.a Lerninhalt ist noch nicht in System vorhanden: Fahrlehrer erfasst Lerninhalt → UC11: Lerninhalt verwalten

2.a Lerninhalt gilt als „abgeschlossen“(muss in keiner weiteren Fahrlektion mehr behandelt werden): Lerninhalt wird als „abgeschlossen“markiert

2.b Fahrlehrer möchte Notizen zu behandeltem Lerninhalt verfassen: Fahrlehrer erfasst Notizen

2.c Fahrlehrer möchte Bewertung zu behandeltem Lerninhalt abgeben: Fahrlehrer erfasst Bewertung

Frequency of Occurrence	Mehrmals pro Fahrlektion

Tabelle 3.21.: UC22: Lernfortschritt erfassen

UC23: iCal Kalender abfragen

UC23	**iCal Kalender abfragen**
Primary Actor	Mitarbeiter
Stakeholders and Interests	• Mitarbeiter: Möchte Termine einer Ressource ausserhalb des Tools (zum Beispiel auf iPhone) anschauen können
Preconditions	• Mitarbeiter ist berechtigt, auf Kalender zuzugreifen • Ressource, für die Termine abgerufen werden, ist im System vorhanden
Postconditions	
Basic Flow	1. Mitarbeiter öffnet Kalender 2. Kalender ruft Schnittstelle auf System auf 3. System liefert Termine für gewählte Ressource aus
Alternative Flows	
Frequency of Occurrence	Täglich

Tabelle 3.22.: UC23: iCal Kalender abfragen

UC24: Statistik abfragen

UC24	**Statistik abfragen**
Primary Actor	Administrator
Stakeholders and Interests	• Administrator: Möchte Informationen zum Fahrbetrieb in Form von Statistiken einsehen können
Preconditions	• Administrator ist angemeldet und besitzt die entsprechenden Berechtigungen • Es sind Daten zum Auswerten in System abgelegt
Postconditions	
Basic Flow	1. Administrator wählt „Statistiken anzeigen“ 2. Administrator wählt gewünschte Statistik 3. System zeigt Statistik an
Alternative Flows	
Frequency of Occurrence	Monatlich

Tabelle 3.23.: UC24: Statistik abfragen

UC25: Rechnung erstellen

UC25	**Rechnung erstellen**
Primary Actor	Administrator
Stakeholders and Interests	• Administrator: Möchte durchgeführte Fahrlektionen verrechnen können
Preconditions	• Administrator ist angemeldet und besitzt die entsprechenden Berechtigungen • Es sind Fahrschülerdaten im System abgelegt • Es sind Fahrlektionsdaten im System abgelegt • Es sind Fahrlektionspreise im System abgelegt
Postconditions	• Fahrlektionen, für die eine Rechnung erstellt worden ist, sind als „verrechnet“ im System gekennzeichnet

Basic Flow

1. Administrator wählt „Rechnung erstellen“
2. Administrator wählt Fahrschüler, für den die Rechnung erstellt werden soll, aus Fahrschüler-Liste
3. System stellt alle noch nicht verrechneten Fahrlektionen dar und selektiert diese für die Verrechnung
4. System berechnet das Total des Rechnungbetrags aufgrund der Anzahl selektierten Fahrlektionen und dem Stundenpreis für die Fahrkategorie sowie allfälligen weiteren Dienstleistungen
5. Administrator druckt Rechnung aus
6. System bucht selektierte Fahrlektionen als „verrechnet“ab

Alternative Flows

3.a Administrator verrechnet nicht alle unbezahlten Fahrlektionen: Administrator deselektiert Fahrlektionen, die nicht verrechnet werden sollen

3.b Dem Fahrschüler sollen nebst den Fahrlektionen noch weitere Dienstleistungen (Theoriekurs, Verkehrskundeunterricht, Versicherung, etc.) in Rechnung gestellt werden: Administrator selektiert gewünschte weitere Dienstleistungen

4.a Administrator gewährt Fahrschüler einen Rabatt in Prozent auf den Totalbetrag:

a) Administrator gibt Rabatt in Prozentzahl ein

b) System berechnet neuen Totalbetrag abzüglich gewährtem Rabatt

4.b Administrator gewährt Fahrschüler einen absoluten Rabatt:

a) Administrator gibt Höhe des Rabatts ein

b) System berechnet neuen Totalbetrag abzüglich gewährtem Rabatt

Frequency of Occurrence	Wöchentlich

Tabelle 3.24.: UC25: Rechnung erstellen

UC26: Rechnung als PDF generieren

Analog zu „UC25: Rechnung erstellen“, mit Ausnahme Schritte 5 und 6 des Basic Flow:

UC26	**Rechnung als PDF generieren**
Extends UC	UC25: Rechnung erstellen

Basic Flow

5. Administrator wählt „Rechnung als PDF generieren“
6. System generiert PDF-Version der Rechnung und gibt diese an Administrator zurück
7. System bucht selektierte Fahrlektionen als „verrechnet“ab

Tabelle 3.25.: UC26: Rechnung als PDF generieren

3.2. Nichtfunktionale Anforderungen

3.2.1. Leistungsanforderungen

Das Abfragen von möglichen Termindaten bei der Terminfindung soll maximal 5 Sekunden dauern.

Über die Durchsatzraten können keine Angaben gemacht werden, da diese von der Internetverbindung des Anwenders abhängig sind.

3.2.2. Mengenanforderungen

Es sollen 10 gleichzeitige Benutzer (Fahrlehrer oder Administratoren) vom System behandelt werden können. Das System ist in der Lage, 50 Fahrzeuge zu verwalten. Weiter soll es möglich sein, 1000 Fahrschüler anzulegen.

3.2.3. Qualitätsmerkmale

Die Qualitätsmerkmale werden nach ISO 9126 [ISO01] erstellt und kategorisiert.

Funktionalität

Angemessenheit

Eine ressourcenschonende Kommunikation zwischen dem Server und dem Client ist sehr wichtig, da die Clients meistens über eine Mobilfunkverbindung auf das System zugreifen. Die zu übertragenden Daten werden deshalb komprimiert. Es wird zudem darauf geachtet, dass eingesetzte Libraries auf dem Client-System für die mobile Nutzung ausgelegt sind.

Richtigkeit

Bei einer Terminabfrage sollen die zurückgelieferten Daten zu 100% den aktuell verfügbaren Terminen entsprechen.

Interoperabilität

Die Client-Software soll auf gängigen Smartphones, Tablets und Desktop-Computern im Browser lauffähig sein. Die Software wird so konzipiert, dass die Darstellung auf die jeweiligen Bildschirmgrössen angepasst wird (Responsive Design).

Die Rechnungen werden in einem gängigen Dateiformat ausgegeben, dass sowohl auf Mobilgeräten, als auch auf Desktop-Computern gelesen werden kann. Hierbei handelt es sich um ein optionales Feature, womit diese Anforderung bei Nichtimplementation obsolet wird.

Ordnungsmässigkeit

Bei der Planung der Fahrlektionen werden die gesetzlichen Bestimmungen [Eid58] im Zusammenhang mit der maximalen Einsatzzeit eines Fahrlehrers pro Tag eingehalten. Es wird geprüft, dass der Fahrlehrer täglich elf Stunden Ruhezeit einhält.

Diese Anforderung entspricht dem optionalen Feature, das in „UC19: Kapazität der Fahrlehrer prüfen“ genauer beschrieben wird.

Fahrlektionen an Sonntagen sind überdies gesetzlich verboten, weshalb der Sonntag bei der Terminfindung jeweils wegfällt.

Sicherheit

Durch die Verwendung der Software sollen keine Sicherheitsdefizite entstehen.

Der Zugriff auf das System ist über einen Login abgesichert. Es erhalten nur Personen mit Benutzeraccount Zugriff auf das System. Die Funktionalitäten der Software lassen sich durch Berechtigungen absichern.
Bei der Umsetzung des Projekts wird kein spezieller Augenmerk auf die Sicherheit gelegt, weshalb auch keine Tests zum Beispiel mittels Security Audit vorgesehen sind.

Zuverlässigkeit

Reife

Bei einer funktionierenden Anbindung an das Internet soll in 90% der Fälle eine korrekte Kommunikation mit dem Server stattfinden.

Fehlertoleranz

Terminkollisionen werden erkannt und der Benutzer wird entsprechend informiert. Es wird dem Benutzer die Möglichkeit geboten, infolge einer Kollision einen neuen Termin zu suchen.

Benutzbarkeit

Verständlichkeit

Das *Graphical User Interface (GUI)* der Webapplikation soll intuitiv zu bedienen sein. Dabei sollen bestehende moderne Konzepte adaptiert werden. Um die Verständlichkeit frühzeitig testen zu können werden erstellte Wireframes mittels Paper-Prototyping getestet.

Erlernbarkeit

Es soll für den Fahrlehrer möglich sein, nach dem Login mit maximal sechs Klicks einen neuen Termin mit einem Fahrschüler zu vereinbaren.

Bedienbarkeit

Es soll möglich sein die App nur mittels Touch Bedienung und Gestures zu bedienen. Der Aufwand um die App bedienen zu können, soll möglichst gering gehalten werden.
Es werden hierbei im Rahmen des Projekts keine automatisierten UI-Tests eingesetzt.

Attraktivität

Das visuelle Auftreten der Clientsoftware soll möglichst ansprechend und schlicht gehalten werden. Das einheitliche und durchdachte Design sollte beim Benutzer einen professionellen Eindruck hinterlassen.
Die Verständlich der Design-Konzepte werden mittel Paper-Protoyping-Tests eruiert.
Es werden hierbei im Rahmen des Projekts keine automatisierten UI-Tests eingesetzt.

Effizienz

Zeitverhalten

Das Zeitverhalten der App ist abhängig von der Stabilität und Geschwindigkeit der Verbindung. Dies ist je nach Standort unterschiedlich, daher ist momentan keine genaue Aussage darüber möglich. Die serverseitige Architektur soll so ausgelegt werden, dass die Ressourcen den Anforderungen entsprechend angepasst werden können. Tests mit dem App-Prototypen werden mehr Aufschluss über das Zeitverhalten geben.

Die maximale Antwortzeit der Applikation bei einer gängigen Internetverbindung (5Mbps) soll 1 Sekunde betragen (Terminlogik ausgeschlossen).

Verbrauchsverhalten

Da der Webview unter iOS maximal 10MB für die Ausführung von JavaScript zur Verfügung stehen, muss der Clientteil so entwickelt werden, dass diese Limite nicht überschritten wird. Weiter müssen die entsprechenden Scripts in maximal 10 Sekunden abgehandelt werden. Resultierend kann gesagt werden, dass der Client-Teil so entwickelt werden muss, dass er sehr ressourcensparend operiert.
Die Auslastung der einzelnen Geräte wird bei der Entwicklung fortlaufend mittels entsprechender Debug-Tools überwacht. Es werden jedoch keine automatisierte Performance-Tests vorgenommen.

Wartbarkeit

Analysierbarkeit

Durch Fehlerlogging auf Serverseite, können Ursachen von Versagen schnell diagnostiziert werden.

Modifizierbarkeit

Änderungen am Serveralgorithmus und der Clientlogik müssen innerhalb von einem Manntag implementiert werden können. Andere Änderungen, die nicht das Kernsystem betreffen, sollten innerhalb von zwei Manntagen implementiert werden können.
Diese Anforderung stellt eine Produktanforderung dar und wird somit nicht im Rahmen dieses Projekts behandelt.

Stabilität

Änderungen an den Kernsystemen sollen zu 95% keine unerwarteten Fehler produzieren, andere Änderungen sollen zu 85% keine unerwarteten Fehler produzieren. Diese Anforderung stellt eine Produktanforderung dar und wird somit nicht im Rahmen dieses Projekts behandelt.

Testbarkeit

Die serverseitige Implementation der Software kann einfach getestet werden, da nach MVC entwickelt wird. Die Frontend-Implementation kann nicht ohne einen grossen Mehraufwand getestet werden (sprich UI-Tests). Einzelne Komponenten können jedoch ebenfalls in diesen Systemen mittels Unit Tests abgedeckt werden.

Übertragbarkeit

Anpassbarkeit

Die Applikation kann auf jeglichen Systemen mit installiertem Browser eingesetzt werden.

Installierbarkeit

Da es sich bei der Applikation um eine Web App handelt, ist keine Installation einer Software notwendig. Für den Betrieb der Software ist einzig ein installierter, aktueller Browser notwendig. Auf gängigen Desktop-Computern, Smartphones und Tablets ist ein solcher Browser bereits installiert.

Koexistenz

Die Koexistenz der Web App ist dadurch gewährleistet, als dass sie sich direkt im Browser aufrufen lässt und keine eigene Applikations-Instanz darstellt.

Austauschbarkeit

Die Software soll die heutige Microsoft Access Datenbank zur Verwaltung der Fahrschüler in naher Zukunft ersetzen. Hierzu werden die heutigen Datenstämme komplett in der neuen Applikation abgebildet. Eine allfällige Datenübernahme wird durch die Fahrschule selbst vorgenommen.

3.2.4. Randbedingungen / Abhängigkeiten

Server

Die Serversoftware soll auf einem *LAMP*-Stack aufsetzen. Diese Anforderung hat folgende Abhängigkeiten zur Folge:

- Betriebssystem: Linux
- Webserver: *Apache* 2.4
- Programmiersprache: *PHP* >= 5.3
- Datenbank: *MySQL* 5.5

Client

Vom Industriepartner wird vorgegeben, dass für den Betrieb der Web App das aktuelle „iPad Air“ mit installiertem Betriebssystem „iOS 7“ eingesetzt wird. Demzufolge wird die Web App in erster Linie für diese Geräte optimiert.
Auf Desktop-Computern muss zum Betrieb der Web App ein aktueller Browser installiert sein. Folgende Browser werden unterstützt:

- Google Chrome (ab Version 30)
- Mozilla Firefox (ab Version 24)
- Microsoft Internet Explorer (ab Version 9)
- Apple Safari (ab Version 7)

Weiter soll die Web App ebenfalls auf aktuellen iPhones (unter „iOS 7“) bedienbar sein.

Kapitel 4 **Domainanalyse**

4.1. Domainmodell

Das folgende Modell soll einen Überblick über die verschiedenen Objekte bei der Fahrschule geben. Im weiteren werden die einzelnen Bereiche „Fahrzeuge“, „Termine“, „Lerninhalte“ und „Verfügbarkeiten“ genauer aufgezeigt und erklärt.

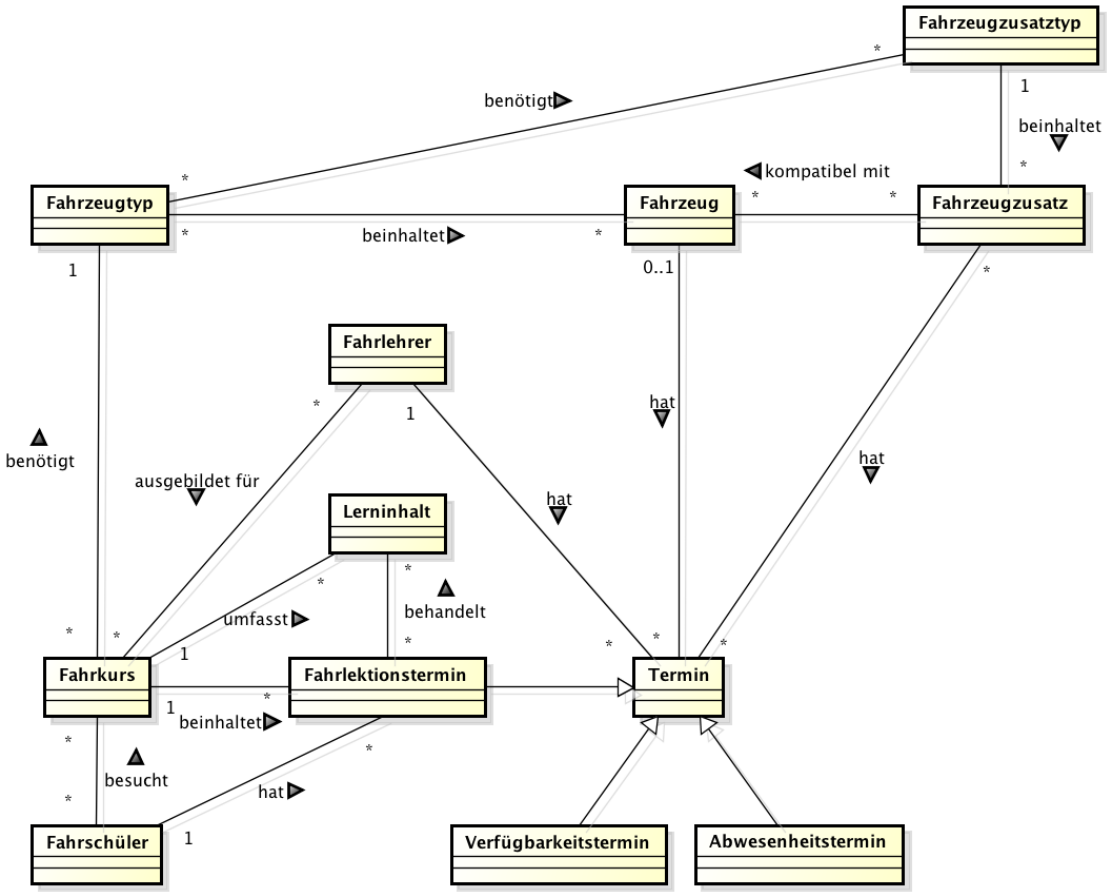


Abbildung 4.1.: Domainmodell: Überblick

4.1.1. Fahrzeuge

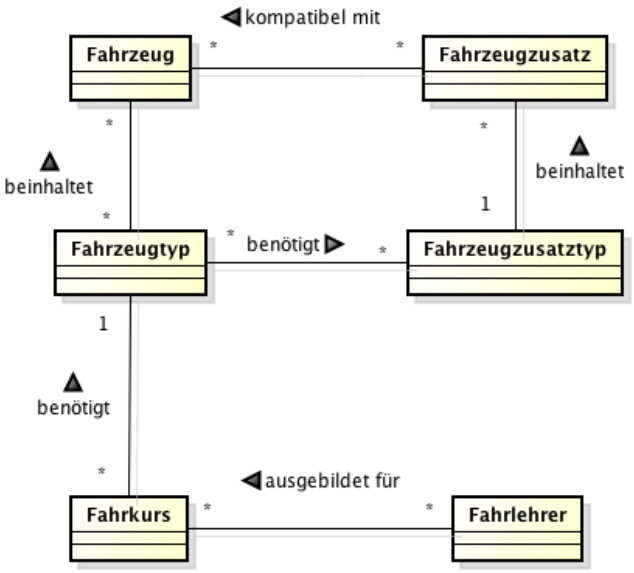


Abbildung 4.2.: Domainmodell: Fahrzeuge

Fahrlehrer

Beispiele: Peter Müller, Thomas Muster, Anna Schmid

Ein Fahrlehrer ist ein Angestellter der Fahrschule. Er ist befugt, verschiedene Fahrkurse durch zu führen und somit auch Termine dafür wahrzunehmen.

Fahrkurs

Beispiele: Kategorie B, BE, C, C1, CE, D

Fahrkurse sind definierte Ausbildungen, die durch das Schweizer Gesetz (Kategorienübersicht) und das Angebot der Fahrschule definiert werden. Die Fahrschule konzentriert sich vorwiegend auf Lastkraftwagenausbildungen (Kategorien C, C1, CE).

Ein Fahrkurs erfordert ebenfalls einen Fahrzeugtyp um den Fahrkurs zu bestreiten. Zudem sind pro Fahrkurs verschiedene Lerninhalte definiert.

Fahrzeuge

Beispiele: LKW Saurer Grün 1, VW Golf V GTI, Anhänger 3 - 1 Achse

Fahrzeuge sind mobile Verkehrsmittel, die zur Durchführung einer Fahrlektion notwendig sind.

Fahrzeugtyp

Beispiele: Lastkraftwagen, Lastkraftwagen mit Anhänger, Car, Car mit Anhänger, Auto mit Anhänger

Fahrzeugtypen bestehen aus einem oder mehreren Fahrzeugen. Diese ermöglichen es, Fahrzeuggespanne zu definieren; beispielsweise einen Lastkraftwagen mit Anhänger.

Fahrzeugzusatz

Beispiele: Anhänger Schwarz 1, Nummernschild ZH 123456

Fahrzeugzusätze sind Mittel, die für einen Fahrzeugtyp zusätzlich benötigt werden.

Fahrzeugzusatztyp

Beispiele: Anhänger Schwer, Anhänger Leicht, Nummernschild Car

Fahrzeugzusatztypen bestehen aus einem oder mehreren Fahrzeugzusätzen. Diese werden benötigt, um Abhängigkeiten abzubilden.

Beispiel Kategorie DE: Gesellschaftswagen mit Anhänger

Grafik eines realen Beispiels:

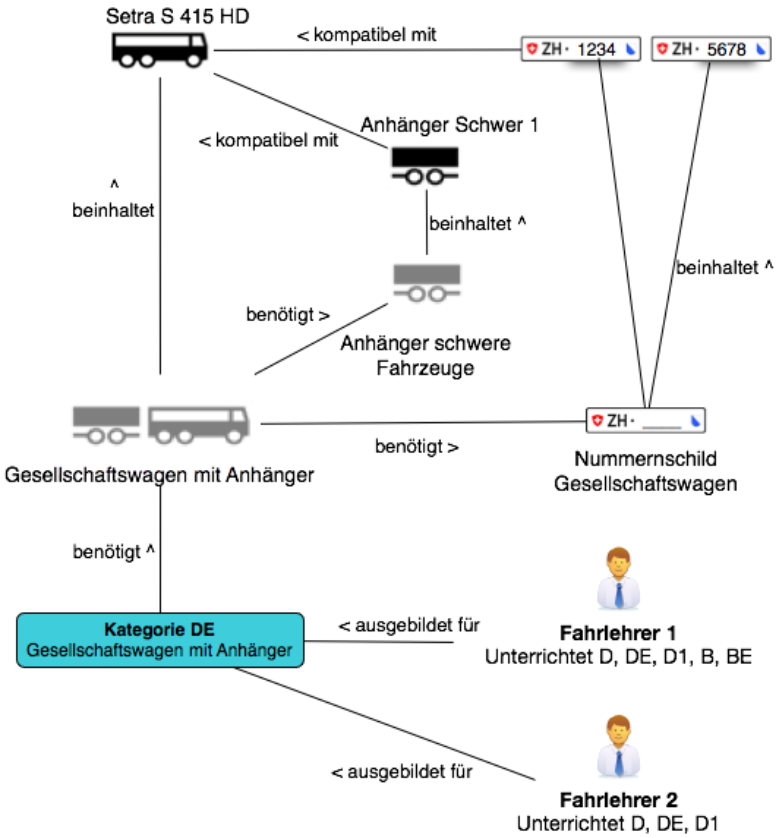


Abbildung 4.3.: Domainmodell: Beispiel anhand der Kategorie DE (Gesellschaftswagen mit Anhänger)

4.1.2. Fahrlektionstermine

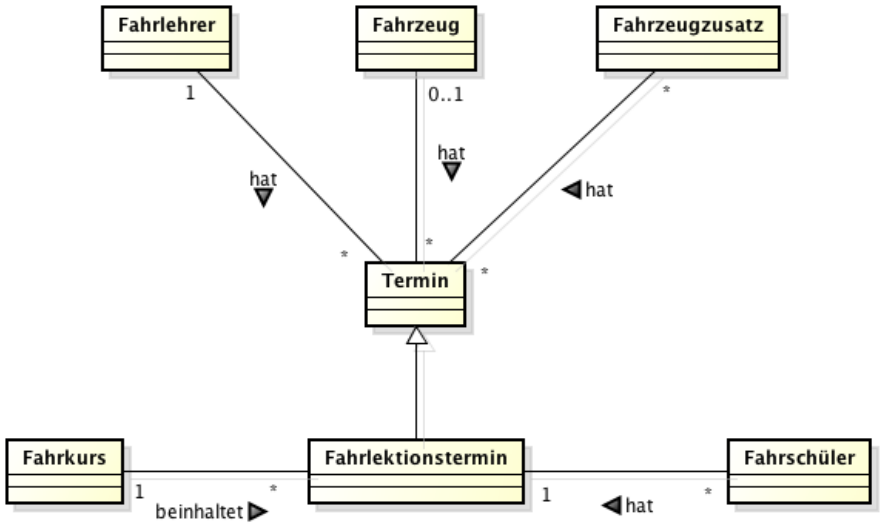


Abbildung 4.4.: Domainmodell: Fahrlektionstermine

Fahrschüler

Beispiele: Fritz Kobel, Cindy Schwarz, Hans Maier,
Ein Fahrschüler ist ein Kunde der Fahrschule. Er besucht normalerweise einen Fahrkurs (Bspw. Kategorie C), und wird dadurch mehrere Termine buchen.

Termin: Fahrlektionstermin

Beispiele: Montag 30. September 13:00 - 15.00, Peter Müller mit Cindy Schwarz, LKW Saurer Grün 1
Ein Termin definiert eine Zeitspanne, an welchem sich ein Fahrlehrer mit dem Fahrschüler trifft. Dieser benötigt im Normalfall ein Fahrzeug (+ Fahrzeugzusätze), jedoch kann der Fahrschüler diese auch selber zur Verfügung stellen. An einem Termin werden ein oder mehrere Lerninhalte behandelt.

4.1.3. Lerninhalte

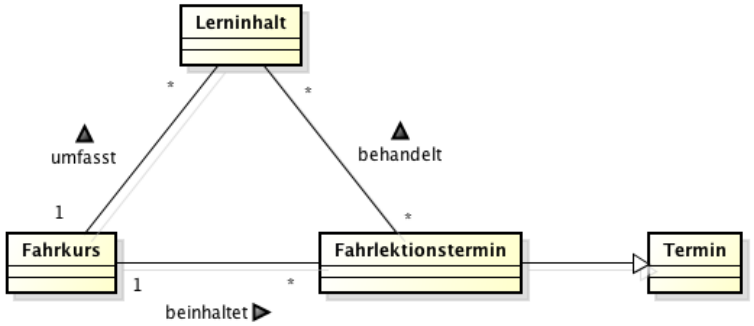


Abbildung 4.5.: Domainmodell: Lerninhalte

Lerninhalt

Beispiele: Kupplung Schleifpunkt, Parken, Autobahn
Lerninhalte definieren Elemente der Ausbildung für einen Fahrkurs. Sie dienen dem Fahrlehrer als Checkliste und werden an den verschiedenen Terminen behandelt. Somit kann der Fahrlehrer auch den Fortschritt eines Fahrschülers verfolgen und festhalten.

4.1.4. Verfügbarkeit

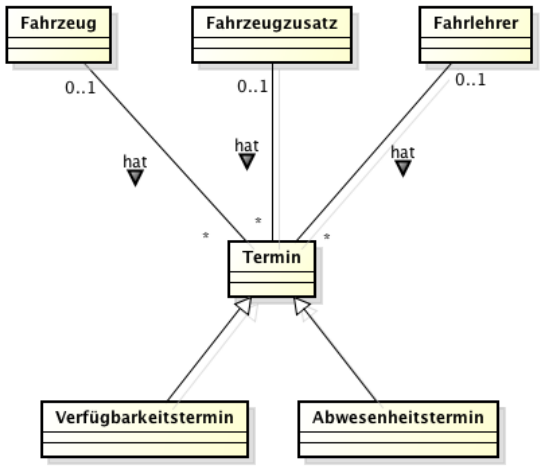


Abbildung 4.6.: Domainmodell: Verfügbarkeit

Termin: Verfügbarkeitstermin

Beispiele: Montag 30. September 9:00 - 17:00 - Peter Müller verfügbar
Für ein Fahrzeug, einen Fahrlehrer oder einen Fahrschüler können Verfügbarkeiten definiert werden. Dies zeigt, dass eine Ressource zu diesem Zeitpunkt für eine Fahrlektion gebucht werden kann.

Termin: Abwesenheitstermin

Beispiele: Montag 30. September 12:00 - 13:00 - Peter Müller abwesend, Dienstag 1. Oktober 8:00 - 13:00 - LKW Saurer Grün 1 Reparatur
Analog zu den Verfügbarkeiten können für Ressourcen Abwesenheiten definiert werden. Wenn ein Fahrlehrer grundsätzlich den ganzen Tag verfügbar ist, jedoch noch einen Termin hat, kann dieser mittels Abwesenheit übersteuert werden.

Kapitel 5 **Design**

5.1. Systemübersicht

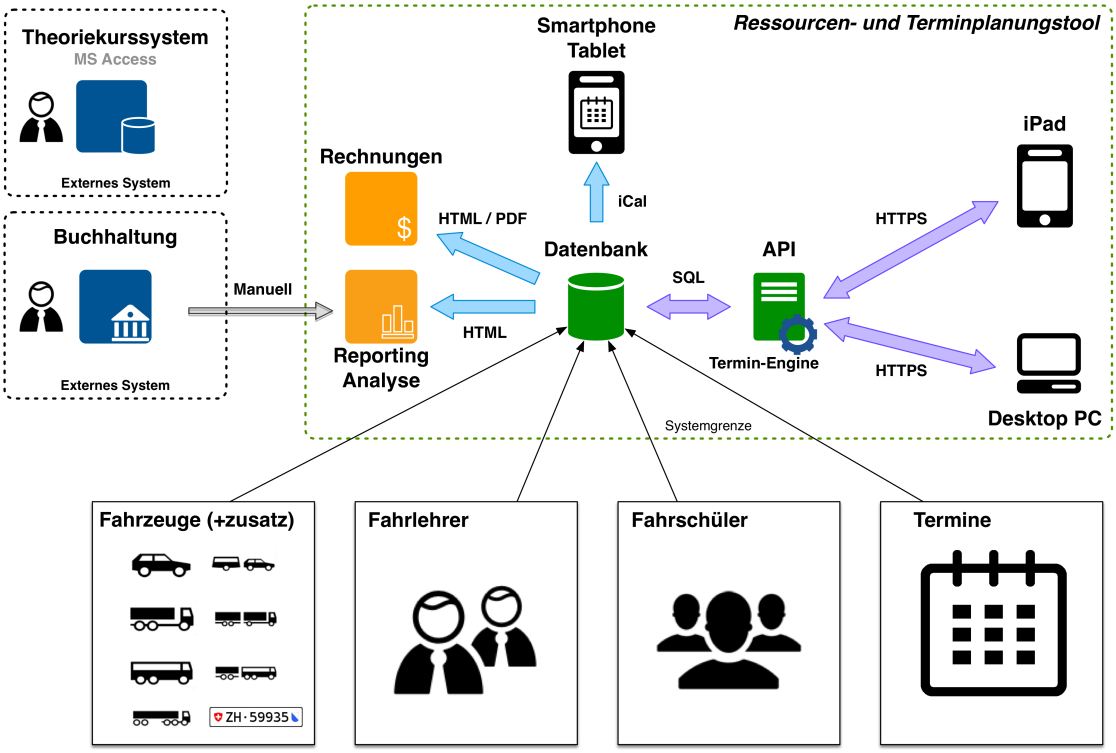


Abbildung 5.1.: Systemübersicht

Ressourcen- und Terminplanungstool

Das Ressourcen- und Terminplanungstool ist das zentrale Element der Fahrschule. Es dient der Verwaltung und Planung aller Ressourcen wie Fahrzeuge, Fahrzeugzusätze, Fahrlehrer und Fahrschüler.

Datenbank

Die Datenbank ist für die persistente Datenspeicherung zuständig. Sie be-

inhaltet unter anderem alle Ressourcen wie Fahrzeuge und deren Zusätze (wie Anhänger oder Nummern), Fahrlehrer und Fahrschüler sowie die verschiedenen Termine.

iCal-Schnittstelle

Die iCal-Schnittstelle erlaubt es dem Anwender, die Termine von ausgewählten Ressourcen (wie Fahrlehrer, Fahrzeuge oder Fahrzeugzusätze) direkt auf seinem Smartphone oder Desktop-Computer in einem kompatiblen Kalender darzustellen. Die Schnittstelle interagiert hierbei zeitnah, was heisst, dass jeder neu erfasste Termin nach einem Server-Abgleich sofort im nativen Kalender dargestellt wird.

API

Die API ermöglicht die Abfrage und Bearbeitung der Daten. Diese Abfragen werden via Mobile- oder Web-Client ausgelöst.

Reporting / Analyse

Um die Daten auszuwerten dient das Reporting und Analyse System, welches die Daten aus der Datenbank liest, diese entsprechend aggregiert, und anzeigt. Reporting / Analyse ist ein optionales Feature → siehe hierzu auch „Use Cases: Übersicht“

Rechnungen

Um Rechnungen erstellen zu können werden die entsprechenden Daten aufgrund abgeschlossener Fahrlektionen aus der Datenbank gewonnen und entsprechend zusammengestellt. Die Rechnungen können sowohl in Form einer Webseite (HTML), als auch als PDF-Dokument generiert werden.
Rechnungen sind ein optionales Feature → siehe hierzu auch „Use Cases: Übersicht“

Buchhaltung

Die Buchhaltung ist ein manueller Prozess, welcher auf das Reporting Tool angewiesen ist. Es benötigt die Daten pro Fahrschüler um entsprechende Berechnungen zu machen.

Theoriekurssystem

Es besteht eine Microsoft Access Applikation, die dem Verwalten der Fahrschüler und der Durchführung von Theoriekursen dient. Dieses System ist ebenfalls angebunden an die Webseite, wo sich Kursteilnehmer direkt anmelden können. Es ist komplett unabhängig vom Termin- und Ressourcenplanungstool und muss daher nicht weiter beachtet werden.

5.2. Technologieentscheidungen

5.2.1. Web Application Framework

Um die Wartbarkeit der Webapplikation sicherstellen zu können, wird ein Web Application Framework evaluiert. Gemäss den erhobenen nichtfunktionalen Anforderungen im Bezug auf die Abhängigkeiten (siehe 3.2.4 „Randbedingungen / Abhängigkeiten“) soll das Framework auf *PHP* basieren. Als weitere Anforderung wird aufgenommen, dass das Framework einen objektorientierten Ansatz zu verfolgen hat.

Symfony2

Der Entscheid für das eingesetzte Framework fällt auf *Symfony2* in der Version 2.3.

Beschreibung

Symfony2 ist ein Open Source Web Application Framework, das auf dem *Model-View-Controller (MVC)*-Konzept basiert. Es wird unter der Führung von Fabien Potencier entwickelt und ist 2011 in einer ersten Version erschienen. Es setzt klar auf *OOP* Prinzipien und hat durch die vielen, mitgelieferten Komponenten (Forms, Doctrine, Security etc.) eine ziemliche steile Lernkurve.
Die Dokumentation ist umfassend und die Community sehr gross und hilfsbereit. Zum jetzigen Zeitpunkt hat Symfony2 über 10’000 Commits von über 800 Contributors (Stand 18. Dezember 2013 gemäss GitHub[1]).

Features

- Datenbankzugriff über *ORM*-Layer (Doctrine 2)
- Scaffolding von CRUD-Interfaces
- Request-Dispatcher und dynamisches Routing für Clean URLs
- Templates auf Basis von Twig
- Datenvalidierung über Constraints
- Komponenten für Session-Verwaltung, Request-Handling und Security
- Umfassendes Caching der Ausgabe
- Pluginfähig mit Bundles

[1] Symfony2 Repository auf Github, siehe *https://github.com/symfony/symfony*

Alternativen

- **Codelgniter**[2] ist ein Framework, das die nötigen Grundlagen liefert um moderne Webapplikationen zu schreiben. Es hat eine gute Dokumentation und erlaubt einen schnellen Fortschritt. Codelgniter wird von der Firma *EllisLab* entwickelt.

- **Zend Framework 2**[3] ist ein Framework für performante und moderne PHP Applikationen. Es erlaubt die schnelle Entwicklung anspruchsvoller Webapplikationen. Zend Framework 2 wird durch das Unternehmen *Zend Technologies* vertrieben.

Begründung Entscheidung

Alle oben genannten Frameworks haben ein sehr ähnliches Feature Set und auch ihre Stärken und Schwächen. Der Entscheid fiel schlussendlich auf Symfony2, da es unsere Anforderungen erfüllt und das einzige Framework ist, mit welchem das Projektteam bereits Erfahrungen sammeln konnte.
Symfony2 ist ein sehr solides Produkt, ist weit verbreitet und hat einen guten Support, was die Entscheidung zudem bestätigt.

Doctrine 2

Doctrine 2 ist ein *Object Relational Mapper* (*ORM*) welcher standardmässig mit *Symfony2* ausgeliefert wird und Persistenz Funktionen zur Verfügung stellt. Es bietet transparent und einfache Handhabung von persistenten PHP Objekten erlaubt. Es baut auf *Database Abstraction Layer* (*DBAL*) auf.

Begründung Entscheidung

Da Doctrine 2 standardmässig mit Symfony2 mitgeliefert wird, werden hier keine weiteren Frameworks in Betracht gezogen.

[2] Codelgniter, siehe *http://ellislab.com/codeigniter*
[3] Zend Framework, siehe *http://framework.zend.com*

5.3. Externes Design der Webapplikation

5.3.1. Vorgehen

Um eine hohe Benutzerakzeptanz erreichen zu können, wird in Woche 6 ein User Testing mit den Anwendern durchgeführt. Hierfür werden Mockups ausgearbeitet, welche die Grundfunktionalitäten der Softwarelösung abbilden. Anlässlich des User Testings werden den Testbenutzern die Mockups vorgelegt um die Abläufe in Form der ausgearbeiteten Use Cases durchzugehen.

Allfällige Unklarheiten, Mängel oder Änderungswünsche, die von den Anwendern eingebracht werden, werden bei der Umsetzung des externen Designs der Webapplikation berücksichtigt und entsprechend implementiert.

5.3.2. Mockups

Die Mockups werden mit dem Tool *Balsamiq Mockups* erstellt. Um beim User Testing ein möglichst systemnahes Resultat zu erhalten sind alle erstellten Mockups interaktiv gestaltet und miteinander verlinkt. Dies ermöglicht es dem Benutzer, anhand einer Demo-Applikation auf dem iPad durch die Mockups zu navigieren.

Grundaufbau

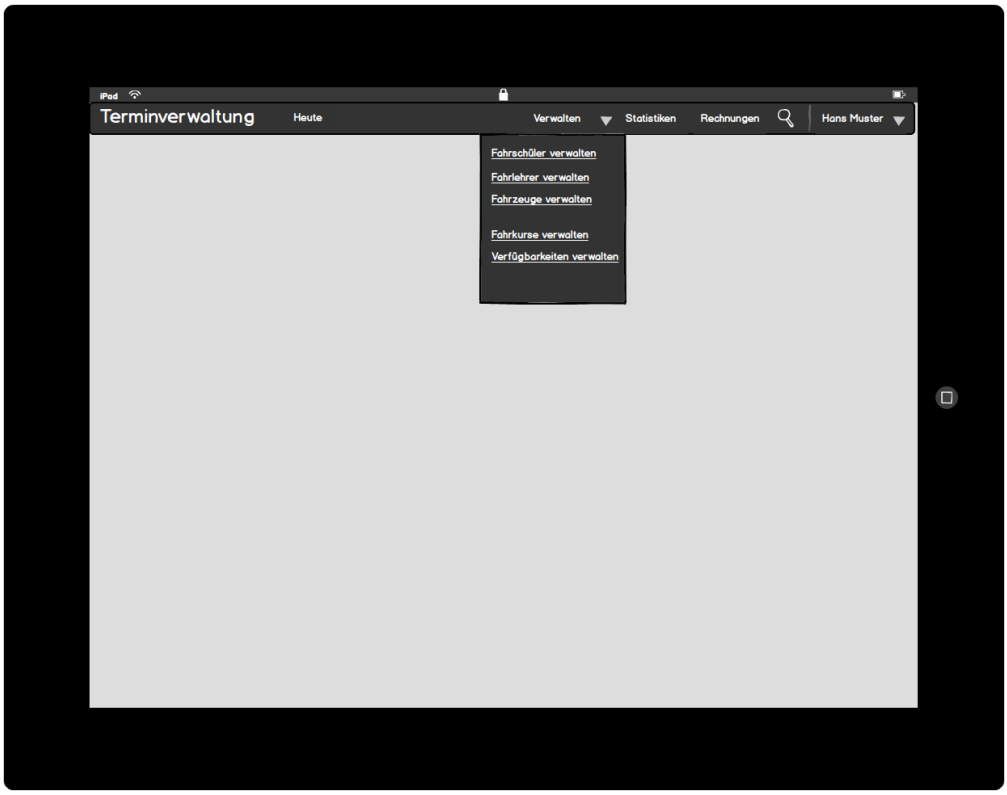


Abbildung 5.2.: Mockup: Grundaufbau

Die Mockups werden mit Hauptaugenmerk auf Tablets, konkret das iPad, konzipiert, da diese Geräte mehrheitlich in der Fahrschule zum Einsatz kommen. Es wird jedoch darauf geachtet, dass eine responsive Darstellung möglich ist, die jeweils abhängig vom Darstellungsmedium die bestmögliche Darstellungsart wiedergibt.

Die Navigation soll auf jeder Ansicht gleich aussehen und am oberen Rand fixiert werden. Sie verfügt über folgende Menüpunkte:

Heute
: Tagesübersicht mit nützlichen Funktionen → Mockup: Ansicht Heute

Verwalten
: Beinhaltet Untermenü zur Bearbeitung der einzelnen Stammdaten → Beispiel: Mockup: Übersicht Fahrschüler

Statistiken
: Gibt Statistiken zur Fahrschule wieder (optionales Feature → UC24: Statistik abfragen)

Rechnungen
: Ansicht zum Erstellen von Rechnungen für Fahrschüler (optionales Feature → UC25: Rechnung erstellen)

Hans Muster
: Name des aktuell angemeldeten Benutzers, beinhaltet Untermenü zum Bearbeiten der Benutzereinstellungen und -daten.

Ansicht Heute

Abbildung 5.3.: Mockup: Ansicht Heute

Die Ansicht „Heute“ ist in Form eines Dashboards konzipiert und soll dem Fahrlehrer einen Einstiegspunkt bieten. Hier hat er einen Überblick über die am heutigen Tag anfallenden Fahrlektionen sowie die von ihm eingetragenen Abwesenheiten. Ein Klick auf eine Fahrlektion öffnet die Detailansicht, wo der Lernfortschritt erfasst und neue Fahrlektionen für den entsprechenden Fahrschüler vereinbart werden können.

Fahrlehrer, die ebenfalls Fahrzeugverantwortliche sind, haben direkt Einsicht in anstehende Erinnerungen für Fahrzeuge oder Fahrzeugzusätze in Form von *Todos*.

Weiter sind die wichtigsten Funktionalitäten als Shortcuts direkt aufrufbar.

Beispielansicht Stammdaten - Übersicht

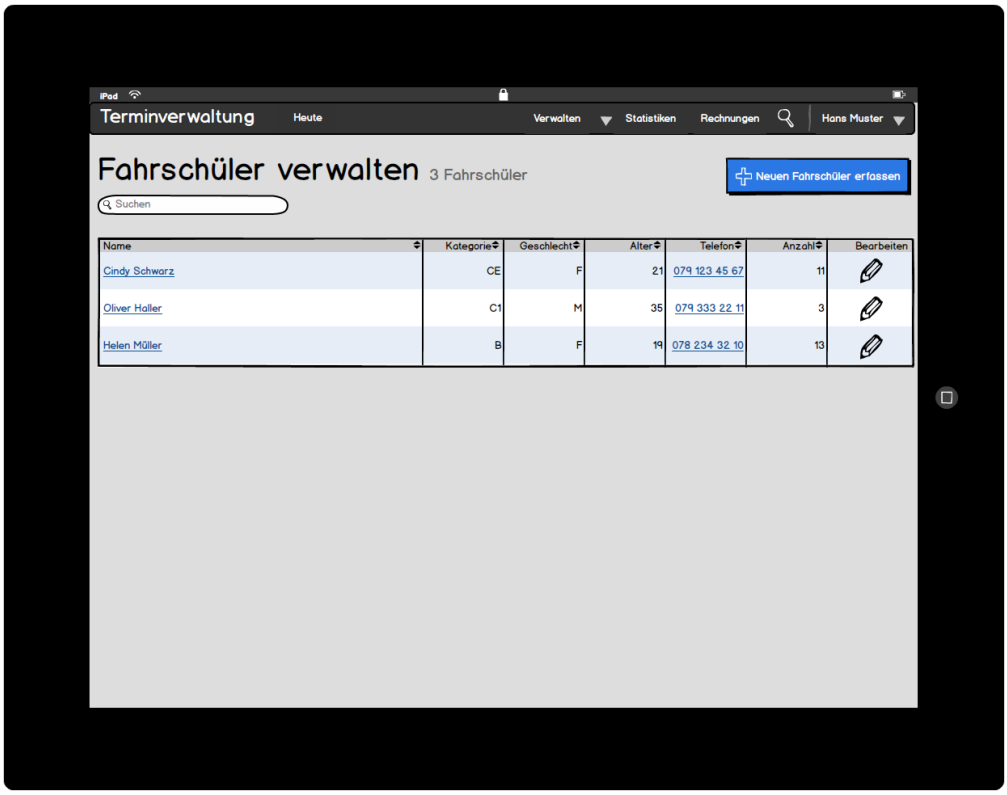


Abbildung 5.4.: Mockup: Übersicht Fahrschüler

Diese Ansicht gibt alle Objekte eines Stammdatentyps (zum Beispiel Fahrschüler) in Form einer Liste wieder. Dabei sollen die wichtigsten Eigenschaften direkt einsehbar sein. Mittels Klick auf einen Eintrag wird eine Detailansicht, die alle Eigenschaften darstellt, geöffnet.

Weiter besteht die Möglichkeit, einen neuen Datensatz anzulegen oder direkt die Ansicht zum Bearbeiten eines Objekts aufzurufen → Mockup: Fahrschüler bearbeiten

Beispielansicht Stammdaten-Eintrag bearbeiten

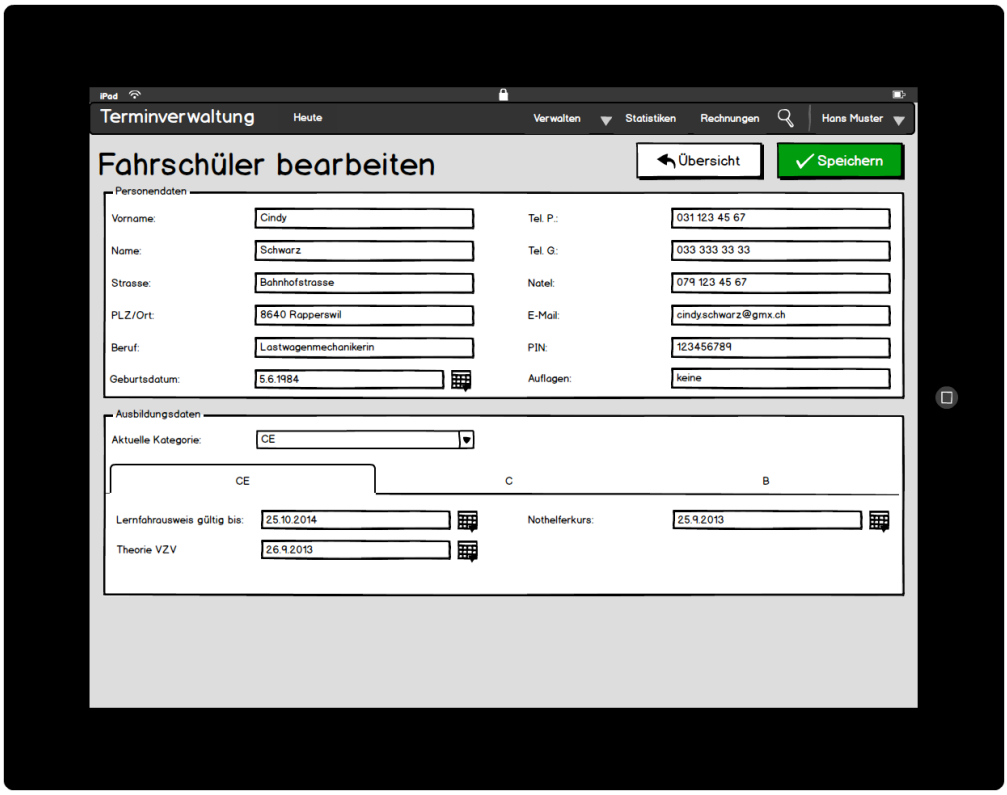


Abbildung 5.5.: Mockup: Fahrschüler bearbeiten

Die Eigenschaften und Relationen eines Objekts können mittels Eingabe in den einzelnen Feldern bearbeitet werden. Beim Speichern der Inhalte wird eine Validierung der Eingabewerte durchgeführt und allfällige Probleme wie Fehleingaben direkt dem Benutzer zurückgemeldet.

Termin finden

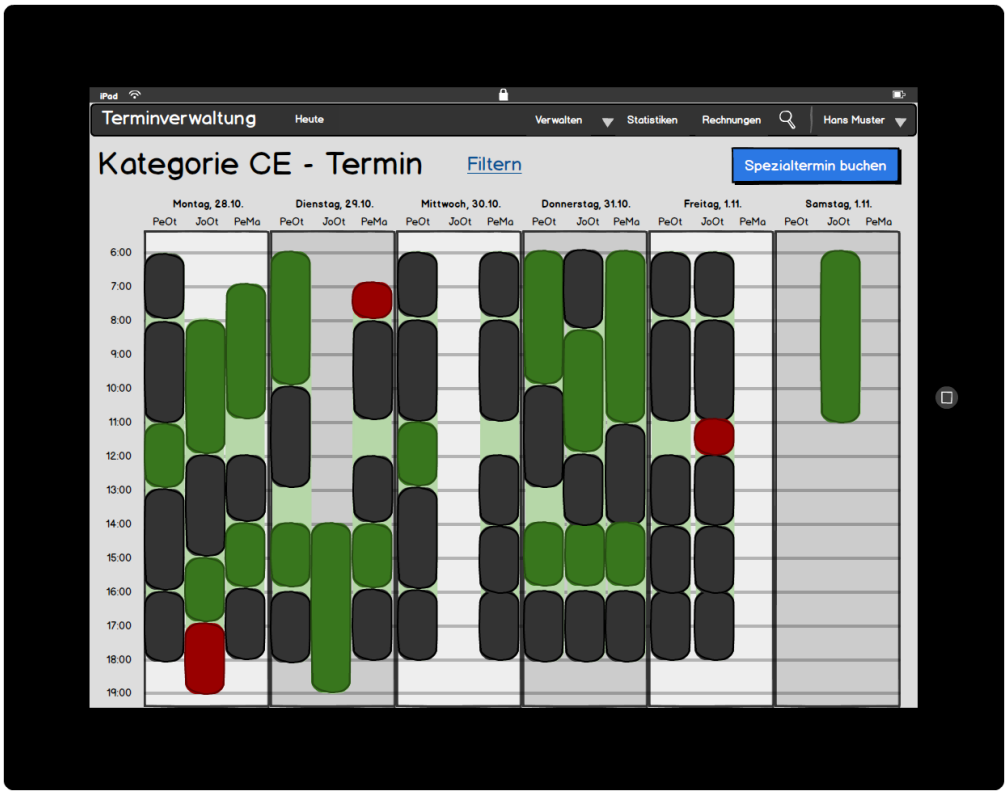


Abbildung 5.6.: Mockup: Termin finden

Bei der Konzipierung der Terminfindungsansicht wird bewusst auf eine dem Benutzer vertraute, intuitive Darstellung in Form eines Kalenders gesetzt. Dieser kann durch verschiedene Filteroptionen individuell angepasst werden. → Mockup: Termine filtern

Pro Tag werden in Form von Spalten jeweils die Verfügbarkeiten der Fahrlehrer für den gewählten Kurs angezeigt. Die Spalten werden mit den Namenskürzeln des jeweiligen Fahrlehrers versehen.

Der Anwender soll sich schnell einen Überblick über die bereits vereinbarten Termine verschaffen können. Solche Termine werden gemäss einer dem Fahrzeug zugewiesenen Farbe hervorgehoben. Wird ein solcher Zeitabschnitt angeklickt, erhält der Benutzer Einsicht in die Detailansicht mit weiteren Informationen zum Termin.

Abwesenheiten der Fahrlehrer oder Fahrzeuge werden rot markiert. Analog zu vereinbarten Terminen kann eine Detailsicht geöffnet werden.

Zeitabschnitte, in welchen eine neue Fahrlektion gebucht werden kann, werden in der Farbe grün hervorgehoben. Wählt der Anwender einen solchen Abschnitt an, öffnet sich ein Dialog zur Vereinbarung einer neuen Fahrlektion für den entsprechenden Fahrschüler → Mockup: Termin buchen

Um einen bestehenden Termin überbuchen zu können, wird der Button „Spezialtermin buchen“ vorgesehen. Dieser erlaubt es dem Benutzer, trotz bestehender Termine oder Abwesenheiten neue Fahrlektionen zu diesem Zeitpunkt zu vereinbaren. Der Benutzer zeigt sich in diesem Fall selbst dafür verantwortlich, allfällige Terminkollisionen aufzulösen.

Termine filtern

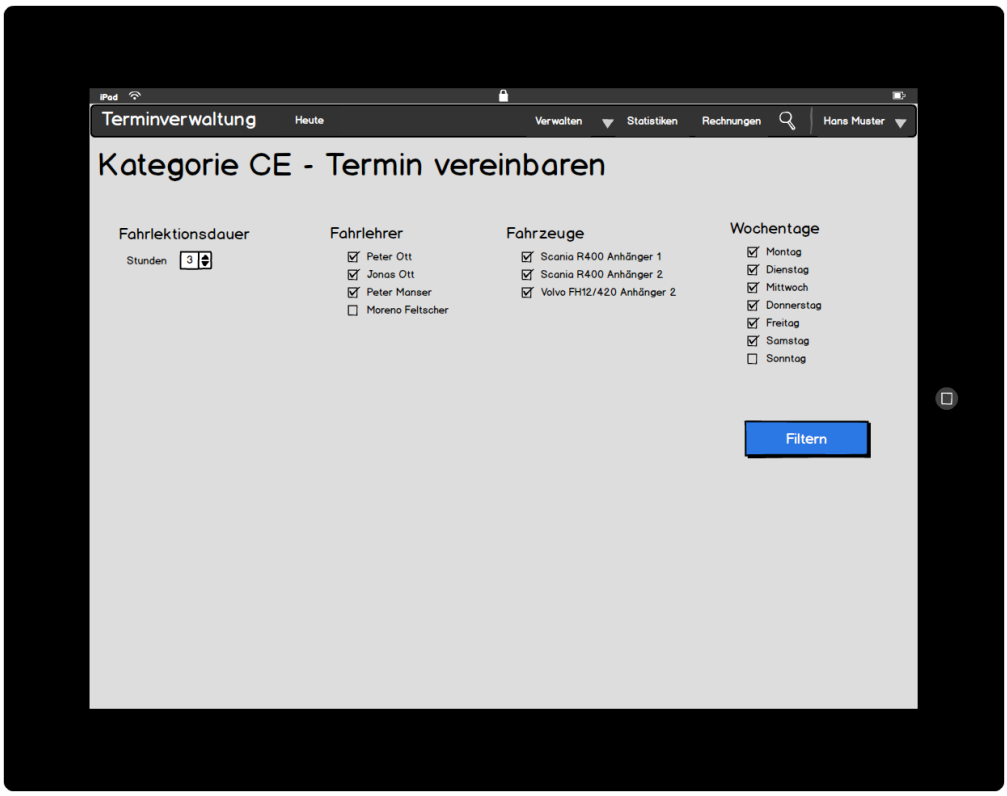


Abbildung 5.7.: Mockup: Termine filtern

Es können verschiedene Filter für die Terminfindung gesetzt werden. Die Filter basieren jeweils auf der vom Fahrschüler besuchten Fahrkategorie.

Termin buchen

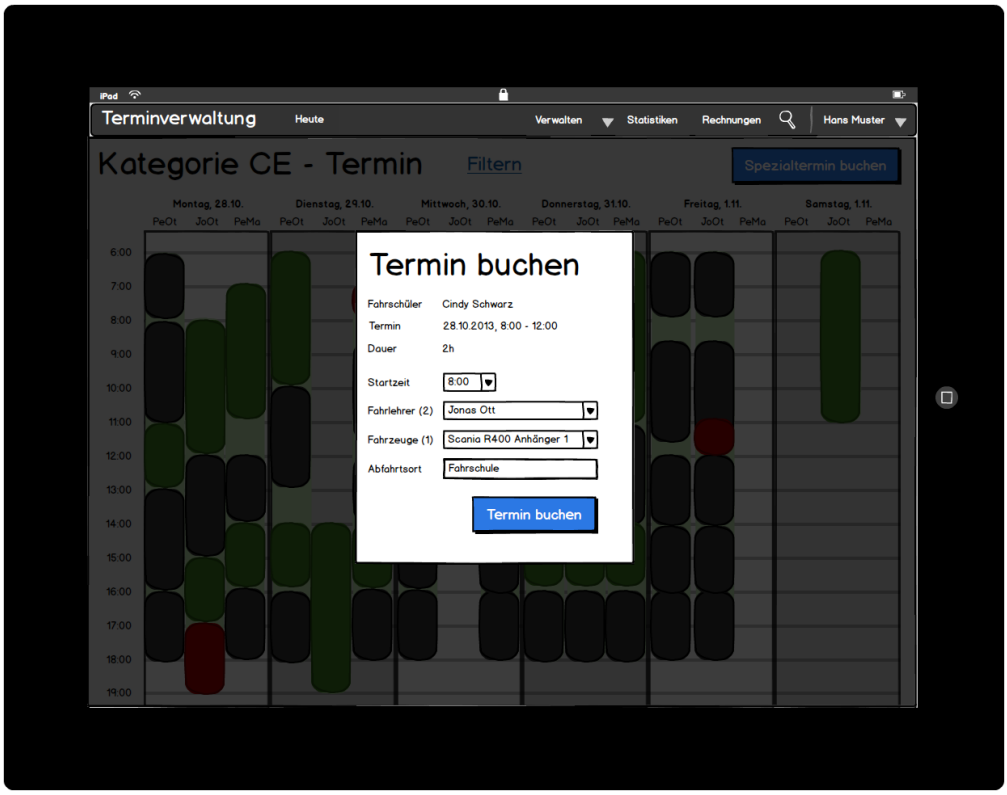


Abbildung 5.8.: Mockup: Termin buchen

Wird ein geeigneter Zeitpunkt für eine neue Fahrlektion gefunden, öffnet sich ein Dialog, der alle relevanten Informationen zur neuen Fahrlektion beinhaltet.

Sollte es mehrere Optionen für die Zuteilung der Ressourcen (Fahrlehrer, Fahrzeug, Fahrzeugzusatz) geben, werden diese entsprechend dargestellt und können durch den Fahrlehrer in Absprache mit dem Fahrschüler angewählt werden.

Mittels Klick auf den Button „Termin buchen“ wird der Termin endgültig gebucht und steht anderen Benutzern nicht mehr zur Verfügung.

Fahrlektion durchführen und Lernfortschritt erfassen

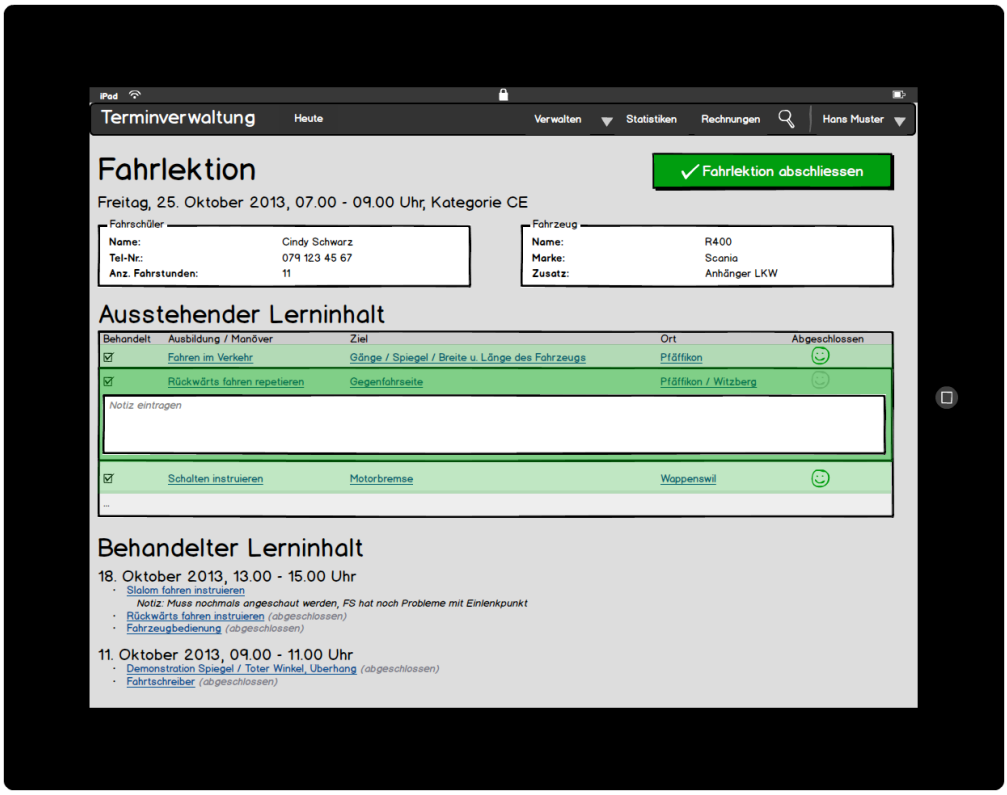


Abbildung 5.9.: Mockup: Fahrlektion durchführen und Lernfortschritt erfassen

Diese Ansicht widerspiegelt das optionale Feature „UC22: Lernfortschritt erfassen“.

Dem Fahrlehrer wird eine Übersicht über die durchzuführende Fahrlektion dargestellt. Diese enthält sowohl eine Zusammenfassung der Fahrschüler- und Fahrzeugdaten, als auch Informationen zu behandelten Lerninhalten.

Während der Fahrlektion kann der unterrichtende Fahrlehrer den Fortschritt erfassen. Hierzu wählt er aus einer Liste aller für diesen Fahrkurs erfassten Lerninhalte ein Objekt aus. Er kann nun Bewertungen dazu abgeben oder Notizen erfassen.

Nach Abschluss der Fahrlektion bestätigt der Benutzer seine Eingaben mit einem Klick auf den Button „Fahrlektion abschliessen“.

5.3.3. Gestaltung des Endprodukts

Aufgrund der im User Testing gewonnenen Erkenntnisse wird die Webapplikation gestaltet.

Der Fokus bei der Entwicklung des Frontends lag zusätzlich auf einer responsiven Darstellung, die es dem Benutzer ermöglicht, die Applikation auf verschiedenen Anzeigegeräten bedienen zu können.

Das Endprodukt kann sowohl auf Desktop-Computern als auch auf Tablets problemlos bedient werden. Die Einsicht von Daten mit einem Smartphone ist ebenfalls möglich, auch wenn dies keine Produkteanforderung darstellt. Das Abrufen von Terminen einer Ressource geschieht über die iCal-Schnittstelle und kann auf jedem kompatiblen Gerät, wie beispielsweise Smartphones, eingerichtet werden.

Folgende Illustration widerspiegelt genannte Produktespezifikationen:

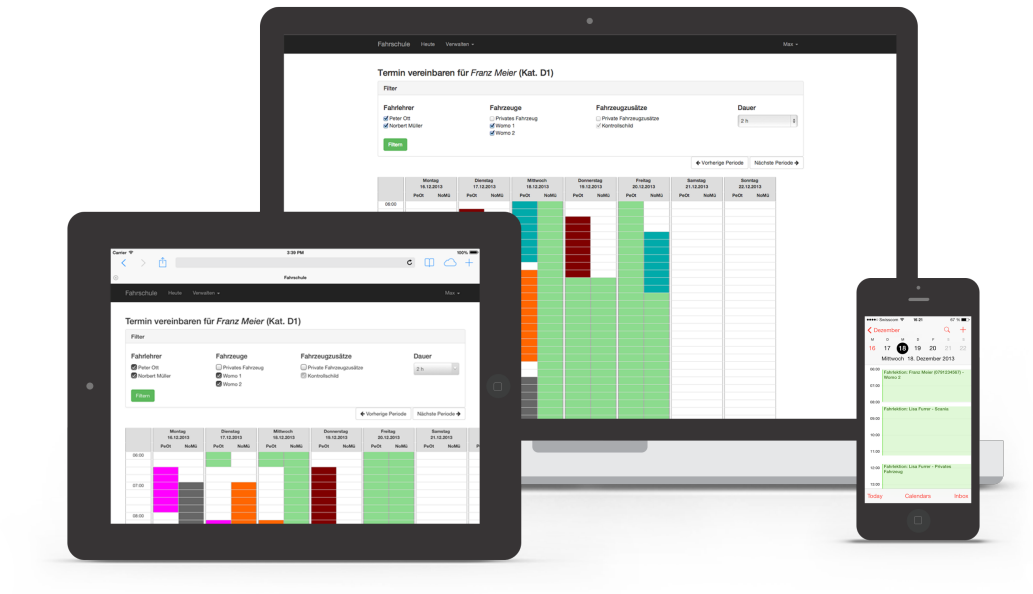


Abbildung 5.10.: Produkteübersicht

Dashboard

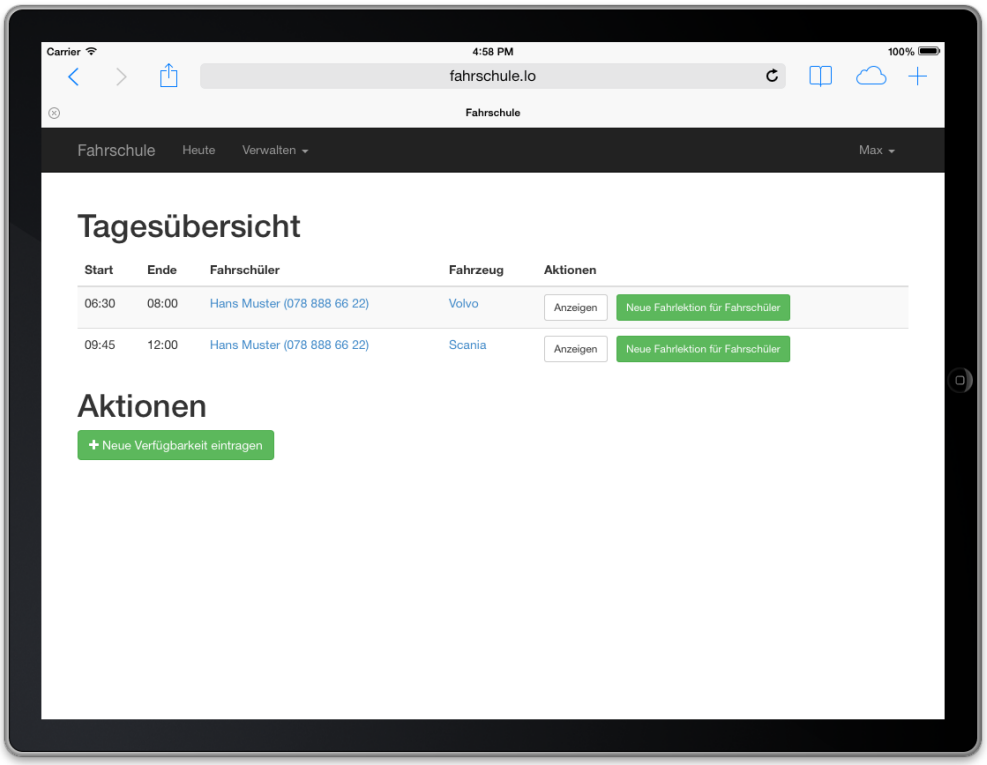


Abbildung 5.11.: Screenshot: Dashboard

Das Dashboard zeigt dem Fahrlehrer seine Termine für den heutigen Tag an und stellt häufig gebrauchte Funktionalitäten zur Verfügung. Für jede dargestellte Fahrlektion kann direkt ein neuer Termin vereinbart werden. Dies ist dann hilfreich, wenn ein Fahrlehrer zusammen mit einem Fahrschüler nach einer abgeschlossenen Fahrlektion gleich einen neuen Termin vereinbaren möchte.

Stammdaten anzeigen

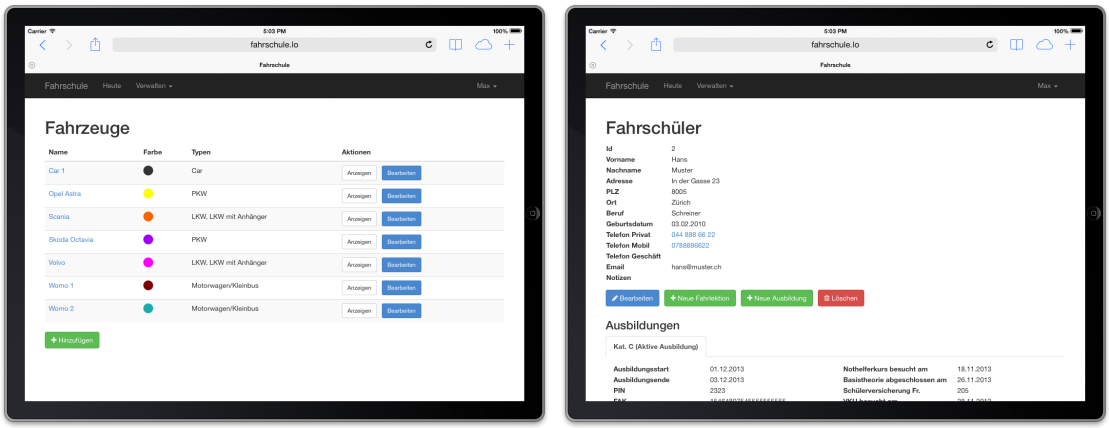


Abbildung 5.12.: Screenshots: Stammdaten anzeigen

Zu jedem Stammdaten-Typ gibt es sowohl eine Übersichts- als auch eine Detailansicht. Die Übersichtsseite listet alle erfasst Daten auf und gibt die wichtigsten Eigenschaften wieder. Möchte man ausführlichere Informationen einsehen, so kann dies über den Button „Anzeigen“, der die Detailansicht öffnet, erreicht werden.

Stammdaten bearbeiten

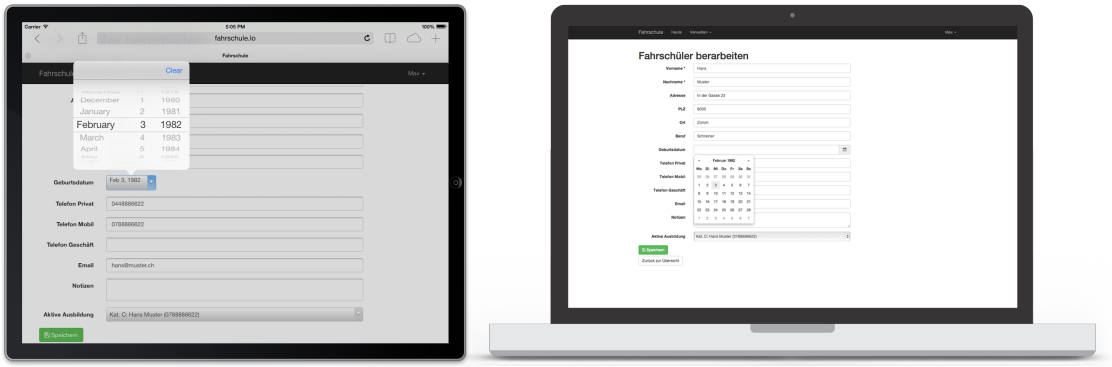


Abbildung 5.13.: Screenshots: Stammdaten bearbeiten

Bei der Entwicklung des Frontends zur Bearbeitung und Eingabe von Stammdaten wurde ein besonderes Augenmerk auf die Accessibility auf den verschiedenen Anzeigegeräten gelegt. So werden die Felder zur Eingabe von Datum- und Zeitwerten abhängig vom Gerätetyp unterschiedlich dargestellt. Kommt ein Mobilgerät wie Tablet oder Smartphone zum Einsatz, so wird die vom Betriebssystem zur Verfügung gestellte Datumsauswahl angezeigt. Auf Desktop-Computern, wo eine identische Implementation fehlt, wird

dasselbe Feld mit einem JavaScript-Datepicker versehen. Siehe dazu auch Abbildung „Screenshots: Stammdaten bearbeiten“.

Beim Speichern von Formulareingaben wird automatisch eine Validierung vorgenommen. Allfällige Fehleingaben werden detektiert und an den Benutzer zurückgemeldet.

Termin finden

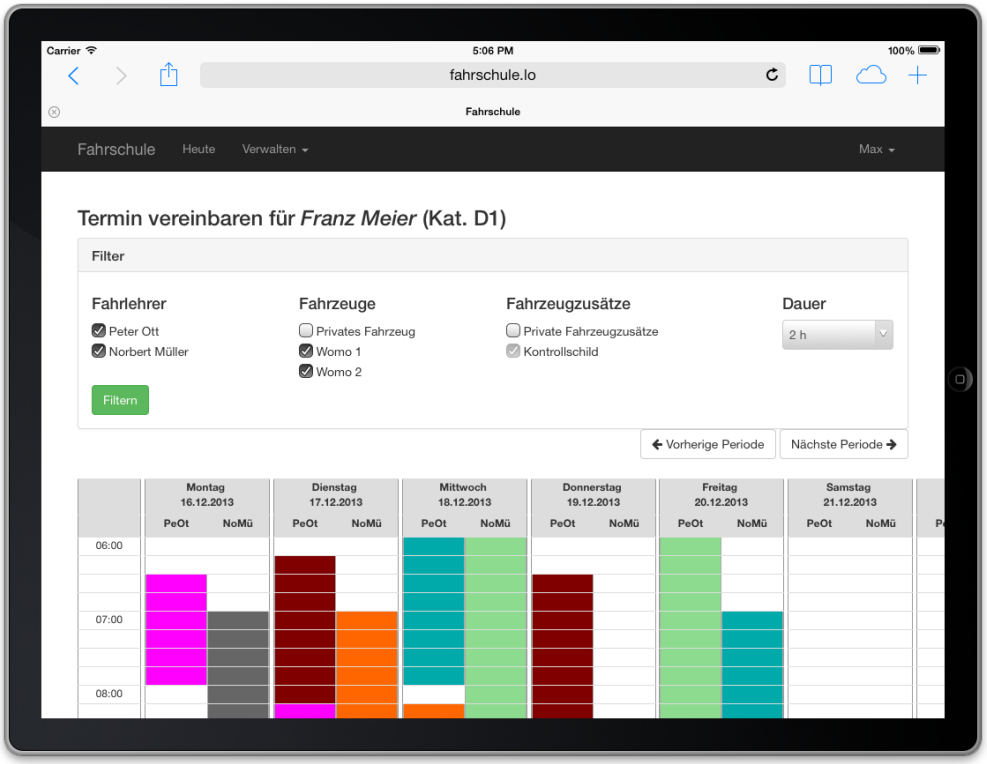


Abbildung 5.14.: Screenshot: Termin finden

Möchte der Fahrlehrer eine neue Fahrlektion vereinbaren, wird ihm eine Übersicht für eine Woche in Form eines Kalenders dargestellt. Flächen, die in hellgrün dargestellt sind, entsprechen verfügbaren Startzeiten der Fahrlektionen. Abschnitte in weiss weisen darauf hin, dass dieser Zeitpunkt keine mögliche Startzeit für eine Fahrlektion ist. Alle übrigen Flächen entsprechen bereits vereinbarten Fahrlektionen und können demnach nicht gebucht werden. Die Farbgrundierung dieser Flächen entspricht jeweils einem Fahrzeug und kann pro Fahrzeug individuell konfiguriert werden (siehe Abbildung „Screenshots: Stammdaten anzeigen“).

Es besteht ebenfalls die Möglichkeit, die Terminfindung mittels Einsatz eines Filters gemäss den Wünschen des Fahrschülers oder des Fahrlehrers entsprechend zu individualisieren. Wird ein Filter angewendet, so werden die Verfügbarkeiten neu berechnet und entsprechend wieder dem Benutzer angezeigt.

5.3.4. Frontend-Technologien

Das Frontend der Webapplikation wird unter Einsatz moderner Webtechnologien wie HTML5 und CSS3 implementiert. Diese Technologien erlauben es dem Entwickler, dem Benutzer ein vertrautes Look and Feel beim Einsatz von Mobilgeräten zu bieten. So wird zum Beispiel bei der Eingabe eines Datums auf einem iPad ein nativer Datepicker angezeigt.

Weiter werden folgende Libraries bei der Entwicklung eingesetzt:

Twig

Twig wird für die HTML-Templates im Frontend eingesetzt. Twig erlaubt die Formatierung und Darstellung von Werten, welche von der Webapplikation zur Verfügung gestellt werden. So können beispielsweise Daten zu Entities direkt über die zugrunde liegenden Getter-Methoden im Frontend ausgegeben und durch den Einsatz von Filtern formatiert werden.

Twig erlaubt es dem Entwickler, die zur Verfügung gestellten Funktionalitäten projektspezifisch durch den Einsatz von Extensions zu erweitern. Von dieser Erweiterbarkeit wird innerhalb der Webapplikation Gebrauch gemacht, um Datumswerte gemäss iCalendar-Spezifikation (siehe „RFC2445 [DS98] : 4.3.5 Date-Time“) zu konvertieren. Dieser Filter kommt bei der iCal-Schnittstelle zum Einsatz. Die Implementation wird folgendermassen vorgenommen:

```php
<?php
// src/DrivingSchool/WebBundle/Twig/DateTimeExtension.php

namespace DrivingSchool\WebBundle\Twig;

/**
 * Custom Twig extension for date time filters
 */
class DateTimeExtension extends \Twig_Extension
{
    /**
     * Returns a list of available filters.
     *
     * @return array
     */
    public function getFilters()
    {
        return array(
            new \Twig_SimpleFilter('icalDateTime', array(
              $this,
              'icalDateTimeFilter')),
        );
```

```
23      }
24
25      /**
26       * Returns a date time for a iCalender vevent
27       *
28       * @param \DateTime $date The DateTime object to convert
29       *
30       * @return string
31       */
32      public function icalDateTimeFilter(\DateTime $date)
33      {
34          $utcDateTime = clone $date;
35          $utcDateTime->setTimeZone(new \DateTimeZone('UTC'));
36          return $utcDateTime->format('Ymd\THis\Z');
37      }
38  }
```

Quellcode 5.1: Beispiel Twig Extension: DateTime Filter für iCal

Der Filter kann nun in Twig Templates verwendet werden. Folgendes Beispiel enthält einen Ausschnitt aus dem Template einer iCal-Response, der das Start- und Enddatum eines Events deklariert:

```
1 DTSTART:{{ appointment.start|icalDateTime }}
2 DTEND:{{ appointment.end|icalDateTime }}
```

Quellcode 5.2: Beispiel Twig Extension: Einbindung DateTime Filter in Template

Einen weiteren Anwendungsfall von Twig Extensions stellt das Auslesen von Klassennamen bei der Ausgabe von Entites dar. Da das Implementationskonzept dasselbe wie beim DateTime Filter darstellt, wird auf eine detaillierte Ausführung in Form von Code verzichtet.

Bootstrap

Für die Umsetzung des Webdesigns wird das Frontend-Framework *Bootstrap* in der aktuellen Version 3 eingesetzt. Bootstrap ermöglicht es dem Entwickler, responsive Designs aufbauend auf einem Grid-System zu entwickeln. Weiter werden die von Bootstrap zur Verfügung gestellten Elemente zur Gestaltung von Tabellen, Buttons und Formularen eingesetzt, um eine einheitliche Darstellung garantieren zu können.
Für die Darstellung von benutzerfreundlichen Icons kommt die Icon-Bibliothek *Font Awesome*[4] zum Einsatz.

[4] siehe *http://fontawesome.io*

Modernizr

Da der Einsatz moderner Technologien, namentlich HTML5 und CSS3, im Bezug auf die Kompatibilität von älteren Systemen und Browsern problematisch sein kann, wird das Toolkit *Modernizr* eingesetzt. Modernizr überprüft die Browserkompatibilität zu modernen HTML5- und CSS3-Features und ermöglicht es so dem Entwickler, explizit auftretende Kompatibilitätsprobleme zu beheben.

Modernizr beinhaltet die JavaScript-Library *HTML5Shiv*[5], die fehlende HTML5-Komponenten für den Internet Explorer nachrüstet. Dies erhöht ebenfalls die Rückwärtskompatibilität der Software.

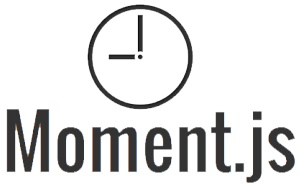
Weiter wird Modernizr für die Erkennung von Mobilgeräten eingesetzt. So kann das Look and Feel der Applikation auf das eingesetzte Medium angepasst werden. Ein Anwendungsfall für diese Unterscheidung stellen Datum- und Zeitfelder bei Formularen dar: So wird auf Mobilgeräten die vom Betriebssystem zur Verfügung gestellte Datums- und Zeitwahl eingesetzt, auf Desktop-Computern jedoch wird ein über JavaScript gesteuerter Datepicker eingesetzt.

jQuery

Die JavaScript-Library *jQuery* wird für die *DOM*-Manipulation verwendet und erlaubt das Entwickeln von modernen, benutzerfreundlichen Applikationen.

Konkret wird jQuery in der vorliegenden Webapplikation zur Manipulation von Datumswerten im Zusammenspiel mit Moment.js verwendet, um auf Desktop-Computern einen benutzerfreundlichen Datepicker einzubinden. Weiter werden Benutzerabfragen wie das Bestätigen von Löschvorgängen mittels jQuery-Events eingebunden.

Moment.js



Moment.js vereinfacht die in nativem JavaScript sehr umständliche Manipulation von Datumsobjekten.
Moment.js wird in der Webapplikation für die Konvertierung zwischen verschiedenen Datumsformaten verwendet. Diese ist notwendig, da sich die Darstellung von Datumeingaben auf Mobilgeräten (nativer Kalender des Betriebssystems) von der auf Desktop-Computern (JavaScript Datepicker) unterscheidet. Weiter müssen alle Daten durch Moment.js vor der Übermittlung von der lokalen Zeitzone in *UTC* umgewandelt werden.

[5] siehe Repository auf Github: *https://github.com/aFarkas/html5shiv*

Sass

Sass erlaubt die saubere Strukturierung von Stylesheets. Durch den Einsatz von Sass kann eine saubere *CSS*-Codebasis erreicht werden, die gut wartbar ist.
Die Sass-Dateien werden direkt auf dem Server kompiliert und als CSS-Dateien an den Client zurückgeliefert. Der Kompiliervorgang wird durch den PHP-Compiler *scssphp*[a] vorgenommen.

Folgendes Beispiel soll den Unterschied zwischen Sass- und herkömmlichem CSS-Code aufzeigen:

```
/* Sass-Code (main.scss): */

$mainBackgroundColor = #fff;
$headerBackgroundColor = #000;
body {
  background: $mainBackgroundColor;
  header {
    background: $headerBackgroundColor;
    a {
      color: #fff;
      text-decoration: none;
      &:hover {
        color: #ff0000;
        background: $mainBackgroundColor;
        text-decoration: underline;
      }
    }
  }
}

/* wird kompiliert zu CSS-Code (main.css): */

body {
  background: #fff;
}
body header {
  background: #000;
}
body header a {
  color: #fff;
  text-decoration: none;
}
body header a:hover {
  color: #ff0000;
  background: #fff;
  text-decoration: underline;
}
```

Quellcode 5.3: Beispiel: Sass-Code

[a] siehe Repository auf Github: *https://github.com/leafo/scssphp*

Die angelegte Sass-Datei kann nun mithilfe von *scssphp* und *Twig* in den HTML-Code eingebunden werden.

```
{% block stylesheets %}
    {% stylesheets debug=false output="css/vendor.css"
        '@DrivingSchoolWebBundle/Resources/public/css/vendor/bootstrap.min.
           css'
        '@DrivingSchoolWebBundle/Resources/public/css/vendor/bootstrap-
           datepicker.css'
    %}
        <link href="{{ asset_url }}" rel="stylesheet" media="screen" />
    {% endstylesheets %}

    {% stylesheets filter="scssphp" output="css/main.css"
        '@DrivingSchoolWebBundle/Resources/scss/main.scss'
        '@DrivingSchoolWebBundle/Resources/scss/font-awesome/font-awesome.
           scss'
    %}
        <link href="{{ asset_url }}" rel="stylesheet" media="all" />
    {% endstylesheets %}
{% endblock %}
```

Quellcode 5.4: Beispiel: Einbinden Sass-Code mithilfe von Twig

Sobald Änderungen am Code vorgenommen werden, wird dieser neu kompiliert und an den Client ausgegeben.

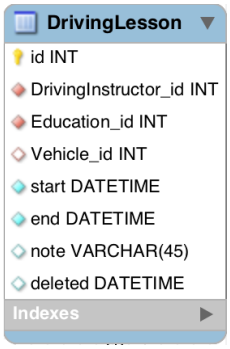
5.4. Internes Design der Webapplikation

5.4.1. Entity-Relationship-Modell

Das *Entity–relationship model* wird aufgrund des Domain Modells erarbeitet. Die Datenfelder konnten anhand des Fragenkatalogs (siehe Anhang E) ermittelt werden. Es wird nicht mehr weiter auf die Entitäten eingegangen, da diese bereits im Domainmodell erläutert werden. Zu beachten ist, dass *n:n Verbindungen* in einer MySQL Datenbank nicht abbildbar sind. Daher werden diese mittels Zwischentabelle aufgelöst. Beispiel: Um die Kompatibilität zwischen Fahrzeugen und deren Zusätzen abbilden zu können: *Vehicle* $\leftrightarrow$ *Compatibility* $\leftrightarrow$ *VehicleAddition*

Notation / Symbolik

Zur Notation wird die Martin-Notation[6] verwendet.



Durch die Tabelle „DrivingLesson“ (Fahrstunde) wird die Symbolik genauer erläutert.

[6] Martin-Notation (auch Krähenfussnotation, s. *http://de.wikipedia.org/wiki/Martin-Notation*)

Art	Feld	Typ	Beschreibung
Primary Key	id	INT	Identifikation, Primärer Schlüssel
Foreign Key	DrivingInstructor_id	INT	Fremdschlüssel zu DrivingInstructor, Pflichtfeld (NOT NULL)
Foreign Key	Education_id	INT	Fremdschlüssel zu Education, Pflichtfeld (NOT NULL)
Foreign Key	Vehicle_id	INT	Fremdschlüssel zu Vehicle, Optionales Feld
1	start	DATETIME	Datumsfeld, Pflichtfeld (NOT NULL)
2	end	DATETIME	Datumsfeld, Pflichtfeld (NOT NULL)
3	note	VARCHAR(45)	Textfeld (Limitiert auf 45 Zeichen, optional
4	deleted	DATETIME	Datumsfeld, optional

Tabelle 5.1.: ERM: Beschreibung Notation

Fahrschule

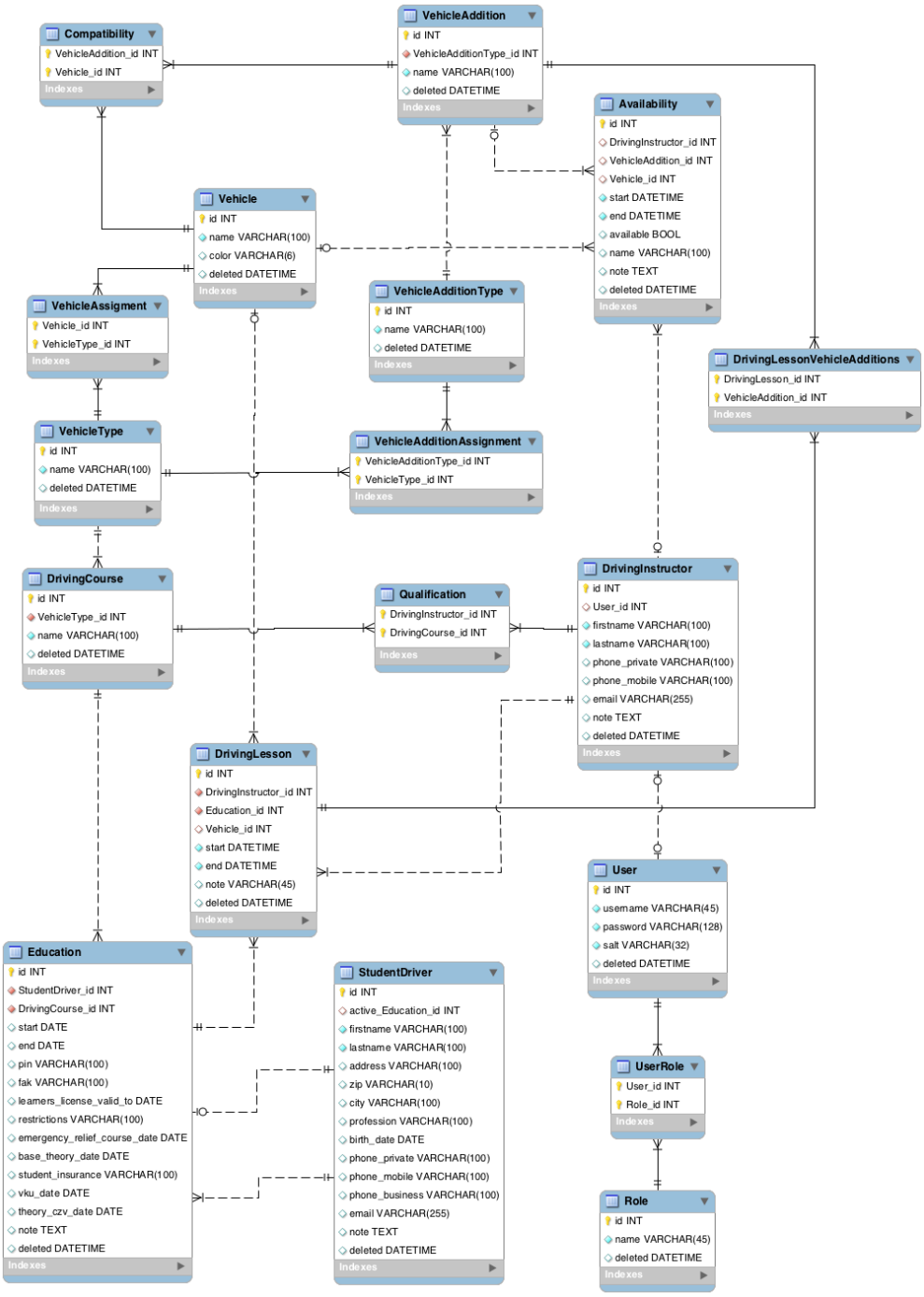


Abbildung 5.15.: ERM: Gesamtübersicht Fahrschule

Tabellen

Tabelle	Beschreibung
Vehicle	Beinhaltet alle **Fahrzeuge**.
VehicleType	Beinhaltet alle **Fahrzeugtypen**.
VehicleAddition	Beinhaltet alle **Fahrzeugzusätze**.
VehicleAdditionType	Beinhaltet alle **Fahrzeugzusatztypen**.
Availability	Beinhaltet alle **Verfügbarkeiten**. Diese können von *Fahrzeugen*, *Fahrzeugzusätzen* oder *Fahrlehrern* stammen.
DrivingCourse	Beinhaltet alle **Fahrkurse**.
DrivingInstrcutor	Beinhaltet alle **Fahrlehrern**.
DrivingLesson	Beinhaltet alle **Fahrlektionen**.
StudentDriver	Beinhaltet alle **Fahrschüler**. Diese haben jeweils eine *aktive Ausbildung*
Education	Beinhaltet alle **Ausbildungen**. Diese stellen eine Beziehung zwischen *Fahrschüler* und *Fahrkurs* dar.
User	Beinhaltet alle **Applikationsbenutzer**. Diese können einem *Fahrlehrer* zugewiesen werden.
Role	Beinhaltet alle **Applikation Berechtigungsrollen**.

Tabelle 5.2.: ERM: Beschreibung Tabellen

Zwischentabellen (*n:n*)

Zwischentabelle	Beschreibung
Compatibility	Bildet die **Kompabilitäten** zwischen *Fahrzeugen* und *Fahrzeugzusätzen* ab.
VehicleAssignment	Bildet die **Fahrzeug Zuweisungen** von *Fahrzeugen* zu den *Fahrzeugtypen* ab.
VehicleAdditionAssignment	Bildet die **Fahrzeugzusätze Zuweisungen** von *Fahrzeugzusätzen* zu den *Fahrzeugtypen* ab.
Qualification	Bildet die **Qualifikationen** von *Fahrlehrern* in *Fahrkursen* ab.
UserRole	Bildet die **Zuordnung** von in *Applikation Berechtigungsrollen* ab.

Tabelle 5.3.: ERM: Beschreibung Zwischentabellen

5.4.2. Package Diagramm

Das Package Diagramm bietet einen Grobüberblick über die wichtigsten Elemente der Webapplikation.

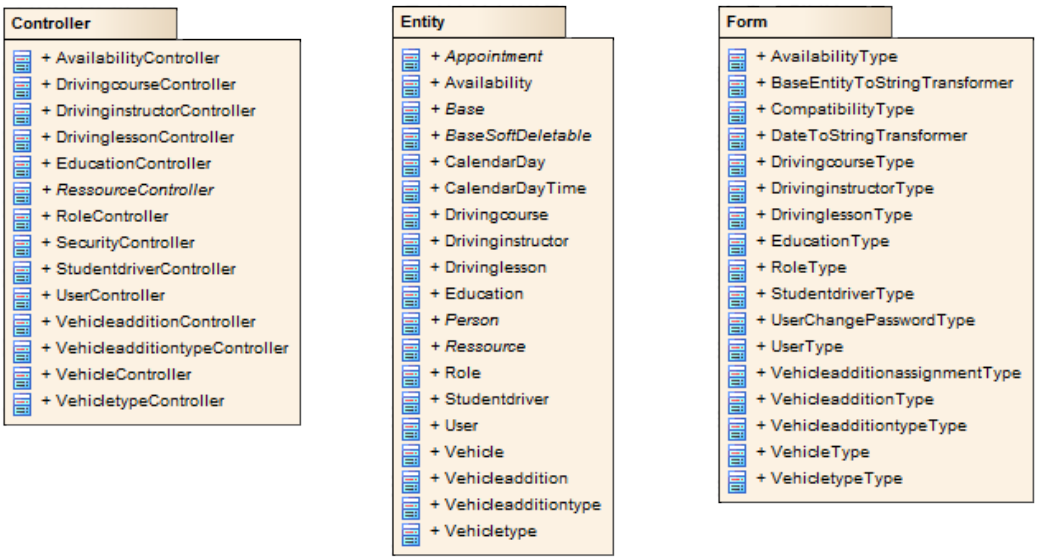


Abbildung 5.16.: Package Diagramm Fahrschule

Controller

Die Controller haben die Aufgabe, HTTP Requests entgegen zu nehmen, die Daten zur verarbeiten und dem Benutzer eine HTTP Response zu liefern. Die Businesslogik der Applikation befindet sich in den Controllern.

Entity

Die Entities repräsentieren die Objekte, welche in der Datenbank entsprechend abgelegt werden. Diese werden via Doctrine-ORM (siehe 5.4.6 „Doctrine2“) mit der Datenbank verknüpft.

Form

Die Forms dienen der Generierung der Formulare, welche für die Modifikation und Erstellung von Daten notwendig sind.

5.4.3. Klassendiagramme

Controllers

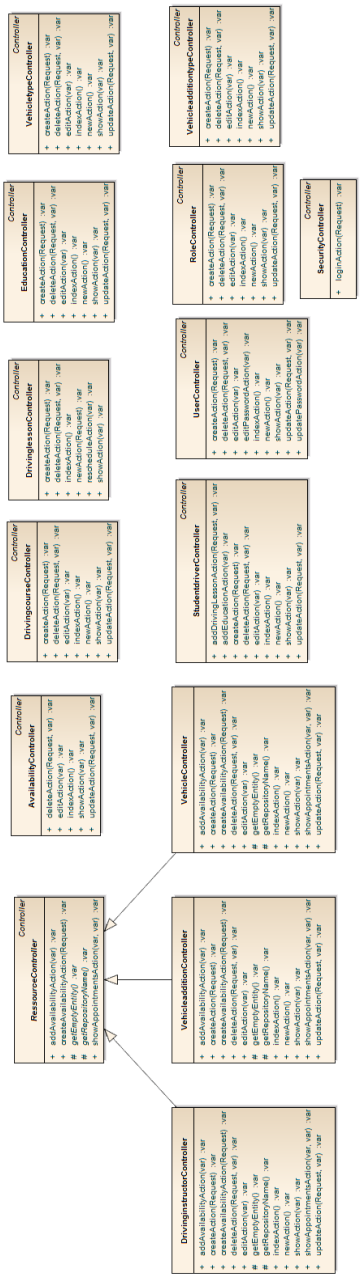


Abbildung 5.17.: Klassendiagramm Controllers

Entities (Models)

Ein abstrahiertes Klassendiagramm aller Entities wird hier dargestellt. Das ganze Diagramm ist im Anhang D zu finden.

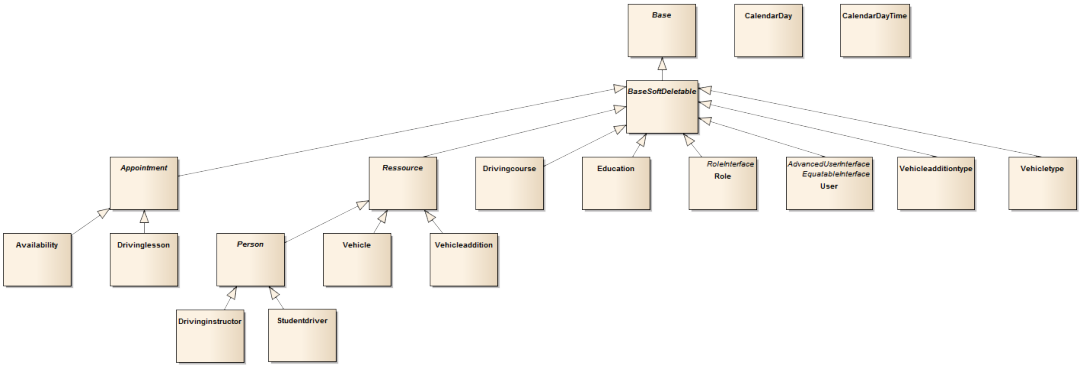


Abbildung 5.18.: Klassendiagramm Entities abstrahiert

5.4.4. Software Engineering / Design Patterns

Low Coupling / High Cohesion

Um eine hohe Qualität der Software sicherzustellen, werden die zwei wichtigen Grundsätze der objektorientierten Programmierung, *Low Coupling* und *High Cohesion* eingehalten.

Low Coupling
: Die Komponenten hängen nur schwach voneinander ab. Auch bei grösseren, internen Änderungen ist nur, beziehungsweise vorwiegend die Komponente an sich betroffen. Die Lauffähigkeit anderer Komponenten wird nicht beeinträchtigt.

High Cohesion
: Jede Komponente bietet Funktionalitäten an, die in ihren Aufgabenbereich fallen. Sie konzentriert sich auf ihren Zweck und berührt keine anderen Aufgabengebiete.

Dependency Injection

Symfony2 unterstützt das Konzept der *Dependency Injection*. Dieses Prinzip verfolgt den Ansatz, alle benötigten Ressourcen dem Objekt zu übergeben, so dass diese keine externen Ressourcen mehr selbst besorgen muss.
Beispielsweise wird für das *UserChangePasswordType* (Passwort wechseln Formular) der aktuelle Benutzer benötigt - also wird dieser „injected“ / übergeben.

```php
<?php
// src/DrivingSchool/WebBundle/Form/UserChangePasswordType.php

/**
 * User change password form.
 */
class UserChangePasswordType extends AbstractType
{
    /**
     * @var \DrivingSchool\WebBundle\Entity\User
     */
    protected $currentUser;

    /**
     * Constructor
     *
     * @param \DrivingSchool\WebBundle\Entity\User $currentUser User
     */
    public function __construct(\DrivingSchool\WebBundle\Entity\User
        $currentUser)
    {
        $this->currentUser = $currentUser;
    }

    //...
}
```

Quellcode 5.5: Beispiel: Dependency Injection - Zugriff auf aktuellen User.

Der Aufruf der Klasse sieht wie folgt aus:

```php
<?php
    // src/DrivingSchool/WebBundle/Controller/UserController.php

    //...
    $currentUser = $this->get('security.context')->getToken()->getUser();
    //...
    $editForm = $this->createForm(new UserChangePasswordType($currentUser),
        $entity);
    //...
```

Quellcode 5.6: Beispiel: Instanziierung mittels Dependency Injection

5.4.5. Abläufe

Fahrlektion-Terminfindung

Ein zentraler, komplexer Ablauf im System ist die Fahrlektion-Terminfindung. Der Fahrlehrer benötigt die Möglichkeit, schnell und sicher eine Übersicht über alle möglichen Termine zu erhalten. Dieser Ablauf wird in Form der folgenden Aktivitätsdiagramme genauer erläutert:

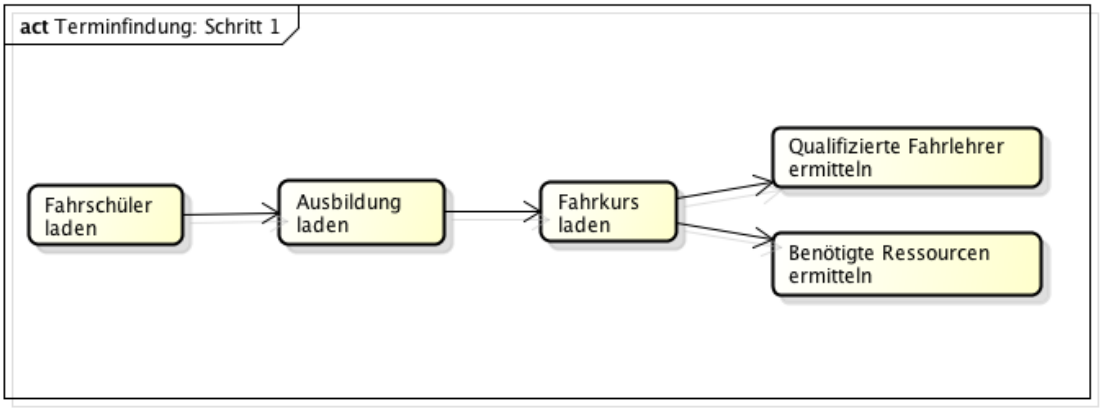


Abbildung 5.19.: Terminfindung: Schritt 1

Um mögliche Termine bestimmen zu können gibt es verschiedene Parameter, die entscheidend sind. Der Fahrschüler beziehungsweise seine aktive Ausbildung bestimmt nämlich, was für eine Fahrlektion benötigt wird. Wie man im Domainmodell erkennen kann, benötigt ein Fahrlehrer eine Qualifikation um eine Fahrlektion durchzuführen. Zudem ist pro Fahrkurs ein Fahrzeugtyp definiert, der bestimmt, welche Fahrzeuge und deren Zusätze zur Auswahl stehen.

Um einen Kalender aufzubauen, muss nun für jede Zeitspanne und Fahrlehrer ermittelt werden, ob die benötigten Ressourcen zur Verfügung stehen. Die Zeitspanne wird anhand der Startzeit des aktuellen Zeitblock und der Endzeit (*Startzeit + gewünschte Fahrlektion-Dauer*) bestimmt.

Die erste Zeitspanne ist am **Montag, 9.12.2013, 8:00 - 10:00**.

	Montag, 9.12.2013		Dienstag, 10.12.2013	
	Fahrlehrer 1	Fahrlehrer 2	Fahrlehrer 1	Fahrlehrer 2
8:00	ZEITSPANNE 1			
8:30				
9:00				
9:30				
10:00				
10:30				
11:00				
11:30				

Abbildung 5.20.: Terminfindung: Zeitspanne 1

Die zweite Zeitspanne ist am **Montag, 9.12.2013, 8:30 - 10:30**.

	Montag, 9.12.2013		Dienstag, 10.12.2013	
	Fahrlehrer 1	Fahrlehrer 2	Fahrlehrer 1	Fahrlehrer 2
8:00				
8:30	ZEITSPANNE 2			
9:00				
9:30				
10:00				
10:30				
11:00				
11:30				

Abbildung 5.21.: Terminfindung: Zeitspanne 2

Für eine solche Zeitspanne wird nun geprüft, ob erforderliche Ressourcen verfügbar sind.

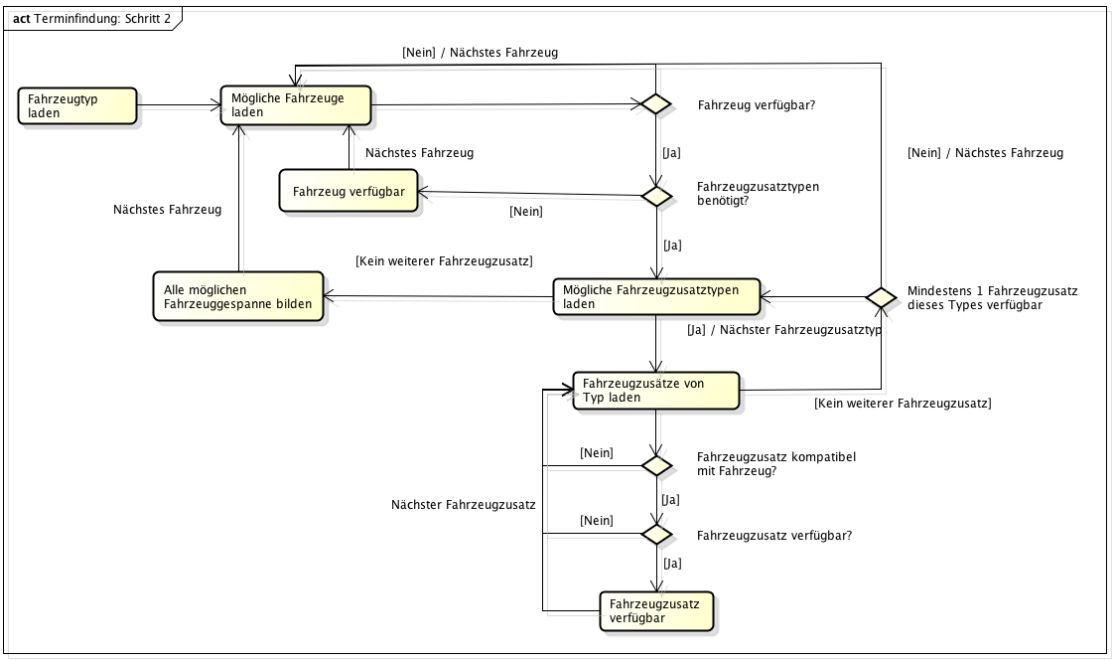


Abbildung 5.22.: Terminfindung: Schritt 2 Verfügbarkeit Ressourcen

Der Fahrzeugtyp bestimmt, welche Fahrzeuge und Fahrzeugzusatztypen in Frage kommen. Gemäss obigem Ablauf werden alle Fahrzeugspanne (Kombination aus **Fahrzeug + n Fahrzeugzusätze**) gesucht. Falls mindestens ein Fahrzeuggespann gefunden wird, gilt die Zeitspanne als verfügbar.

Definition Verfügbarkeit: Für die Ressource müssen folgende Kriterien gelten:

- Darf nicht gelöscht sein ($deleted = null$).
- Es muss eine Verfügbarkeit über die gesamte Zeitspanne vorhanden sein.
- Es darf keine Abwesenheit in dieser Zeitspanne definiert sein.
- Es darf keine andere Fahrlektion in dieser Zeitspanne stattfinden.

Terminbuchung

Verschiedene Fahrlehrer können gleichzeitig auf die Applikation zugreifen und somit auch gleichzeitig Termine für dieselben Ressourcen buchen. Daher ist es wichtig sicherzustellen, dass es dadurch zu keinen Konflikten kommt. Dies kann durch Transaktionen

sichergestellt werden. Das Lesen (Prüfen auf Verfügbarkeit) und Schreiben (Eintragen der Fahrlektion) muss in der selben Transaktion, isoliert von allen anderen Aktionen ablaufen.

Erfolgreiche Terminerstellung Eine Terminbuchung verläuft erfolgreich, wenn diese isoliert von allen anderen ablaufen kann.

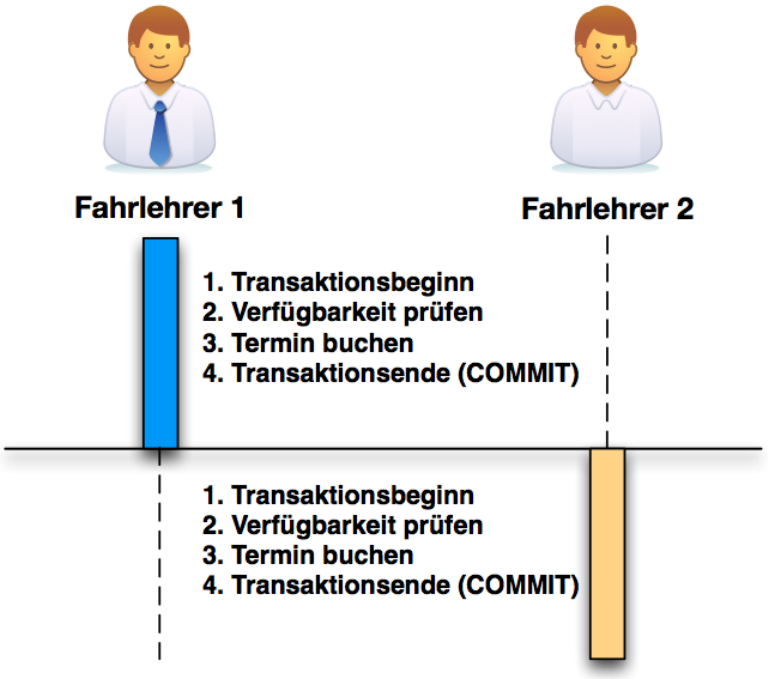


Abbildung 5.23.: Terminbuchung: Erfolgreich

Abbruch Terminerstellung Eine Terminbuchung schlägt fehl, wenn währenddessen eine andere Transaktion auf die Datenbank schreibt.

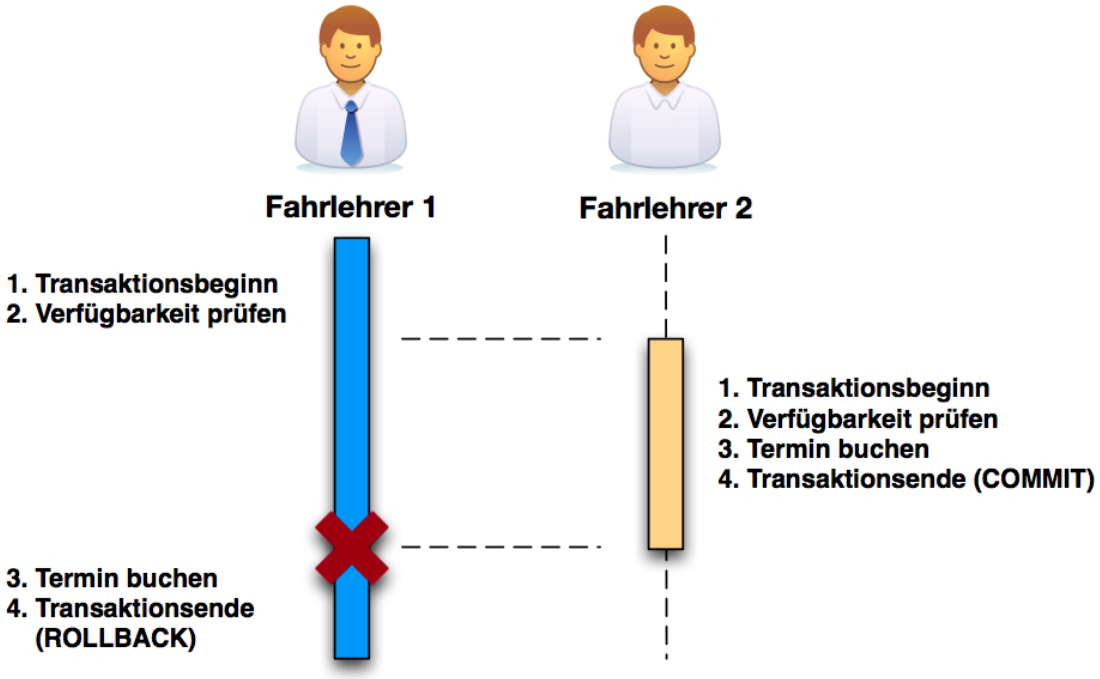


Abbildung 5.24.: Terminbuchung: Abbruch

5.4.6. Backend-Technologien

Symfony2

Symfony2 hat die Aufgabe, Requests entgegen zu nehmen und diese an die entsprechenden Controller weiterzuleiten. Danach handelt der Controller die logischen Operationen ab und gibt eine Response zurück. Folgendes Beispiel[Sena] zeigt mittels Grafik und Code auf, wie der *Request-Response-Ablauf* aussieht.

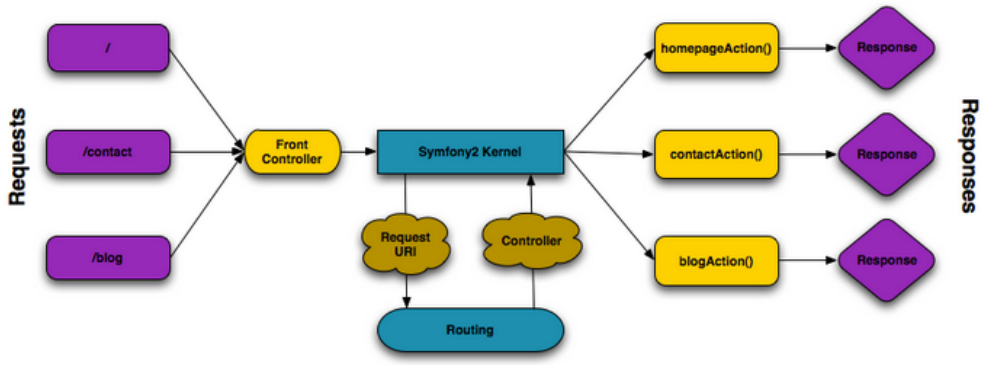


Abbildung 5.25.: Symfony2: Request-Response-Ablauf

Ein entsprechender Controller sieht wie folgt aus:

```php
<?php
use Symfony\Component\HttpFoundation\Response;

/**
 * @Route("/")
 */
class MainController
{
    /**
     * @Route("/contact", name="contact")
     * @Method("GET")
     */
    public function contactAction()
    {
        return new Response('<h1>Contact us!</h1>');
    }
}
```

Quellcode 5.7: Beispiel Symfony2 Controller

Doctrine2

Anhand eines Beispiels[Senb] wird mittels Grafik und Code erläutert, wie das Mapping von PHP Objekt zur Datenbank gemacht wird.

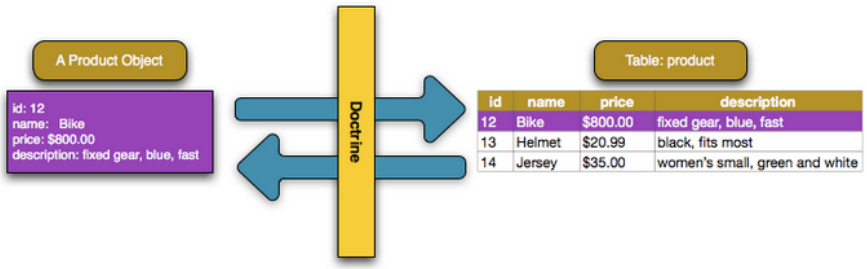


Abbildung 5.26.: Doctrine 2: Mapping

Der entsprechende Code dazu sieht folgendermassen aus:

```php
<?php
use Doctrine\ORM\Mapping as ORM;

/**
 * @ORM\Entity
 * @ORM\Table(name="product")
 */
class Product
{
    /**
     * @ORM\Column(type="integer")
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="AUTO")
     */
    protected $id;

    /**
     * @ORM\Column(type="string", length=100)
     */
    protected $name;

    /**
     * @ORM\Column(type="decimal", scale=2)
     */
    protected $price;

    /**
     * @ORM\Column(type="text")
     */
    protected $description;
}
```

Quellcode 5.8: Beispiel Doctrine ORM

5.4.7. Architektur

Physische Architektur

Die physische Architektur gestaltet sich wie folgt:

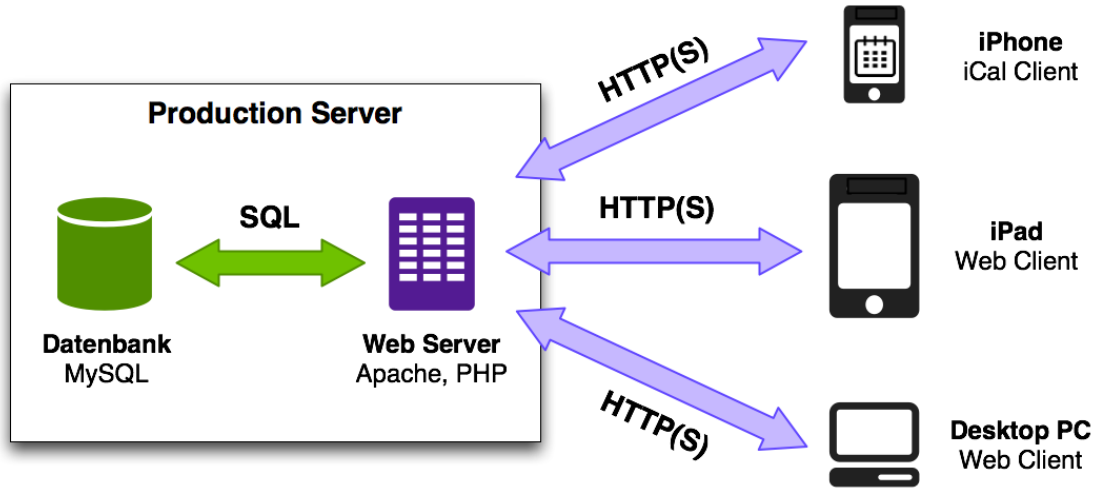


Abbildung 5.27.: Physische Architektur

Das aktuelle Setup sieht vor, dass auf einem Produktionsserver eine *MySQL*- und *Apache*-Instanz betrieben wird. Diese können auch problemlos auf 2 verschiedenen Serverinstanzen laufen. Der Zugriff der Web- bzw. iCal-Clients erfolgt via HTTP(S) Protokoll.

Logische Architektur

Die logische Architektur sieht folgendermassen aus:

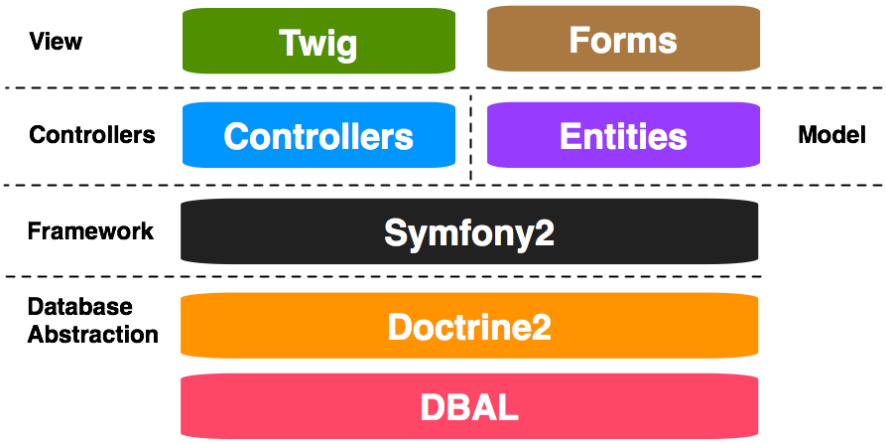


Abbildung 5.28.: Logische Architektur

Die logische Architektur wird in grob in fünf Schichten unterteilt.

View
: Der View Layer beinhaltet alle *Twig*-Templates und sowie die Forms. Diese dienen der Darstellung und werden durch den Controller aufgerufen.

Controller
: Der Controller Layer umfasst alle Controller, welche für die Business-Logik verantwortlich sind. Sie erhalten die Requests von Symfony und handeln diese ab.

Model
: Der Model Layer umfasst alle Entities, welche die Datenobjekte darstellen. Diese erlauben eine einfache Handhabung der Daten.

Framework
: Der Framework Layer umfasst *Symfony2*, welches die Grundlage für die Webapplikation darstellt.

Database Abstraction
: Der Database Abstraction Layer umfasst *Doctrine2* und *DBAL*. Er abstrahiert die Datenbank und erlaubt den einfachen Zugriff darauf.

Kapitel 6 Ergebnis

Gemäss durchgeführter Analyse wurde im Rahmen dieser Bachelorarbeit unter Einsatz moderner Webtechnologien eine Webapplikation zur Ressourcen- und Terminplanung für Fahrschulen entwickelt.

Das dabei entstandene Produkt deckt alle „Mandatory Features“ gemäss den erarbeiteten Use Cases (siehe 3.1.3 „Use Cases: Übersicht) ab.

Die Verwendung aktueller Web Application Frameworks sichern die Wartbarkeit des Produkts ab. Die responsive Weboberfläche erlaubt es dem Benutzer, auf verschiedenen Geräten wie Tablets, Smartphones und Desktop-Computern mit der Applikation interagieren zu können.

Das Deployment auf einem produktiven System ist gemäss Anleitung möglich und wird durch den Industriepartner selbst vorgenommen.

Allfällige Hilfestellungen beim produktiven Einsatz der Applikation seitens des Projektteams werden ausserhalb der Bachelorarbeit geregelt.

6.1. Ausblick

Im Rahmen der Analyse wurden verschiedene optionale Features erarbeitet. Diese beinhalten folgende Funktionalitäten:

- Pflege von Lerninhalten
- Lernfortschritte von Fahrschülern erfassen
- Erstellen von Rechnungen
- Erinnerung für Ressource verwalten
- Abfragen von Statistiken
- Rollen/Berechtigungen verwalten

Die saubere Grundlage in Form der ausgearbeiteten optionalen Use Cases (siehe 3.1.3 „Use Cases: Übersicht) ermöglicht eine effiziente Implementation der fehlenden Funktionalitäten im Rahmen einer allfälligen Nachfolgearbeit.

Anhang A **Abbildungen, Tabellen & Quellcodes**

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcodeverzeichnis

Quellcodes

Anhang B Literatur

[DS98] F. Dawson und D. Stenerson. *Internet Calendaring and Scheduling Core Object Specification (iCalendar)*. RFC 2445. Nov. 1998. URL: http://www.rfc-editor.org/rfc/rfc2445.txt (besucht am 14. 12. 2013).

[Eid58] Schweizerische Eidgenossenschaft. *Verordnung über die Zulassung von Fahrlehrern und Fahrlehrerinnen und ihre Berufsausübung*. 1958. URL: http://www.admin.ch/ch/d/gg/pc/documents/1464/Vorlage_1_d.pdf (besucht am 25. 10. 2013).

[Fou04] The Apache Software Foundation. *Apache License, Version 2.0*. http://www.apache.org/licenses/LICENSE-2.0.html. 2004. (Besucht am 08. 11. 2013).

[Fre07] Inc. Free Software Foundation. *GNU GENERAL PUBLIC LICENSE, Version 3.0*. http://www.gnu.org/licenses/gpl-3.0.txt. 2007. (Besucht am 08. 11. 2013).

[Fre91] Inc. Free Software Foundation. *GNU GENERAL PUBLIC LICENSE, Version 2.0*. http://www.gnu.org/licenses/gpl-2.0.txt. 1991. (Besucht am 08. 11. 2013).

[Gro12] The PHP Group. *The PHP License, version 3.01*. http://www.php.net/license/3_01.txt. 2012. (Besucht am 08. 11. 2013).

[ISO01] ISO 9126-1:2001. *Software engineering – Product quality*. 9126 1:2001. ISO, Geneva, Switzerland, 2001.

[Lar] Craig Larman. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development (3rd Edition)*. ISBN: 978-0131489066. URL: http://www.amazon.com/Applying-UML-Patterns-Introduction-Object-Oriented/dp/0131489062/ref=sr_1_1/002-2801511-2159202?ie=UTF8&s=books&qid=1194351090&sr=1-1 (besucht am 25. 10. 2013).

[Sena] SensioLabs. *Symfony2 and HTTP Fundamentals*. URL: http://symfony.com/doc/2.3/book/http_fundamentals.html (besucht am 17. 12. 2013).

[Senb] SensioLabs. *Symfony2 Databases and Doctrine*. URL: http://symfony.com/doc/2.3/book/doctrine.html (besucht am 17. 12. 2013).

[Tec] Massachusetts Institute of Technology. *The MIT License (MIT)*. http://opensource.org/licenses/MIT. (Besucht am 08. 11. 2013).

[Uni98] Berkeley University of California. *The BSD 3-Clause License*. http://opensource.org/licenses/BSD-3-Clause. 1998. (Besucht am 10. 11. 2013).

Anhang C Glossar

Apache

Apache ist ein quelloffener und freier Webserver der Apache Software Foundation und wird unter der Apache-Lizenz [Fou04] vertrieben.
`http://httpd.apache.org` 81, 127, 133, 136

Balsamiq Mockups

Balsamiq Mockups ist ein Software auf der Basis von Adobe Air zum Erstellen von interaktiven Mockups.
`http://balsamiq.com` 93, 133

Bootstrap

Bootstrap ist eine Sammlung von Hilfsmitteln für die Gestaltung von responsiven Webapplikationen. Es enthält auf HTML, CSS und JavaScript basierende Gestaltungsvorlagen für Typografie, Formulare, Buttons, Tabellen, Grid-System und Navigationselemente. Bootstrap wird als freie Software unter der Apache-Lizenz [Fou04] verbreitet.
`http://getbootstrap.com` 107, 133

capifony

capifony ist ein Tool für automatisierte Deployments von *Symfony2* Webapplikationen.
`http://capifony.org` 133

CRUD

"Create, Read, Update and Delete": Die Zusammenfassung der Datenmanipulationsoperationen 34, 37, 39, 44, 47, 50, 53, 56, 133

CSS

Cascading Style Sheet: Deklarative Sprache für Stilvorlagen von strukturierten HTML-Dokumenten 109, 133, 136, 137

DBAL

Database Abstraction Layer 92, 128, 133, 135

Doctrine2

Doctrine ist ein *Object Relational Mapper* (*ORM*) was auf *Database Abstraction Layer* (*DBAL*) aufsetzt. Doctrine ist in der Programmiersprache *PHP* geschrieben.
http://www.doctrine-project.org/ 128, 133

DOM

Document Object Model: Eine Spezifikation einer Schnittstelle für den Zugriff auf HTML-Dokumente 108, 133, 135

ERM

Entity–relationship model 111, 133

Git

Git ist eine freie Software zur verteilten Versionsverwaltung und steht unter der GPLv2-Lizenz [Fre91]
http://git-scm.com 133, 135

GitLab CE

GitLab CE ist eine freie auf Ruby on Rails basierende Softwarelösung zur Bereitstellung eines privaten *Git*-Repositories. GitLab CE bietet dem Entwickler eine Vielzahl von Werkzeugen zur Arbeit in Teams, wie zum Beispiel das Anlegen von Merge-Requests zum Code Review. Gitlab CE steht unter der MIT-Lizenz [Tec]
http://gitlab.org/gitlab-ce/ 133

GitLab CI

GitLab CI ist eine freie Continous Integration Software auf der Basis von Ruby on Rails. Gitlab CI steht unter der MIT-Lizenz [Tec]
http://gitlab.org/gitlab-ci/ 133

GUI

Graphical User Interface 78, 133

HTML

Hypertext Markup Language: Textbasierte Auszeichnungssprache zur Strukturierung von Webseiten 133

jQuery

jQuery ist eine freie JavaScript-Library, die Funktionen zur *DOM*-Navigation und -Manipulation zur Verfügung stellt. jQuery steht unter der MIT-Lizenz [Tec]
https://jquery.com 108, 133

LAMP

Linux *Apache* MySQL *PHP*, Programmkombination, welche im Sinne einer Software-Distribution eine Infrastruktur, in deren Rahmen dynamische Webseiten und -anwendungen entwickelt und bereit gestellt werden können, definiert. 81, 133

Modernizr

Modernizr ist eine JavaScript-Library, mithilfe welcher die von Browsern zur Verfügung gestellten HTML5- und CSS3-Features eruiert werden können. Modernizr wird als freie Software unter der MIT-Lizenz [Tec] verbreitet.
`http://modernizr.com` 108, 133

Moment.js

Sass (Syntactically Awesome Stylesheets) ist eine Stylesheet-Sprache, die als Präprozessor die Erzeugung von *CSS*-Dateien erleichtert. Sass wird als freie Software unter der MIT-Lizenz [Tec] verbreitet.
`http://sass-lang.com` 108, 133

MVC

Model-View-Controller 91, 133, 137

MySQL

MySQL ist ein relationales Datenbankverwaltungssystem, welches als Open Source unter der GPL-Lizenz [Fre91] durch die Oracle Corporation vertrieben wird.
`http://www.mysql.com` 81, 127, 133

OOP

Objektorientierte Programmierung 91, 133

ORM

Object Relational Mapper 91, 92, 133, 135

PHP

PHP: Hypertext Preprocessor, Programmiersprache die hauptsächlich zur Erstellung dynamischer Webseiten oder Webapplikationen verwendet wird. PHP wird als freie Software unter der PHP-Lizenz [Gro12] verbreitet und zeichnet sich durch breite Datenbankunterstützung und Internet-Protokolleinbindung sowie die Verfügbarkeit zahlreicher Funktionsbibliotheken aus.
`http://php.net` 81, 91, 133, 135–137

PHPDoc

PHPDoc ist ein Tool zur Dokumentation von *PHP*-Code. 133

PHPUnit

PHPUnit ist ein Testingframework für *PHP*-Code. PHPUnit ist OpenSource unter der [Uni98]
http://phpunit.de/manual/3.7/en/ 133

RUP

Rational Unified Process; Iteratives Projektvorgehen 133

Sass

Sass (Syntactically Awesome Stylesheets) ist eine Stylesheet-Sprache, die als Präprozessor die Erzeugung von *CSS*-Dateien erleichtert. Sass wird als freie Software unter der MIT-Lizenz [Tec] verbreitet.
http://sass-lang.com 109, 133

Symfony2

Symfony2 ist ein Webapplikations-Framework, das in der Programmiersprache *PHP* geschrieben ist und als freie Software unter der MIT-Lizenz [Tec] verbreitet wird. Symfony folgt dem *MVC*-Schema und unterstützt Namespaces sowie Dependency Injection Containers.
http://symfony.com 91, 92, 119, 128, 133, 134

Twig

Twig ist eine freie Template Engine für *PHP* und steht unter der MIT-Lizenz [Tec].
http://twig.sensiolabs.org 106, 110, 128, 133

Ubuntu

Ubuntu ist eine Linux-Distribution, die auf Debian basiert und unter der GPLv3-Lizenz [Fre07] vertrieben wird.
http://ubuntu.com 133

UTC

Coordinated Universal Time: Koordinierte Weltzeit 108, 133

Vagrant

Vagrant ist eine Software zur Erstellung von virtuellen Entwicklungsumgebungen. Vagrant wird als freie Software unter der MIT-Lizenz [Tec] verbreitet.
http://vagrantup.com 133

VirtualBox

VirtualBox ist eine Virtualisierungssoftware, die unter der GPLv2-Lizenz [Fre91] durch die Oracle Corporation vertrieben wird.
https://www.virtualbox.org 133

Anhang D Klassendiagramm Entitäten

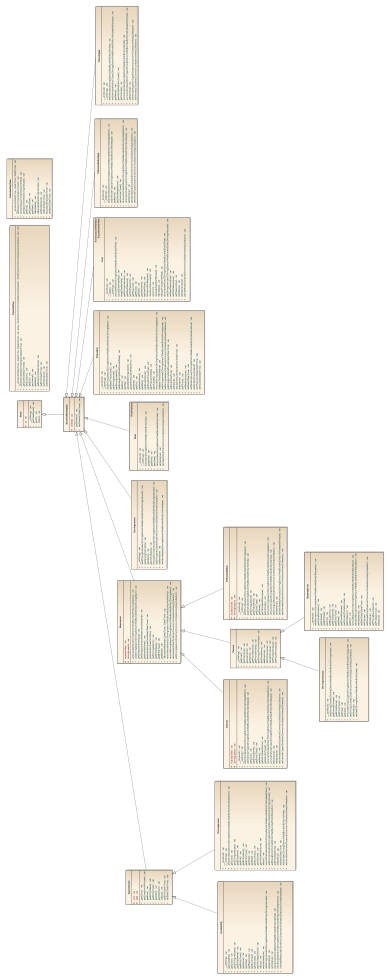


Abbildung D.1.: Klassendiagramm Entities

Anhang E Fragenkatalog anonymisiert

Um den Umfang und die Anforderungen der zu erarbeitenden Softwarelösung besser verstehen zu können, wurde vom Projektteam ein Fragenkatalog ausgearbeitet. Dieser Fragenkatalog wurde anlässlich des Kickoff-Meetings mit dem Kunden besprochen.
Dieser Fragenkatalog enthält aufgrund der Anonymisierung keine Antworten.

Ressourcen / Fahrzeuge

Nr.	Frage	Antwort	Anmerkung
1	Welche Ressourcen (Kategorien) gibt es? (Auto, Taxi, LKW, LKW Anhänger, ...)		
2	Gibt es Abhängigkeiten zwischen Ressourcen? (Bspw. LKW <=> Anhänger)		
3	Was für Lebenszyklus durchlebt eine Ressource? (Aktiv, Inaktiv, Reparatur, In Gebrauch, Verkauft, Defekt, Vermietet, ...)		
4	Wie viele Fahrzeuge pro Kategorie zählen Sie zu Ihrem Fahrzeugpark?		
5	Wer ist für Reparaturen verantwortlich?		
6	Was passiert, wenn ein Fahrzeug nicht mehr verfügbar (defekt, etc.) ist? Wie ist der Informationsfluss im Bezug auf Termine (Telefon, Nachricht an Fahrschüler)?		

Tabelle E.1.: Fragenkatalog Ressourcen / Fahrzeuge

Fahrlehrer

Nr.	Frage	Antwort	Anmerkung
7	Wie viele Fahrlehrer beschäftigen Sie?		
8	Was sind wichtige Informationen zu einem Fahrlehrer? (Name, Vorname, Geburtsdatum, Telefonnummer, E-Mail, Adresse, Dokumente? [Ausbildungsnachweise], Zusatzinfos [Evtl. Sehschwächen etc.])		
9	Gibt es Teilzeitarbeitende? Oder gibt es fixe Zeitfenster wo ein Fahrlehrer nicht verfügbar ist?		
10	In welchen Gebieten verkehren Ihre Fahrlehrer? (Netzabdeckung)		

Tabelle E.2.: Fragenkatalog Fahrlehrer

Fahrschüler

Nr.	Frage	Antwort	Anmerkung
11	Wie viele Fahrschüler sind aktiv in Ausbildung?		
12	Wie viele Fahrschüler absolvieren eine Ausbildung pro Jahr?		
13	Was sind wichtige Informationen zu einem Fahrschüler? (Name, Vorname, Geburtsdatum, Telefonnummer, E-Mail, Adresse, Dokumente? [Nothelfer, ID, ...], Zusatzinfos [Evtl. Sehschwächen etc.])		
14	Soll ein Fahrschüler Verfügbarkeiten definieren können?		
15	Soll ein Fahrschüler Termine provisorisch selber buchen können? (Bestätigt durch Fahrlehrer)		
16	Gibt es einen Verantwortlichen pro Fahrschüler?		
17	Muss/soll ein Fahrschüler selber dafür sorgen, dass er zu Terminen kommt? Meldet sich ein Fahrlehrer bei einem Fahrlehrer, falls er lange nichts mehr von einem Fahrschüler hört?		
18	Wie ist der Prozess bei einem neuen Fahrschüler? (Ruft er in der Zentrale an? Spezifisch einen Fahrlehrer? Wer erfasst ihn? Erfasst man bei der ersten Lektion noch mehr?)		
19	Was für Stati kann ein Fahrschüler annehmen? (Aktiv, Inaktiv, Austritt)		

Nr.	Frage	Antwort	Anmerkung
20	Was passiert, wenn ein Fahrschüler an einem Termin verhindert ist (Informationsfluss)?		
21	Wird ein Logbuch über einen Fahrschüler geführt? (Lerninhalte „parken etc.“ abgehakt, Notizen zur Fahrstunde)		
22	Thema Fahrprüfung: Werden Notizen z.B. bei Nichtbestehen geführt, um bei Wiederholungsfall Erkenntnisse einfliessen lassen zu können		

Tabelle E.3.: Fragenkatalog Fahrschüler

Lektionstypen

Nr.	Frage	Antwort	Anmerkung
23	Was gibt es für Lektionstypen? (Autofahrstunde, LKW Fahrstunde, ...)		
24	Welche Abhängigkeiten hat eine Lektion? (1-* Fahrlehrer benötigt, x Ressourcen benötigt, etc.)		
25	Hat eine Fahrstunde eine Fixe Dauer?		
26	Kann man definieren, welche Fahrlehrer welche Lektionen unterrichten können (Bsp. Herr Muster darf LKW Fahrstunden geben)		
27	Gibt es auch Lektionstypen ohne Ressourcen? (Bspw. Theorielektion)		
28	Gibt es Lektionstypen wo mehrere Fahrschüler teilnehmen?		
29	Gibt es Abhängigkeiten zwischen Lektionstypen? (Bspw. Autoprüfung vor Anhängerprüfung)		
30	Wenn Abhängigkeiten: Sollen/-müssen diese vom System abgebildet werden?		
31	Gibt es Ausbildungen, welche genau definieren, was für Lektionen man besuchen muss?		
32	Gibt es auch spezielle Kurse welche beispielsweise nur 1 Tag gehen?		

Tabelle E.4.: Fragenkatalog Lektionstypen

Termine

Nr.	Frage	Antwort	Anmerkung
33	Wie wird gegenwärtig die Planung Ihrer Ressourcen durchgeführt?		
34	Wie sieht ein normaler Termin aus?		
35	Gibt es Prioritäten im Bezug auf die Fahrschüler (kurz vor Prüfung = höhere Priorität)?		
36	Gibt es Termine wo mehrere Fahrschüler teilnehmen?		
37	Gibt es Termine wo mehrere Fahrlehrer teilnehmen? (Wie wird das verrechnet?)		

Nr.	Frage	Antwort	Anmerkung
38	Was kann mit einen Termin passieren? (Findet statt, Verschoben, Fahrschüler nicht erschienen, Fahrlehrer nicht erschienen, Abgebrochen, Abgesagt, Kurzfristig abgesagt, ...)		
39	Können Fahrschüler gewisse Fahrlehrer anderen vorziehen, beziehungsweise gewisse Fahrlehrer bei der Terminfindung ganz ausschliessen?		

Tabelle E.5.: Fragenkatalog Termine

Abrechnung

Nr.	Frage	Antwort	Anmerkung
40	Wie werden die Kosten einer Lektion definiert? (Fix: 80 Fr. / Fahrstunde)		
41	Gibt es Reduktionen? (Evtl. 10% für Fritz)		
42	Gibt es „Packages“? (10er Pass)		

Tabelle E.6.: Fragenkatalog Abrechnung

Technologie

Nr.	Frage	Antwort	Anmerkung
43	Welche Informationen sollen jederzeit (=offline) verfügbar sein? (z.B. Einsicht in eigene Termine bei Fahrlehrer, Ressourcen-Übersicht)		
44	Wie sieht Ihre Serverlandschaft aus?		
45	Welche Software wird gegenwärtig in Ihrer Firma eingesetzt?		
46	Welche Smartphones sind bei Ihnen im Einsatz?		
47	Welches OS / welcher Browser steht zur Verfügung?		
48	Wäre eine Hardwarebeschaffung möglich?		

Tabelle E.7.: Fragenkatalog Technologie

Sonstiges

Nr.	Frage	Antwort	Anmerkung
49	Welche Ihrer Mitarbeiter würden sich als Interviewpartner eignen?		
50	Paper Prototyping: Notwendig?		
51	Welche Rollen im Bezug auf die Applikation gibt es in der Firma?		
52	Werden Personen beschäftigt, welche mehrere Rollen / Verantwortlichkeiten inne haben?		

Tabelle E.8.: Fragenkatalog Technologie