



IPA 2018

ABSCHLUSSARBEIT

Inventarverwaltung

Alpay Yildirim
musterfirma ag
Zürcherstrasse 16
5001 Muster

Inhaltsverzeichnis

Vorwort	5
Organisation der Arbeitsergebnisse.....	6
Umfeld und Ablauf	7
Aufgabenstellung	7
Aufbau	7
Funktionen	7
Projektorganisation	9
Mittel und Methoden.....	9
Vorkenntnisse.....	9
Vorarbeiten	10
Arbeiten in den letzten 6 Monaten	10
Hilfestellung.....	10
Zeitplan.....	10
Arbeitsjournal.....	12
Montag, 5.03.2018 (Ganzer Tag).....	12
Dienstag, 6.03.2018 (Ganzer Tag)	12
Mittwoch, 7.03.2018 (Halber Tag)	13
Freitag, 9.03.2018 (Ganzer Tag)	13
Montag, 12.03.2018 (Ganzer Tag).....	13
Dienstag, 13.03.2018 (Ganzer Tag)	14
Mittwoch, 14.03.2018 (Halber Tag)	14
Freitag, 16.03.2018 (Ganzer Tag)	15
Montag, 19.03.2018 (Ganzer Tag).....	15
Dienstag, 20.03.2018 (Ganzer Tag)	15
Mittwoch, 21.03.2018 (Halber Tag)	16
Freitag, 23.03.2018 (Halber Tag).....	16
Projekt	17
Zusammenfassung (Kurzfassung).....	17
Ausgangslage (kurze Ausgangssituation)	17
Umsetzung.....	17
Ergebnis	17
Einleitung.....	18
Informieren	19

Ziele der Aufgabenstellung	19
Vorgaben	19
Fragen.....	19
Planen.....	20
Realisierungskonzept	20
Datenmodell.....	22
Testkonzept.....	23
Sitemap.....	33
Entscheiden	34
Varianten	34
Entscheid	34
Realisieren	35
Projektumgebung.....	35
Datenbank	36
Entity Framework.....	36
Partial Models	37
Dependency Injection	39
Authentifizierung & Autorisierung.....	42
Layout.....	45
Utilities	47
Artikel	47
AutoNumeric	49
Suche	50
PagedList	52
Kategorien, Räume, Benutzer, FiBu Konten.....	53
Validierung & Fehlermeldungen	53
PDF Export.....	55
Excel Export.....	56
Anleitung	57
Mehrsprachigkeit	57
Design.....	57
Kontrollieren.....	58
Testprotokoll	58
Testbericht.....	59

Reflexion.....	60
Glossar	61
Abbildungsverzeichnis.....	63
Quellenverzeichnis	64
Anhang.....	65
Anleitung	65
MSSQL	73
C#.....	74
Views (cshtml)	109

Vorwort

Diese Dokumentation gehört zur IPA von Alpay Yildirim bei der Firma Muster ag in Zürich. In dieser Dokumentation ist der Ablauf der Arbeit beschrieben und das Vorgehen dahinter.

Die Dokumentation ist in 2 Teile gegliedert.

Im ersten Teil wird die detaillierte Aufgabenstellung und der Ablauf der Arbeit aufgezeigt. Es ist ersichtlich mit welchen Mitteln gearbeitet wurde und welche Vorkenntnisse vorhanden sind.

Im zweiten Teil kommt dann die Projekt Dokumentation in der die eigentliche Arbeit beschrieben wird, was für Probleme aufgetreten sind und wie die Arbeit getestet wurde.

Durch die ganze Arbeit hinweg wurde mit der Projektplanungsmethode IPERKA gearbeitet.

Die Reflexion und das Fazit stehen am Schluss.

Organisation der Arbeitsergebnisse

Die Dokumentation sowie die restlichen Dateien, die nicht im Visual Studio bearbeitet werden können, werden täglich manuell ins Shared-Verzeichnis hochgeladen, das auf dem Server muster8 liegt.

Im Ordernamen ist immer das Datum des jeweiligen Tages vorhanden, somit kann auf jede Version zugegriffen werden.

Das Projekt wird im Visual Studio programmiert, jeden Tag werden die Arbeitsergebnisse im internen TFS mustertfs2 gespeichert. Damit kann auf jede Version zurückgegriffen werden und die Fortschritte sind gesichert.

Der Server auf dem die Datenbank gespeichert ist, macht täglich automatisch Backups.

Umfeld und Ablauf

Aufgabenstellung

Aufbau

Für einen standort- und systemunabhängigen Zugriff wird das Online Inventar als Webapplikation aufgebaut. Die technische Auslegung gliedert sich in eine dreischichtige Applikationsarchitektur mit jeweils folgendem Aufbau:

Presentation Layer: .NET MVC5 Frontend Applikation in C#/Razor

- Model: Zugriff auf Data Access Layer mittels Dependency Injection.
- View: GUI Design basierend auf .Net Razor, HTML5, CSS3, JQuery, Bootstrap Framework.
- Controller: Zugriff auf Business Logic Layer mittels Dependency Injection mit zusätzlicher GUI Logik basierend auf MVC Scaffolding Items (CRUD).
- Authentifizierung: Authentifizierung an Windows AD via LDAP.
- Autorisierung: Autorisierung in Applikation mittels OWIN Identity.

Business Logic Layer: .NET Class Library in C#

- Services: Klassen mit Funktionen für die Applikationslogik welche den Datenzugriff vom Presentation Layer auf Data Access Layer moderieren.
- Interfaces: Interfaces auf die entsprechenden Klassen für die Implementierung mittels Dependency Injection.

Data Access Layer: .NET Class Library in C#

- Models: Entity Data Model Klassen, welche das Datenbankmodell mittels Entity Framework abbilden.
- ModelsPartial: Erweiterungen der Entity Data Model Klassen für Business- und Präsentationslogik.
- Accessors: Klassen mit Funktionen für den Datenzugriff von Business Logic Layer auf die Datenbank.
- Interfaces: Interfaces auf die entsprechenden Klassen für die Implementierung mittels Dependency Injection.
- Persistence: MS SQL Datenbank für die Datenspeicherung.

Funktionen

Die Benutzerinteraktion umfasst folgende Funktionen:

Zugriff

- Login: Eingabe Login Daten und Weiterleitung an Windows AD via LDAP zur Authentifizierung. OWIN SignIn zum Anlegen der applikationsinternen Identität für die Autorisierung der Zugriffe.
- Logout: OWIN SignOut und Redirect auf Login-Seite.

Datenerfassung

- Benutzer und Rollen: Erfassung Benutzer und Zuordnung auf Rollen durch den Administrator.

- FiBu-Konti: Erfassung FiBu-Konti durch den Benutzer.
- Kategorien: Erfassung Kategorien und Zuordnung auf FiBu-Konti durch den Benutzer.
- Räume: Erfassung Räume und Zuordnung auf Lokalisierung durch den Benutzer.
- Artikel: Erfassung Datensätze und Zuordnung auf Kategorien, Räume und Nutzer durch den Benutzer.

Datenbearbeitung

- Funktionen zum Bearbeiten und Löschen von Benutzern.
- Funktionen zum Bearbeiten und Löschen von FiBu-Konti.
- Funktionen zum Bearbeiten und Löschen von Kategorien.
- Funktionen zum Bearbeiten und Löschen von Räumen.
- Funktionen zum Bearbeiten und Löschen von Artikeln.

Datenausgabe

- Suche: Filterfunktion unter Einbezug von Kategorie, FiBu-Konto, Raum, Nutzer, Beschaffungsdatum. Kombiniert mit Freitextsuche nach Stichworten in Inventar-Nummer, Bezeichnung und Beschreibung.
- Auswertung: Funktionen für den Export von Suchresultaten in Excel und den Druck von Labels mittels PDF.

Funktionstest

- Alle Funktionen müssen mit der Erfassung, Bearbeitung, Ausgabe und Löschung über alle Entitäten auf die Korrektheit der Ergebnisse geprüft und bis zur fehlerfreien Funktionsweise ausgearbeitet werden.

Dokumentation

- Kurze Anleitung der Funktionsweise für den innerbetrieblichen Endanwender; in der Anwendung online abrufbar.

Projektorganisation

Lehrbetrieb und Durchführungsort:

muster ag
Zürcherstrasse 32
5001 Zürich
062 711 11 11
info@muster.ch

Kandidat:

Yildirim Alpay
Musterstrasse 8, 5001 / Zürich (AG)
062 711 11 11 (am besten erreichbar)
062 711 11 11
alpay.yildirim@muster.ch

BerufsbildnerIn/ Lehrfirma:

BerufsbildnerNachname BerufsbildnerVorname
muster ag
Zürcherstrasse 32, 5001 / Zürich
062 711 11 11(am besten erreichbar)
062 711 11 11
BerufsbildnerVorname.BerufsbildnerNachname@muster.ch

Verantwortliche Fachkraft:

Mustermann Max
muster ag
Zürcherstrasse 32, 5001 / Zürich (AG)
+41 62 711 11 11 (am besten erreichbar)
+41 62 711 11 11
Max.Mustermann@muster.ch

Hauptexperte:

ExpertenVorname ExpertenNachname
062 711 11 11 (am besten erreichbar)
h.r.ExpertenVorname@muster.ch

Mittel und Methoden

Programmierung als MVC5 Applikation in .NET/C#/Razor in Visual Studio

- Frontend-Programmierung in HTML5/CSS3/JQuery
- Dreischichtige Applikationsarchitektur (Data Access Layer, Business Logic Layer, Presentation Layer)
- Entity Framework
- Dependency Injection
- Authentifizierung mittels LDAP
- Autorisierung mittels OWIN Identity

Vorkenntnisse

Der Aufbau im vorgegebenen Rahmen ist dem Kandidaten bekannt und wurde mehrheitlich seit ca. 1½ Jahren von ihm eingesetzt.

Vorarbeiten

Programmierte Komponenten, die bei uns regelmässig für den Aufbau von Projekten eingesetzt werden stehen dem Kandidaten (ebenso wie in der täglichen Arbeit für Kunden) auch für dieses Projekt zur Verfügung und müssen im Rahmen dieser Aufgabe nicht noch einmal alle von Grund auf neu entwickelt werden (analog der Anforderung und Handhabung im täglichen Umfeld mit Kundenprojekten).

Arbeiten in den letzten 6 Monaten

Entwicklung von Webapplikationen, darunter mit und ohne Verwendung eines .NET CMS sowie auf WebForms wie auch auf MVC Basis.

Die beiden grössten Aufträge waren:

- Weiterentwicklung interne IT Management Webapplikation
- Kunden Websites auf CMS Basis

Die eingesetzten Werkzeuge dafür waren:

- Visual Studio
- SQL Management Studio

Hilfestellung

Der Fachvorgesetzte Max Mustermann steht zur Verfügung bei allfälligen Problemen oder Fragen.

Es wurden Tutorials für die folgenden Code Abschnitte benutzt und sind auch während der Projektarbeit verfügbar. Die Links zu den jeweiligen Tutorials befinden sich im Quellenverzeichnis:

- PDF
- Excel
- LDAP
- MetaData
- Unity / Dependency Injection
- OWIN

Das Projekt ch.muster.se.inv wurde schon im Vorfeld auf dem TFS als leere Solution angelegt.

Zeitplan

Folgende Termine sind für den Ablauf der IPA gültig:

Starttermin: 05.03.2018

Datum der Abgabe: 23.03.2018

Mit Hilfe der Aufgabenstellung wurde folgender Zeitplan erstellt. Darin ist der Soll-ist Vergleich klar ersichtlich. Der Zeitplan ist aufgebaut im 2-Std-Raster, Violett ist die Soll-Zeit und Rot die Ist-Zeit. Grün sind die Meilensteine, der erste Meilenstein ist das Abschliessen der Phase Entscheidung und der zweite das Abschliessen der Realisierung.

Sollzeit	
Istzeit	
Meilenstein	

Arbeitsjournal

Montag, 5.03.2018 (Ganzer Tag)

Tätigkeiten	Aufbau der Dokumentation Zeitplan grob erstellen Erster Teil der Dokumentation Aufgabenstellung studieren Kriterienkatalog studieren
Erreichte Ziele	Aufgabenstellung verstanden Erster Teil der Dokumentation erstellt Phase «Informieren» abgeschlossen
Probleme	-
Hilfestellung	-
Ausserplanmässige Arbeiten	-
Reflexion	Ich konnte gut in meine IPA starten, der erste Tag verlief ohne Probleme. Die Erstellung des Zeitplanes und die Planung dazu kosteten mich mehr Zeit als erwartet. In der Dokumentation konnte ich schon mal den ersten Teil fertigstellen. Ich habe noch die Aufgabenstellung und den Kriterienkatalog studiert.

Dienstag, 6.03.2018 (Ganzer Tag)

Tätigkeiten	Konzept für die Realisierung erstellen Modell der Datenbank erstellen Testkonzept erstellen Lösungsvariante festlegen Erstes Gespräch mit dem Experten
Erreichte Ziele	Konzept für die Realisierung fertiggestellt Testkonzept fertiggestellt Phase «Planen» und «Entscheiden» abgeschlossen Meilenstein «Entscheiden abgeschlossen» erreicht
Probleme	-
Hilfestellung	-
Ausserplanmässige Arbeiten	-
Reflexion	Ich habe die Grösse des Testkonzeptes für dieses Projekt unterschätzt, es waren dann doch mehr Testfälle nötig als ich erwartet habe um alle Funktionen abzudecken. Ich konnte alles erledigen was für den heutigen Tag vorgesehen war, es wurde aber recht knapp. Der erste Besuch des Experten lief gut ab und alle Fragen die ich noch offen hatte konnten beantwortet werden.

Mittwoch, 7.03.2018 (Halber Tag)

Tätigkeiten	Einrichten der Projektumgebung Erstellen der Datenbank
Erreichte Ziele	Projektumgebung eingBerufsbildnerVornametet Datenbank erstellt
Probleme	-
Hilfestellung	-
Ausserplanmässige Arbeiten	-
Reflexion	Dies war der erste halbe Tag meiner IPA. Ich konnte alle geplanten Tätigkeiten abschliessen und konnte mir sogar mehr Zeit verschaffen, da ich mit dem Einrichten der Projektumgebung schneller fertig wurde als erwartet und die Erstellung der Datenbank schon heute abschliessen konnte.

Freitag, 9.03.2018 (Ganzer Tag)

Tätigkeiten	Dependency Injection umsetzen Authentifizierung mit OWIN & LDAP beginnen
Erreichte Ziele	Dependency Injection umgesetzt Authentifizierung per OWIN & LDAP begonnen
Probleme	-
Hilfestellung	-
Ausserplanmässige Arbeiten	-
Reflexion	Der heutige Tag verlief gut, ich konnte die Dependency Injection im Projekt umsetzen und konnte sogar schon mit der Authentifizierung beginnen.

Montag, 12.03.2018 (Ganzer Tag)

Tätigkeiten	Authentifizierung per OWIN & LDAP implementieren Autorisierung (Login & Logout), Rollen fertigstellen Übersicht / Details Artikel, Kategorien, Räume, Benutzer, FiBu Konten Erstellen
Erreichte Ziele	Authentifizierung per OWIN & LDAP implementiert Autorisierung (Login & Logout), Rollen fertiggestellt Übersicht / Details Artikel, Kategorien, Räume, Benutzer, FiBu Konten erstellt
Probleme	-
Hilfestellung	-
Ausserplanmässige Arbeiten	-

Reflexion	Der heutige Tag verlief leider nicht ganz wie geplant. Die ganze Implementierung vom Login und den Rollen die der Benutzer haben kann war doch umfänglicher als ich gedacht habe, vor allem in Anbetracht der Dokumentation. Trotzdem konnte ich alle Arbeiten erledigen und auch schon mit den Views der Models beginnen.
-----------	--

Dienstag, 13.03.2018 (Ganzer Tag)

Tätigkeiten	Übersicht / Details Artikel, Kategorien, Räume, Benutzer, FiBu Konten fertigstellen Bearbeitung Artikel, Kategorien, Räume, Benutzer, FiBu Konten fertigstellen Validierung Felder (Datum, Preis, Dropdown u.s.w) fertigstellen
Erreichte Ziele	Übersicht / Details Artikel, Kategorien, Räume, Benutzer, FiBu Konten fertiggestellt Bearbeitung Artikel, Kategorien, Räume, Benutzer, FiBu Konten fertiggestellt Validierung Felder (Datum, Preis, Dropdown u.s.w) fertiggestellt Zusätzliche Sachen implementieren, wie das ErrorSummary das beim Login zurückgegeben wird und dass in der Navigation der aktive Navigationspunkt angezeigt wird, eine schönere Validierung mit deutlicheren Farben. Dazu noch das Feld Preis mit autonumeric.js.
Probleme	-
Hilfestellung	-
Ausserplanmässige Arbeiten	-
Reflexion	Der heutige Tag verlief sehr gut, ich wurde mit den Tätigkeiten schneller fertig als erwartet und bin dem Zeitplan schon wieder voraus, nebenbei konnte ich auch zusätzliche Sachen hinzufügen, die das Projekt über die Aufgabenstellung hinaus verbessern.

Mittwoch, 14.03.2018 (Halber Tag)

Tätigkeiten	Suche (Übersicht Artikel) implementieren Dokumentieren
Erreichte Ziele	Suche (Übersicht Artikel) implementiert
Probleme	-
Hilfestellung	-
Ausserplanmässige Arbeiten	-
Reflexion	An diesem halben Tag verlief alles wie geplant, da ich das letzte Mal schneller

	fertig wurde als erwartet, konnte ich mehr Zeit in den nächsten Punkt investieren und wurde mit der Suche fertig. Nun habe ich im Zeitplan einen Vorsprung.
--	---

Freitag, 16.03.2018 (Ganzer Tag)

Tätigkeiten	Excel & PDF Export fertigstellen Anleitung Inventarverwaltung beginnen
Erreichte Ziele	Excel & PDF Export fertiggestellt Anleitung Inventarverwaltung begonnen
Probleme	-
Hilfestellung	-
Ausserplanmässige Arbeiten	-
Reflexion	Heute verlief alles wie erwartet, ich bin zeitlich im Vorsprung und konnte schon mit der Anleitung für die Inventarverwaltung beginnen.

Montag, 19.03.2018 (Ganzer Tag)

Tätigkeiten	Anleitung Inventarverwaltung fertigstellen Zusätzliche Tätigkeiten (Mehrsprachigkeit implementieren) Dokumentation ergänzen
Erreichte Ziele	Anleitung Inventarverwaltung fertiggestellt Mehrsprachigkeit implementieren Quellen & Glossar vervollständigt Phase «Realisieren» abgeschlossen Meilenstein «Realisieren abgeschlossen» erreicht
Probleme	-
Hilfestellung	-
Ausserplanmässige Arbeiten	-
Reflexion	Der heutige Tag verlief sehr gut, nur bei der Anleitung benötigte ich eine halbe Stunde mehr Zeit als erwartet, das hatte aber keinen grossen Einfluss auf die Zeitplanung, da ich schon 5 Stunden im Zeitplan voraus bin. Deswegen habe Ich in der Webapplikation zusätzlich noch die Mehrsprachigkeit implementiert.

Dienstag, 20.03.2018 (Ganzer Tag)

Tätigkeiten	Umgebung fürs Testen einrichten Testfälle Testen Zweites Gespräch mit dem Experten Dokumentation ergänzen
Erreichte Ziele	Umgebung fürs Testen eingBerufsbildnerVornametet Testfälle getestet
Probleme	-

Hilfestellung	-
Ausserplanmässige Arbeiten	-
Reflexion	Heute lief alles wie erwartet, Ich konnte alles erledigen was geplant war und konnte meine Dokumentation inhaltlich verbessern, nebenbei habe ich darauf geachtet, dass alles so eingehalten wurde, wie es im Kriterienkatalog beschrieben ist.

Mittwoch, 21.03.2018 (Halber Tag)

Tätigkeiten	Anhänge in die Dokumentation hinzufügen Reflexion schreiben
Erreichte Ziele	Anhänge in die Dokumentation hinzufügen Reflexion schreiben
Probleme	-
Hilfestellung	-
Ausserplanmässige Arbeiten	-
Reflexion	Der heutige halbe Tag verlief so wie erwartet, die Dokumentation ist nun fertiggestellt. Es ist ein gutes Gefühl das alles bis jetzt so gut geklappt hat.

Freitag, 23.03.2018 (Halber Tag)

Tätigkeiten	Kriterienkatalog betrachten und prüfen ob alles eingehalten wurde IPA BBerufsbildnerVorname abgeben
Erreichte Ziele	Kriterienkatalog betrachtet und geprüft ob alles eingehalten wurde IPA BBerufsbildnerVorname abgegeben
Probleme	-
Hilfestellung	-
Ausserplanmässige Arbeiten	-
Reflexion	Der letzte Tag verlief reibungslos, ich habe mir wieder die ganze Dokumentation durchgelesen und geschaut das auch wirklich alle Kriterien eingehalten wurden. Ich bin mit dem Resultat und der geleisteten Arbeit sehr zufrieden.

Projekt

Zusammenfassung (Kurzfassung)

Diese Zusammenfassung richtet sich an Leser mit Fachwissen in der Informatik und vermittelt eine erste Übersicht, welche zur Erleichterung dienen soll um die Arbeit und deren Inhalt verständlicher zu machen.

Ausgangslage (kurze Ausgangssituation)

Gegenwärtig wird das Inventar des Mobiliars und der Büroausrüstung der Firma muster ag in einer Excel Tabelle geführt. Neu soll in einer internen Webapplikation unser Inventar online erfasst und ausgewertet werden.

Dabei sollen Artikel zugeordnet werden auf Kategorien, FiBu Konti, Räume und Nutzer. Mit diesen Assoziierungen sollen Inventarlisten für die Buchführung und Abschreibungen, für die Erhebung des Versicherungswertes, für die Ausrüstung einzelner Mitarbeiter und Räume sowie für das Lifecycle Management unserer EDV Ausrüstung generiert werden. Das Projekt wird im internen Umfeld mit den in Kundenprojekten ebenfalls angewendeten Technologien umgesetzt.

Umsetzung

Die Inventarverwaltung wird als ASP.NET MVC Webapplikation realisiert, die im Inneren aus drei Schichten besteht, dabei besteht die Grundarchitektur aus dem Business Logic Layer, Data Access Layer und dem Presentation Layer. Mittels Dependency Injection sollen diese drei Schichten (Layers) untereinander referenziert werden, dies ist möglich mit Unity.

Durch die ganze Webapplikation wird das MVC-Pattern angewandt. Mit dem ADO.NET Entity Data Model geschieht der Zugriff auf die Datenbank. In den Controller Klassen liegt die Logik. Mithilfe von Services im Business Logic Layer und Accessors im Data Access Layer. Der Code der für die Darstellung benötigt wird, wird in die Views mit Razor geschrieben.

Zur Sicherheit vor fremden Zugriffen ist die Webapplikation passwortgeschützt. Die Autorisierung wird innerhalb der Applikation geregelt mit OWIN Identity.

Ergebnis

Die Inventarverwaltung hat 5 Bereiche (Artikel, Kategorien, Räume, Benutzer, FiBu Konten), jeder Bereich besitzt die jeweiligen CRUD Methoden (Create, Read, Update, Delete).

Ein Artikel hat eine Inventar Nummer, Bezeichnung, Beschreibung, Kategorie, Raum, Nutzer, Beschaffungsdatum, Beschaffungswert. Das Beschaffungsdatum kann mithilfe eines Date Pickers gewählt werden.

Die Artikel können als Excel File in Form einer Liste heruntergeladen werden oder als PDF. Das PDF ist dabei so gelayoutet, dass es verwendet werden kann als Vorlage für Labels, die man dann später auf die jeweiligen Gegenstände kleben kann.

Die Artikel die in der Inventarverwaltung erfasst sind, können mit verschiedenen Suchfiltern gefiltert werden.

Einleitung

In diesem Kapitel beschäftige ich mich mit dem Aufbau des Projektes. Das Projekt ist nach der IPERKA-Methode aufgebaut, damit stets eine gegliederte Struktur im Projektablauf ersichtlich ist. Hierzu sind die 6 folgenden Schritte notwendig oder wie sie auch genannt werden «Phasen».

Informieren:

Der Auftrag und dessen Aufgabenstellung werden so gut wie möglich geklärt, die Informationen werden gewertet und sortiert und wesentliche Punkte so früh wie möglich erkannt sowie allfällige Fragen geklärt. Dies ist wichtig um den Zeitplan fertigzustellen und zum Verständnis was überhaupt erwartet wird.

Planen:

Um einen guten und strukturierten Projektablauf zu garantieren muss das Projekt präzise geplant werden. Wenn möglich werden mehrere Lösungswege für komplexere Abläufe geplant. Dabei wird ein Testkonzept, sowie ein Realisierungskonzept erstellt.

Entscheiden:

Die unterschiedlichen Lösungswege werden miteinander verglichen und davon wird dann der bestmögliche ausgewählt. Dabei muss man sich überlegen ob die Ideen sinnvoll sind und ob man es dann auch wirklich so umsetzen kann.

Realisieren:

Erst wenn der Ablauf klar definiert ist und alle Entscheidungen getroffen sind, wird mit der Realisierung begonnen. Die Arbeitsabläufe werden stetig protokolliert und die Ist-Werte werden im Zeitplan eingetragen.

Kontrollieren:

Die Resultate müssen so getestet werden wie es im Testkonzept steht, aber erst wenn die Entwicklung vollständig beendet ist. Fehlgeschlagene Test werden hierbei direkt korrigiert und behoben oder als Fehler vermerkt und dokumentiert.

Auswerten:

Zum Schluss gibt es eine Reflexion über die Arbeit und die Erfahrungen die man gesammelt hat. Dabei geht man durch alle Schritte die man innerhalb des Projekts gemacht hat und es wird untersucht was gut war und wo Verbesserungen möglich sind. Das ist auch praktisch für die Umsetzung zukünftiger Projekte.

Informieren

Das Informieren ist die erste Phase der IPERKA-Methode.

Das Klären von wichtigen Fragen und das genaue Durchlesen der Aufgabenstellung sind die wichtigsten Punkte beim Informieren, ebenfalls kann es nie schaden sich schon im Voraus Gedanken über mögliche Probleme zu machen.

Ziele der Aufgabenstellung

In der Webapplikation kann jeder Benutzer der angemeldet ist Artikel, Kategorien, Räume und FiBu Konten erstellen, löschen oder ändern.

Nur ein Benutzer der die Rolle «Administrator» hat darf Benutzer erstellen, löschen oder ändern.

Die Webapplikation wird intern aufrufbar sein unter `inv.se.muster.ch`.

Damit die Daten in Sicherheit sind und fremder Zugriff verhindert werden kann muss die Webapplikation mit einem Passwort geschützt sein. Die Mitarbeiter der muster ag sollen sich mit ihren gewohnten Zugangsdaten anmelden können.

Die Artikel können als Excel exportiert werden.

Die Artikel können auch als PDF exportiert werden und dann für Labels benutzt werden.

Mit Hilfe von Suchfiltern können die Artikel nach spezifischen Kriterien gefiltert werden.

Eine kurze Anleitung über die Benutzung der Inventarverwaltung muss erstellt werden und durch einen Link aufrufbar sein in der Applikation.

Vorgaben

Die Webapplikation ist in drei Schichten unterteilt:

- Data Access Layer (C# Library, Entity Framework)
- Presentation Layer (C#, ASP.Net MVC 5)
- Business Logic Layer (C# Library)

Der gegenseitige Zugriff innerhalb der Schichten erfolgt per Dependency Injection.

Die Daten werden auf einer MS SQL Datenbank gespeichert

Fragen

Frage an Max Mustermann (05.03.2018): Ist es Teil der IPA, das Projekt auf dem Webserver zu veröffentlichen?

Antwort: Der Webserver ist schon eingerichtet. Am Ende muss Alpay Yildirim nur noch das Projekt in das Verzeichnis «`ch.muster.se.inv`» hochladen, das sich auf dem Server «`ehost51`» befindet.

Planen

Das Planen ist die zweite Phase der IPERKA-Methode.

In dieser Phase wird der Zeitplan fertiggestellt und das Testkonzept sowie das Realisierungskonzept erstellt. Das Datenmodell mit den jeweiligen Tabellen und Feldern wird entworfen.

Realisierungskonzept

Drei neue Projekte werden in der leeren Solution ch.muster.se.inv auf dem TFS mustertfs2 erstellt.

- ch.muster.se.inv.dal für den Data Access Layer
- ch.muster.se.inv.web für den Presentation Layer
- ch.muster.se.inv.bll für den Business Logic Layer

Die Datenbank wird wie im Datenmodell abgebildet implementiert und mit dem Entity Framework im DAL in dem Ordner «Models» erstellt.

Im DAL hat es insgesamt sechs Accessors die den Zugriff auf das Model regeln. Für diese sechs Accessors wurden auch Interfaces erstellt. In ihnen sind jeweils die CRUD (Create-Read-Update-Delete) Methoden vorhanden.

Im BLL werden die sechs Services erstellt für alle Accessors. Momentan dient es nur zur Weiterleitung vom WEB zum DAL. Über die Accessor Interfaces greifen sie auf die Accessors zu.

Als ASP.NET MVC 5 Web Application wird das WEB angelegt, somit hat es schon mal den fundamentalen Aufbau.

Die Projekte werden nun untereinander referenziert und die Klassen den Interfaces zugeordnet. Dies ist möglich mittels Dependency Injection, das mit Unity umgesetzt wird. BLL referenziert DAL und das WEB referenziert das BLL und DAL.

Unter Controllers im WEB können die Controller zu den jeweiligen Models nun automatisch erstellt werden. Nur der direkte Zugriff vom WEB auf das Entity Framework ist nicht gewünscht und muss geändert werden, so dass nur über die CRUD Methoden vom BLL aus auf die Datenbank zugegriffen wird.

Ebenfalls können die Views für das Frontend im WEB automatisch generiert werden. Für alle Models werden alle CRUD-Views benötigt, das heisst jedes Model hat auch eine Create, Read, Edit, Delete View.

Es wurde die Klasse «Account» im Ordner Models im WEB erstellt, für das Login. Über LDAP am Windows AD geschieht die Authentifizierung. Die Autorisierung erfolgt über OWIN Identity.

Für nicht eingeloggte Benutzer dürfen keine Informationen ersichtlich sein. Nicht einmal die Navigation. Deshalb wird der Login Status schon auf der Layout Page berücksichtigt und die Rolle bei den Benutzer Views geprüft. Dies damit wie gefordert nur der Administrator Zugriff auf die Benutzer View hat und es abändern kann.

Die Suchfilter für die Artikel sind auf dem Index der Artikel und auf der Startseite nach dem Einloggen. Dabei gibt es eine Freitext Suche und bestimmte Dropdowns die verwendet werden können zum Suchen, z.b nach Raum sortieren oder nach allen Artikeln während einem Start und Enddatum.

Man kann sich ein Excel erstellen lassen auch mit eingesetzten Suchfiltern, so ist es wesentlich einfacher den Überblick über das ganze Inventar zu halten und es ist auch möglich so Labels im PDF zu erstellen. Dies ist möglich mit iTextSharp, einer .NET library die man per Nugget in das Projekt einbinden kann und das entwickelt wurde für die Generierung von PDFs aus Codes.

Das Projekt wird am Schluss noch ins Verzeichnis «ch.muster.se.inv» auf dem Webserver ehost 51 hochgeladen. Die Website ist dann über <http://inv.se.muster.ch> erreichbar. Der Webserver war bereits eingerichtet.

Datenmodell

Die Datenbank wird auf dem Server ehost41 unter dem Namen ch.muster.se.inv erstellt.

Die Inventarverwaltung benötigt insgesamt sechs Tabellen, dies sind Artikel, Benutzer, FiBuKonto, Kategorie, Raum und Rolle.



Abbildung 1: Datenbankmodell im Visio

Testkonzept

Es wird mit dem folgenden Testkonzept getestet, damit die Arbeit korrekt geprüft wird. Die Tests werden innerhalb der muster ag mit einem Firefox Browser (Version 59.0b11 (64-Bit)) auf die URL <http://inv.se.muster.ch> geprüft. Das Testkonzept wurde Mithilfe des Realisierungskonzepts erstellt. Zu allererst werden die Testfälle erstellt, in der fünften IPERKA Phase «Kontrollieren» werden die Testfälle dann getestet, mit dem Firefox Browser unter Windows 10 Pro (Version 1607).

Testfall-Nr:	1
Anforderung:	Login muss funktionieren
Beschreibung:	Der Benutzer sollte sich mit seinen Zugangsdaten in der Webapplikation anmelden können.
Voraussetzung:	- Die Webapplikation ist am Laufen
Eingabe:	1.) Gültige Zugangsdaten eingeben 2.) Auf den «Anmelden» Button klicken
Erwartetes Resultat:	Der Benutzer sollte eingeloggt sein.

Testfall-Nr:	2
Anforderung:	Logout muss funktionieren
Beschreibung:	Wenn der Benutzer angemeldet ist, muss es ihm möglich sein sich wieder von der Webapplikation abzumelden.
Voraussetzung:	- Die Webapplikation ist am Laufen - Der Benutzer ist angemeldet - Der Benutzer ist auf einer Seite
Eingabe:	1.) Auf den «Abmelden» Button klicken
Erwartetes Resultat:	Der Benutzer sollte ausgeloggt sein.

Testfall-Nr:	3
Anforderung:	Artikel erstellen und speichern
Beschreibung:	Jeder Benutzer muss neue Artikel erstellen und speichern können.
Voraussetzung:	<ul style="list-style-type: none"> - Die Webapplikation ist am Laufen - Der Benutzer ist angemeldet - Der Benutzer ist auf der Index View von den Artikeln
Eingabe:	<ol style="list-style-type: none"> 1.) Auf den Button „Artikel erfassen“ klicken 2.) In die Textfelder: Inventar Nummer, Bezeichnung und Beschreibung «Test 123» eingeben 3.) Bei dem Dropdown Kategorie «Monitore» auswählen 4.) Bei dem Dropdown Raum «Büro Applikationsentwicklung» auswählen 5.) Bei dem Dropdown Nutzer «Alpay Yildirim» auswählen 6.) Als Beschaffungsdatum den «05.05.2015» nehmen 7.) Das Feld Beschaffungswert mit dem Wert «123» ausfüllen 8.) Auf den Button „Erstellen“ klicken
Erwartetes Resultat:	Artikel sollte erstellt und gespeichert sein.

Testfall-Nr:	4
Anforderung:	Artikel bearbeiten und speichern
Beschreibung:	Jeder Benutzer muss vorhandene Artikel bearbeiten und speichern können.
Voraussetzung:	<ul style="list-style-type: none"> - Die Webapplikation ist am Laufen - Der Benutzer ist angemeldet - Der Benutzer ist auf der Index View von den Artikeln
Eingabe:	<ol style="list-style-type: none"> 1.) Auf den Button „Bearbeiten“ klicken in der Liste beim Eintrag „Test“. 2.) In die Textfelder: Inventar Nummer, Bezeichnung und Beschreibung «Test 456» eingeben 3.) Bei dem Dropdown Kategorie «Drucker» auswählen 4.) Bei dem Dropdown Raum «Büro Administration» auswählen 5.) Bei dem Dropdown Nutzer «Hans Mustermann» auswählen 6.) Als Beschaffungsdatum den «14.12.2016» nehmen 7.) Das Feld Beschaffungswert mit dem Wert «478» ausfüllen 8.) Auf den Button „Speichern“ klicken
Erwartetes Resultat:	Artikel sollte bearbeitet und gespeichert sein.

Testfall-Nr:	5
Anforderung:	Artikel löschen
Beschreibung:	Jeder Benutzer muss vorhandene Artikel löschen können.
Voraussetzung:	<ul style="list-style-type: none"> - Die Webapplikation ist am Laufen - Der Benutzer ist angemeldet - Der Benutzer ist auf der Index View von den Artikeln
Eingabe:	1.) Auf den Button „Löschen“ klicken in der Liste beim Eintrag „Test 456“. 2.) Auf den Button „Löschen“ klicken
Erwartetes Resultat:	Artikel sollte gelöscht sein.

Testfall-Nr:	6
Anforderung:	Kategorie erstellen und speichern
Beschreibung:	Jeder Benutzer muss neue Kategorien erstellen und speichern können.
Voraussetzung:	<ul style="list-style-type: none"> - Die Webapplikation ist am Laufen - Der Benutzer ist angemeldet - Der Benutzer ist auf der Index View von den Kategorien
Eingabe:	1.) Auf den Button „Kategorie erfassen“ klicken 2.) In das Textfeld Bezeichnung «Test 123» eingeben 3.) Bei dem Dropdown FiBu Konto «1510» auswählen 4.) Auf den Button „Erstellen“ klicken
Erwartetes Resultat:	Kategorie sollte erstellt und gespeichert sein.

Testfall-Nr:	7
Anforderung:	Kategorie bearbeiten und speichern
Beschreibung:	Jeder Benutzer muss vorhandene Kategorien bearbeiten und speichern können.
Voraussetzung:	<ul style="list-style-type: none"> - Die Webapplikation ist am Laufen - Der Benutzer ist angemeldet - Der Benutzer ist auf der Index View von den Kategorien
Eingabe:	1.) Auf den Button „Bearbeiten“ klicken in der Liste beim Eintrag „Test 123“. 2.) In das Textfeld Bezeichnung «Test 456» eingeben 3.) Bei dem Dropdown FiBu Konto «1520» auswählen 4.) Auf den Button „Speichern“ klicken
Erwartetes Resultat:	Kategorie sollte bearbeitet und gespeichert sein.

Testfall-Nr:	8
Anforderung:	Kategorie löschen
Beschreibung:	Jeder Benutzer muss vorhandene Kategorien löschen können.
Voraussetzung:	<ul style="list-style-type: none"> - Die Webapplikation ist am Laufen - Der Benutzer ist angemeldet - Der Benutzer ist auf der Index View von den Kategorien
Eingabe:	1.) Auf den Button „Löschen“ klicken in der Liste beim Eintrag „Test 456“. 2.) Auf den Button „Löschen“ klicken
Erwartetes Resultat:	Kategorie sollte gelöscht sein.

Testfall-Nr:	9
Anforderung:	Raum erstellen und speichern
Beschreibung:	Jeder Benutzer muss neue Räume erstellen und speichern können.
Voraussetzung:	<ul style="list-style-type: none"> - Die Webapplikation ist am Laufen - Der Benutzer ist angemeldet - Der Benutzer ist auf der Index View von den Räume
Eingabe:	1.) Auf den Button „Raum erfassen“ klicken 2.) In das Textfeld Bezeichnung «Test 123» eingeben 3.) Bei dem Dropdown Lokalisierung «UG» auswählen 4.) Auf den Button „Erstellen“ klicken
Erwartetes Resultat:	Raum sollte erstellt und gespeichert sein.

Testfall-Nr:	10
Anforderung:	Raum bearbeiten und speichern
Beschreibung:	Jeder Benutzer muss vorhandene Räume bearbeiten und speichern können.
Voraussetzung:	<ul style="list-style-type: none"> - Die Webapplikation ist am Laufen - Der Benutzer ist angemeldet - Der Benutzer ist auf der Index View von den Räume
Eingabe:	1.) Auf den Button „Bearbeiten“ klicken in der Liste beim Eintrag „Test 123“. 2.) In das Textfeld Bezeichnung «Test 456» eingeben 3.) Bei dem Dropdown Lokalisierung «OG» auswählen 4.) Auf den Button „Speichern“ klicken
Erwartetes Resultat:	Raum sollte bearbeitet und gespeichert sein.

Testfall-Nr:	11
Anforderung:	Raum löschen
Beschreibung:	Jeder Benutzer muss vorhandene Räume löschen können.
Voraussetzung:	<ul style="list-style-type: none"> - Die Webapplikation ist am Laufen - Der Benutzer ist angemeldet - Der Benutzer ist auf der Index View von den Räumen
Eingabe:	1.) Auf den Button „Löschen“ klicken in der Liste beim Eintrag „Test 456“. 2.) Auf den Button „Löschen“ klicken
Erwartetes Resultat:	Raum sollte gelöscht sein.

Testfall-Nr:	12
Anforderung:	Benutzer erstellen und speichern
Beschreibung:	Jeder Benutzer mit der Rolle „Administrator“ muss neue Benutzer erstellen und speichern können.
Voraussetzung:	<ul style="list-style-type: none"> - Die Webapplikation ist am Laufen - Der Benutzer ist angemeldet - Der Benutzer hat die Rolle „Administrator“ - Der Benutzer ist auf der Index View von den Benutzern
Eingabe:	1.) Auf den Button „Benutzer erfassen“ klicken 2.) In die Textfelder: Benutzername, Vorname, Name, E-Mail und Telefon Intern «Test 123» eingeben 3.) Bei dem Dropdown Rolle «Benutzer» auswählen 4.) Auf den Button „Erstellen“ klicken
Erwartetes Resultat:	Benutzer sollte erstellt und gespeichert sein.

Testfall-Nr:	13
Anforderung:	Benutzer bearbeiten und speichern
Beschreibung:	Jeder Benutzer mit der Rolle „Administrator“ muss vorhandene Artikel bearbeiten und speichern können.
Voraussetzung:	<ul style="list-style-type: none"> - Die Webapplikation ist am Laufen - Der Benutzer ist angemeldet - Der Benutzer hat die Rolle „Administrator“ - Der Benutzer ist auf der Index View von den Benutzern
Eingabe:	<ol style="list-style-type: none"> 1.) Auf den Button „Bearbeiten“ klicken in der Liste beim Eintrag „Test 123“. 2.) In die Textfelder: Benutzername, Vorname, Name, E-Mail und Telefon Intern «Test 456» eingeben 3.) Bei dem Dropdown Rolle «Administrator» auswählen 4.) Auf den Button „Speichern“ klicken
Erwartetes Resultat:	Benutzer sollte bearbeitet und gespeichert sein.

Testfall-Nr:	14
Anforderung:	Benutzer löschen
Beschreibung:	Jeder Benutzer mit der Rolle „Administrator“ muss vorhandene Benutzer löschen können.
Voraussetzung:	<ul style="list-style-type: none"> - Die Webapplikation ist am Laufen - Der Benutzer ist angemeldet - Der Benutzer hat die Rolle „Administrator“ - Der Benutzer ist auf der Index View von den Benutzern
Eingabe:	<ol style="list-style-type: none"> 1.) Auf den Button „Löschen“ klicken in der Liste beim Eintrag „Test 456“. 2.) Auf den Button „Löschen“ klicken
Erwartetes Resultat:	Benutzer sollte gelöscht sein.

Testfall-Nr:	15
Anforderung:	FiBu Konto erstellen und speichern
Beschreibung:	Jeder Benutzer muss neue FiBu Konten erstellen und speichern können.
Voraussetzung:	<ul style="list-style-type: none"> - Die Webapplikation ist am Laufen - Der Benutzer ist angemeldet - Der Benutzer ist auf der Index View von den FiBu Konten
Eingabe:	1.) Auf den Button „FiBu Konto erfassen“ klicken 2.) In die Textfelder: FiBu Konto und Bezeichnung «Test 123» eingeben 3.) Auf den Button „Erstellen“ klicken
Erwartetes Resultat:	FiBu Konto sollte erstellt und gespeichert sein.

Testfall-Nr:	16
Anforderung:	FiBu Konto bearbeiten und speichern
Beschreibung:	Jeder Benutzer muss vorhandene FiBu Konten bearbeiten und speichern können.
Voraussetzung:	<ul style="list-style-type: none"> - Die Webapplikation ist am Laufen - Der Benutzer ist angemeldet - Der Benutzer ist auf der Index View von den FiBu Konten
Eingabe:	1.) Auf den Button „Bearbeiten“ klicken in der Liste beim Eintrag „Test 123“. 2.) In die Textfelder: FiBu Konto und Bezeichnung «Test 456» eingeben 3.) Auf den Button „Speichern“ klicken
Erwartetes Resultat:	FiBu Konto sollte bearbeitet und gespeichert sein.

Testfall-Nr:	17
Anforderung:	FiBu Konto löschen
Beschreibung:	Jeder Benutzer muss vorhandene FiBu Konten löschen können.
Voraussetzung:	<ul style="list-style-type: none"> - Die Webapplikation ist am Laufen - Der Benutzer ist angemeldet - Der Benutzer ist auf der Index View von den FiBu Konten
Eingabe:	1.) Auf den Button „Löschen“ klicken in der Liste beim Eintrag „Test 456“. 2.) Auf den Button „Löschen“ klicken
Erwartetes Resultat:	FiBu Konto sollte gelöscht sein.

Testfall-Nr:	18
Anforderung:	Unberechtigt auf die Index Seite von «Benutzer» zugreifen
Beschreibung:	Ein normaler Benutzer ohne die Rolle „Administrator“ kann nicht auf die Seiten gelangen die mit dem Erstellen, Bearbeiten und Löschen der Benutzer zu tun haben.
Voraussetzung:	<ul style="list-style-type: none"> - Die Webapplikation ist am Laufen - Der Benutzer ist angemeldet - Der Benutzer hat die Rolle „Benutzer“ - Der Benutzer ist auf der Index View von den FiBu Konten
Eingabe:	1.) Auf die Index Seite von Benutzer gehen
Erwartetes Resultat:	Es sollte nicht möglich sein.

Testfall-Nr:	19
Anforderung:	Suchfilter funktioniert
Beschreibung:	Jeder Benutzer muss Artikel suchen können.
Voraussetzung:	<ul style="list-style-type: none"> - Die Webapplikation ist am Laufen - Der Benutzer ist angemeldet - Der Benutzer ist auf der Index View von den Artikeln
Eingabe:	<ol style="list-style-type: none"> 1.) In das Textfeld Suche «Test 123» eingeben 2.) Bei dem Dropdown Kategorie «Monitore» auswählen 3.) Bei dem Dropdown Raum «Büro Applikationsentwicklung» auswählen 4.) Bei dem Dropdown Nutzer «Alpay Yildirim» auswählen 5.) Bei dem Dropdown FiBu Konto «1510» auswählen 6.) Bei dem Dropdown Von «2017» auswählen 7.) Bei dem Dropdown Bis «2018» auswählen 8.) Auf den Button „Suchen“ klicken
Erwartetes Resultat:	Es sollte ein Resultat zurückkommen, mit den jeweiligen Artikeln die auf diese Suchkriterien zutreffen.

Testfall-Nr:	20
Anforderung:	Labels der Artikel per PDF exportieren
Beschreibung:	Jeder Benutzer muss Labels der Artikel per PDF exportieren können.
Voraussetzung:	<ul style="list-style-type: none"> - Die Webapplikation ist am Laufen - Der Benutzer ist angemeldet - Der Benutzer ist auf der Index View von den Artikeln
Eingabe:	1.) Auf den Button „Labels erstellen in PDF“ klicken.
Erwartetes Resultat:	Die Artikel sollten nun als PDF-Datei zurückkommen.

Testfall-Nr:	21
Anforderung:	Artikel per Excel exportieren
Beschreibung:	Jeder Benutzer muss Artikel per Excel exportieren können.
Voraussetzung:	<ul style="list-style-type: none"> - Die Webapplikation ist am Laufen - Der Benutzer ist angemeldet - Der Benutzer ist auf der Index View von den Artikeln
Eingabe:	1.) Auf den Button „Artikel exportieren in Excel“ klicken.
Erwartetes Resultat:	Die Artikel sollten nun als Excel-Datei zurückkommen.

Sitemap

Es wurden die folgenden Sitemaps erstellt, die beim Realisieren einen guten Überblick verschaffen sollen. Die Sitemap zeigt die logische Struktur der Webapplikation auf.

- Account
 - Login (Der Besucher kommt auf diese Seite, falls er nicht eingeloggt ist)
- Artikel
 - Index (Liste aller Artikel, inklusive Suchfiltern)
 - Create (Artikel erstellen)
 - Delete (Artikel löschen)
 - Details (Artikel ansehen)
 - Edit (Artikel editieren)
- Benutzer (Nur für Benutzer mit der Rolle «Administrator» zugänglich)
 - Index (Liste aller Benutzer)
 - Create (Benutzer erstellen)
 - Delete (Benutzer löschen)
 - Details (Benutzer ansehen)
 - Edit (Benutzer editieren)
- FiBuKonto
 - Index (Liste aller FiBu Konten)
 - Create (FiBu Konto erstellen)
 - Delete (FiBu Konto löschen)
 - Details (FiBu Konto ansehen)
 - Edit (FiBu Konto editieren)
- Kategorie
 - Index (Liste aller Kategorien)
 - Create (Kategorie erstellen)
 - Delete (Kategorie löschen)
 - Details (Kategorie ansehen)
 - Edit (Kategorie editieren)
- Raum
 - Index (Liste aller Räume)
 - Create (Raum erstellen)
 - Delete (Raum löschen)
 - Details (Raum ansehen)
 - Edit (Raum editieren)

Entscheiden

Das Entscheiden ist die dritte Phase der IPERKA-Methode.

In dieser Phase wird entschieden welche Lösungsvarianten am sinnvollsten sind und geprüft ob sie in dieser Form umgesetzt werden können.

Varianten

Wie es im Realisierungskonzept ersichtlich ist, gibt es mehrere Möglichkeiten dem Nutzer in der Applikation eine Rolle zuzuteilen. Zum einen könnte man die Rollenverteilung hart codiert im Code implementieren. Man könnte aber auch die Rolle in der Datenbank festlegen, wenn man es als einen Fremdschlüssel in der Benutzer Tabelle mit dazu speichert.

Eine andere Entscheidung die noch getroffen werden muss, ist die Position des Logout Buttons. Die erste Lösungsvariante wäre den Logout Button oben rechts zu platzieren, dies wäre auch die handelsübliche Variante. Da wir aber die Webapplikation nur intern benutzen, könnte man den Logout Button auch ganz weglassen, dies wäre dann die zweite Lösungsvariante.

Entscheid

Bei den Rollen habe ich mich für die zweite Lösungsvariante mit der Datenbank entschieden, da dies die elegantere Methode ist und auch dem üblichen Vorgehen entspricht, der einzige Vorteil der ersten Variante wäre gewesen, dass sie schnell implementiert werden kann. Die zweite Variante ist die geeignetere für dieses Projekt und auch wesentlich sauberer.

Beim Logout Button habe ich mich entschieden ihn doch zu implementieren, weil der Benutzer sich auch ausloggen können soll. Dies ist notwendig, falls der Benutzer von einem fremden PC, zum Beispiel von zuhause auf das Inventar zugreifen muss und temporär abwesend vom PC ist. Der Benutzer muss imstande sein sich abmelden zu können, deswegen wird der Logout Button implementiert und ich entscheide mich hierbei für die erste Lösungsvariante.

Es ist möglich das Projekt im geplanten Zeitraum zu realisieren. So wie in der Phase «Planen» beschrieben wird das Projekt realisiert.

Realisieren

Das Realisieren ist die vierte Phase der IPERKA-Methode.

Elemente die wichtig sind werden hier erklärt und dargestellt, diese Phase ist hauptsächlich für die Dokumentation aller Programmierarbeiten. Im Code sind auch Kommentare vorhanden, wenn diese bei der Verständlichkeit mithelfen.

Projektumgebung

Wie in der Planung beschrieben ist die Webapplikation in drei Teilprojekte unterteilt, welche mithilfe von Dependency Injection über Interfaces verknüpft sind. Somit können sie, wenn nötig auch leicht ausgetauscht werden.

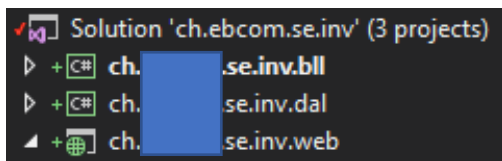


Abbildung 2: Die drei Layers (Schichten) im Visual Studio

Der Business Logic Layer und das Data Access Layer wurden als leere Class Library angelegt. Der Presentation Layer wurde als ASP.NET Web Application MVC ohne Authentifizierung erstellt.

Datenbank

Unter dem Namen ch.muster.se.inv wurde die Datenbank auf dem ehost41 angelegt. Wie im Datenmodell ersichtlich, enthält sie sechs Tabellen.

Die Option Identity Specification ist für die Primary Keys aktiviert. Die ID wird dadurch automatisch generiert und deswegen muss man sich nicht in der Programmierung darum kümmern.

Im Entity Framework wurden die Verknüpfungen zwischen den Tabellen erstellt.

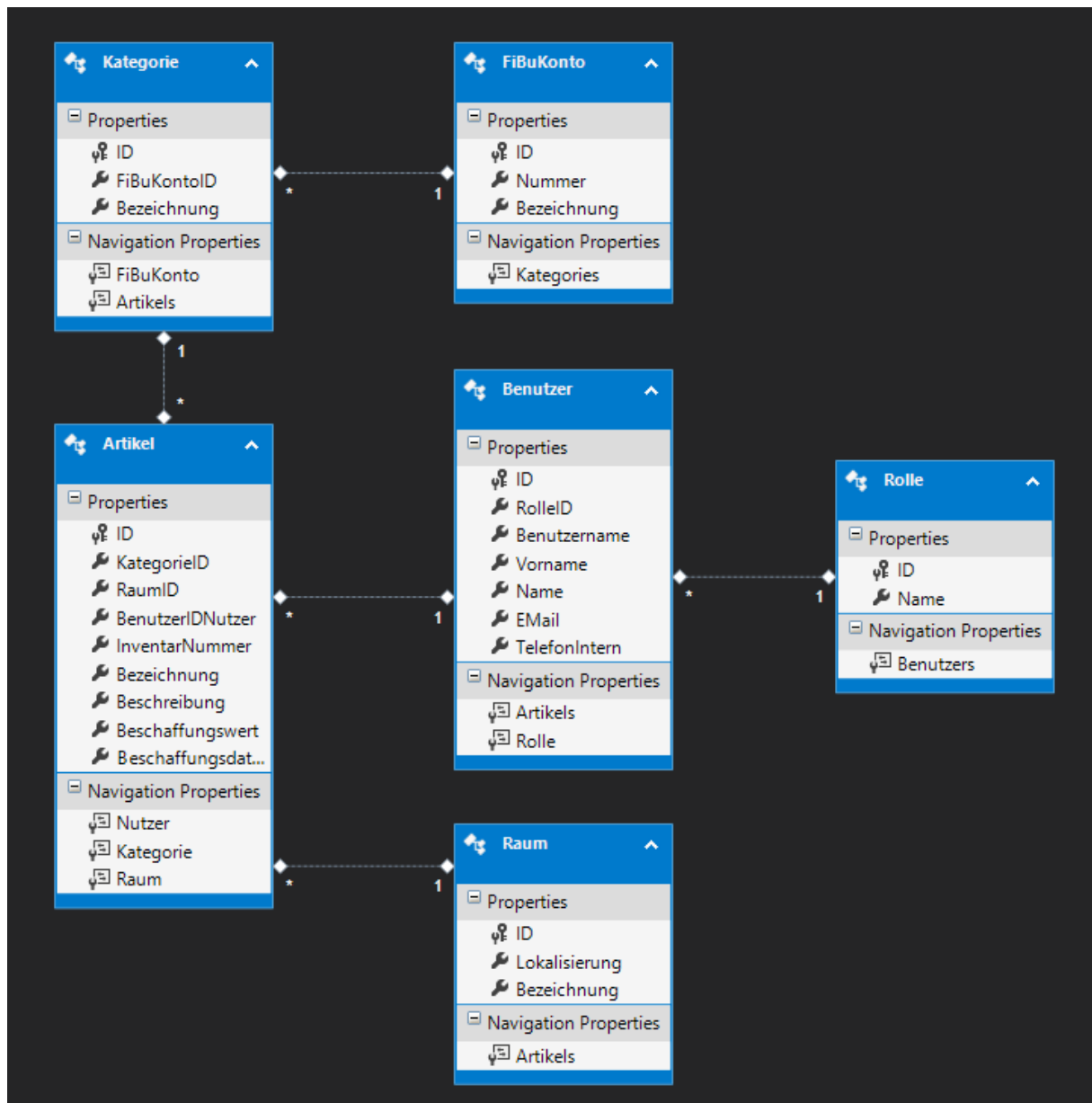


Abbildung 3: Datenbank Diagramm im Visual Studio

Entity Framework

Das ADO.NET Entity Data Model wurde im Ordner «Models» im Data Access Layer erstellt. Die Verbindungen zwischen den Tabellen wurden so wie im Datenmodell angegeben angelegt. Nur der Connection String muss hier zusätzlich von der App.config im Data Access Layer in die Web.config im Presentation Layer eingefügt werden.

Partial Models

Partial Models ergänzen die Models, die vom Entity Framework erstellt wurden. Dies hat den grossen Vorteil, dass bestimmte zusätzliche Attribute definiert werden können und dass nach einer Änderung in der Datenbank nicht wieder alles neu in der vom Entity Framework generierten Model definiert werden muss.

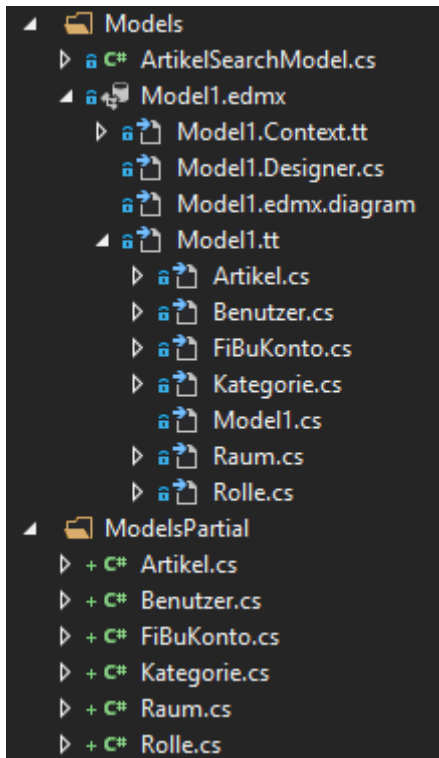


Abbildung 4: Ordnerstruktur der Models und der ModelsPartial

In einem Partial Model können viele Eigenschaften zusätzlich definiert werden, z.b der Preis. Hier wird der Preis schon richtig formatiert als String zurückgegeben. In der Klasse «ArtikelMeta» sind Meta Tags in der die Eigenschaften für die jeweiligen Properties des Modells definiert sind.

```
namespace ch.muster.se.inv.dal.Models
{
    // Die Metadaten der jeweiligen Properties in Artikel sind in der Klasse
    ArtikelMeta angegeben.
    // Zusätzliche Properties für Artikel wie z.b Preis werden hier angegeben und
    nicht in der Datenbank.
    [MetadataType(typeof(ArtikelMeta))]
    public partial class Artikel
    {
        public string Preis
        {
            get
            {
                return Beschaffungswert.ToString("N");
            }
        }

        [Display(Name = "Artikel")]
        public string Gegenstand
        {

```

```

        get
        {
            return Bezeichnung + " (" + Beschreibung + ")";
        }
    }

    [Display(Name = "Raum")]
    public string Ort
    {
        get
        {
            return Raum.Lokalisierung + " - " + Raum.Bezeichnung;
        }
    }
}

public class ArtikelMeta
{
    [Required]
    [Display(Name = "Kategorie")]
    public int KategorieID { get; set; }

    [Required]
    [Display(Name = "Raum")]
    public int RaumID { get; set; }

    [Required]
    [Display(Name = "Nutzer")]
    public int BenutzerIDNutzer { get; set; }

    [Required]
    [Display(Name = "Inventar Nummer")]
    public string InventarNummer { get; set; }

    [Required]
    [Display(Name = "Bezeichnung")]
    public string Bezeichnung { get; set; }

    [Display(Name = "Beschreibung")]
    public string Beschreibung { get; set; }

    [Required]
    [DataType(DataType.Currency)]
    [Display(Name = "Beschaffungswert (in CHF)")]
    public Nullable Beschaffungswert { get; set; }

    [Display(Name = "Beschaffungsdatum")]
    [Required]
    [DataType(DataType.Date)]
    [DisplayFormat(DataFormatString = "{0:dd.MM.yyyy}")]
    public Nullable<System.DateTime> Beschaffungsdatum { get; set; }
}
}

```

Mit den UIHints können z.B. die Anzeigenamen verändert werden, die dann in der View mit diesem Namen dargestellt werden. Properties können mit dem Tag Required eine zusätzliche Fehlermeldung generieren, falls versucht wird einen Artikel zu erstellen ohne einen Namen und der Datentyp und dessen Formatierung kann auch angegeben werden wie man beim Beschaffungsdatum sieht mit [DataType(DataType.Date)] und [DisplayFormat(DataFormatString = "{0:dd.MM.yyyy}")].

Dependency Injection

Mit der Dependency Injection wurden die Teilprojekte untereinander referenziert. Das NuGet Paket Unity.Mvc5 wurde installiert und im Global.asax wurde mit `UnityConfig.RegisterComponents();` die Methode `RegisterComponents` im dazu generierten File `App_Start/Unityconfig.cs` aufgerufen.

Nun müssen die folgenden Klassen und Interfaces im Business Logic Layer und Data Access Layer angelegt werden, damit die Dependency Injection funktioniert.

- Business Logic Layer
 - Services
 - ArtikelService
 - BenutzeroService
 - FiBuKontoService
 - KategorieService
 - RaumService
 - RolleService
 - Interfaces
 - IArtikelService
 - IBenutzeroService
 - IFiBuKontoService
 - IKategorieService
 - IRaumService
 - IRolleService
- Data Access Layer
 - Accessors
 - ArtikelAccessor
 - BenutzerAccessor
 - FiBuKontoAccessor
 - KategorieAccessor
 - RaumAccessor
 - RolleAccessor
 - Interfaces
 - IArtikelAccessor
 - IBenutzerAccessor
 - IFiBuKontoAccessor
 - IKategorieAccessor
 - IRaumAccessor
 - IRolleAccessor

Das dazugehörige Interface wird immer von den jeweiligen Services und Accessors implementiert.

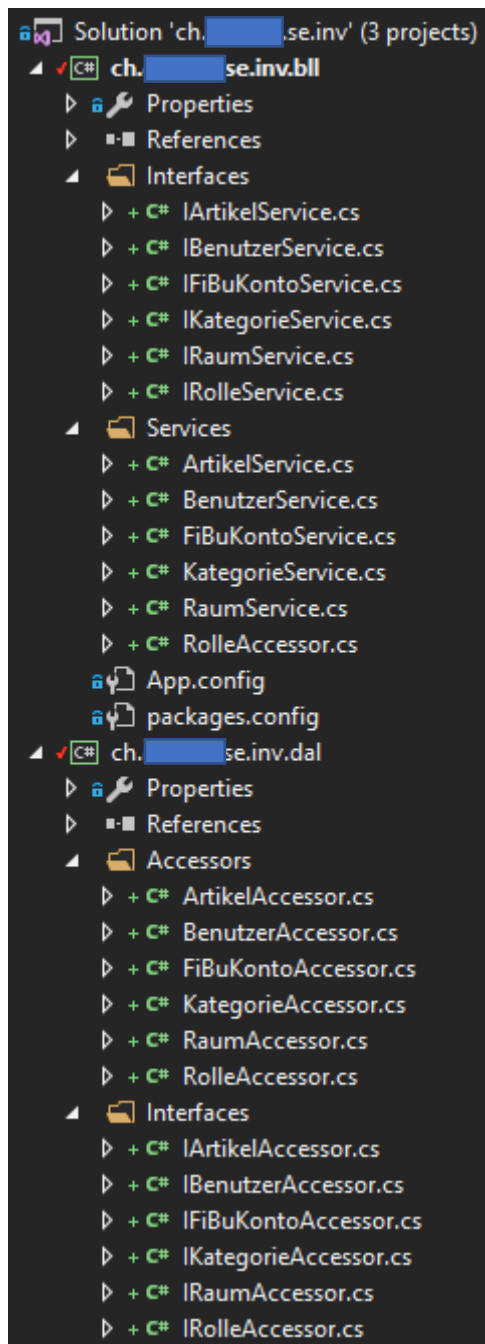


Abbildung 5: Ordnerstruktur Services & Accessors

Die Ordnerstruktur für die Accessors und Services mit ihren Interfaces sieht dann insgesamt so aus, wie auf der Abbildung.


```
namespace ch.muster.se.inv.web
{
    public static class UnityConfig
    {
        public static void RegisterComponents()
        {
            var container = new UnityContainer();

            // DAL
            container.RegisterType<IArtikelAccessor, ArtikelAccessor>();
            container.RegisterType<IBenutzerAccessor, BenutzerAccessor>();
            container.RegisterType<IFiBuKontoAccessor, FiBuKontoAccessor>();
            container.RegisterType<IKategorieAccessor, KategorieAccessor>();
            container.RegisterType<IRaumAccessor, RaumAccessor>();
            container.RegisterType<IRolleAccessor, RolleAccessor>();

            // BLL
            container.RegisterType<IArtikelService, ArtikelService>();
            container.RegisterType<IBenutzerService, BenutzerService>();
            container.RegisterType<IFiBuKontoService, FiBuKontoService>();
            container.RegisterType<IKategorieService, KategorieService>();
            container.RegisterType<IRaumService, RaumService>();
            container.RegisterType<IRolleService, RolleService>();

            DependencyResolver.SetResolver(new UnityDependencyResolver(container));
        }
    }
}
```

Die Komponenten werden dann unter dem Verzeichnis «App_Start» in der «UnityConfig.cs» erfasst. Die Registrierung der Komponenten ist notwendig, damit die Dependency Injection funktioniert.

Authentifizierung & Autorisierung

Das Login wurde mit OWIN und LDAP umgesetzt.

OWIN ist auch dafür zuständig den User nach einer angegebenen Zeit an Inaktivität auszuloggen, dies ist auch gut um unbefugten Zugriff zu vermeiden. Für das Projekt habe ich dazu eine Zeit von 30 Minuten ausgewählt.

Im Presentation Layer musste für OWIN eine Startup Klasse erstellt werden.

```
namespace ch. [REDACTED] se.inv.web
{
    0 references
    public class Startup
    {
        0 references | 0 exceptions
        public void Configuration(IApplicationBuilder app)
        {
            app.UseCookieAuthentication(new CookieAuthenticationOptions
            {
                AuthenticationType = "ApplicationCookie",
                LoginPath = new PathString("/Account/Login"),

                // Wenn der User 30 Minuten Inaktiv ist, wird er automatisch ausgeloggt
                ExpireTimeSpan = new TimeSpan(0, 30, 0),
                SlidingExpiration = true
            });

            // Diese Codezeile wurde aus dem File kopiert das in der Dokumentation angegebenen ist
            AntiForgeryConfig.UniqueClaimTypeIdentifier = ClaimTypes.NameIdentifier;
        }
    }
}
```

Abbildung 6: Startup.cs OWIN Konfiguration

Die Parameter werden in diesem Codeabschnitt mit dem Objekt «CookieAuthenticationOptions» für das Login festgelegt. Die Zeit kann mit ExpireTimeSpan angegeben werden, die dann bestimmt ab wann man automatisch ausgeloggt werden soll. Wenn man SlidingExpiration auf true setzt, ist das automatische Ausloggen nach einer bestimmten Zeit ohne Aktivität aktiviert.

Wenn SlidingExpiration auf false gesetzt wäre würde man schon nach der bestimmten Zeit nach dem Einloggen ausgeloggt werden, da es die Zeit nicht zurücksetzen würde, bei Aktivität des Benutzers.

In der Filterconfig.cs, die sich im Verzeichnis «App_Start» befindet, muss festgelegt werden dass alle Controller nur erreichbar sein dürfen, wenn man eingeloggt ist.

```
public class FilterConfig
{
    1 reference | 0 exceptions
    public static void RegisterGlobalFilters(GlobalFilterCollection filters)
    {
        filters.Add(new HandleErrorAttribute());
        // Mit diesem Filter wird angegeben das alle Controller nur im eingeloggten Zustand zugreifbar sein dürfen.
        filters.Add(new AuthorizeAttribute());
    }
}
```

Abbildung 7: FilterConfig.cs, festlegen des Filters für die Autorisierung

Im Verzeichnis «Controller» wird der AccountController erstellt, da die View «Login» die sich im Verzeichnis «Account» befindet für jeden Benutzer auch ohne eingeloggten Status zugänglich sein soll, wird das Attribut [AllowAnonymous] hinzugefügt. Das [AllowAnonymous] Attribut ermöglicht den Zugriff auf die Login View, ohne dass man eingeloggt ist. Im AccountController sind die Methoden fürs Login per LDAP und für das Logout, die Authentifizierung per LDAP geschieht im Account Model. Der Benutzer kann sich nur anmelden, wenn er in der Datenbank erfasst ist und auch im Active Directory erfasst ist, Die Rolle des Users ist in der Datenbank definiert und wird in einem Claim gespeichert, so dass die Rolle des Nutzers über die ganze Applikation hinweg bekannt ist, bis zum Logout.

```
[HttpPost]
public ActionResult Login(Account model)
{
    if (!ModelState.IsValid)
    {
        return View(model);
    }

    // Es wird hier überprüft ob die Login Daten korrekt sind
    if (model.Login())
    {
        // Hier wird abgefragt ob es in der Benutzer Tabelle schon einen Benutzer
        // gibt mit dem jeweiligen Benutzernamen, falls nicht, wird ein null
        // zurückgegeben
        Benutzer benutzer =
            _benutzerService.GetBenutzerByBenutzername(model.Username);

        // Wenn der Benutzer nicht in der Datenbank vorhanden ist, wird er auf
        // das hingewiesen
        if (benutzer == null)
        {
            // Der Benutzer wird darauf hingewiesen, das er einen Account
            // braucht
            ModelState.AddModelError(string.Empty, "Login fehlgeschlagen,
            bitte überprüfen Sie ihre Eingabe.");

            return View(model);
        }

        var identity = new ClaimsIdentity(new[] {
            new Claim(ClaimTypes.Name, model.FullName),
            new Claim(ClaimTypes.NameIdentifier, model.Username),
            new Claim(ClaimTypes.Role, benutzer.Rolle.Name) // Die Rolle wird
            hier in der Identity gespeichert, vorübergehend bis die Sitzung beendet ist
        }, "ApplicationCookie");

        var ctx = Request.GetOwinContext();
        var authManager = ctx.Authentication;

        authManager.SignIn(identity);

        return Redirect(GetRedirectUrl(model.ReturnUrl));
    }

    ModelState.AddModelError("", "Ungültige Benutzerdaten");
    return View(model);
}
```

Im Presentation Layer befindet sich im Verzeichnis «Models» auch das Account Model. Dieser hat die Methode Login, die die Kommunikation mit dem Active Directory per LDAP regelt. Im Account Model sind die Properties Username, Password und FullName definiert. Wie schon erwähnt geschieht hier der eigentliche Zugriff auf die Active Directory per LDAP, beim strADPath ist der Pfad zum Active Directory definiert.

```
public class Account
{
    [Required]
    [Display(Name = "Benutzername")]
    7 references | 0 exceptions
    public string Username { get; set; }

    [Required]
    [Display(Name = "Passwort")]
    [DataType(DataType.Password)]
    4 references | 0 exceptions
    public string Password { get; set; }

    [HiddenInput]
    2 references | 0 exceptions
    public string FullName { get; set; }
}
```

Abbildung 8: Account.cs mit der Login Methode

Layout

Die Layout Page befindet sich im Presentation Layer unter dem Verzeichnis «Shared». Die Layout Seite ist zuständig für das einheitliche Design der Applikation. Der Header und der Footer sind im Layout File definiert. Im Header ist die Navigation, damit die Navigation nur im eingeloggten Zustand ersichtlich ist, habe ich dafür eine Partial View «Header» erstellt. Der Header wird nur angezeigt, wenn der Benutzer eingeloggt ist. Wenn der Benutzer nicht eingeloggt ist, werden die Items in der Navigation nicht angezeigt. Hierbei habe ich zusätzlich beachtet, dass das Projekt im responsive Webdesign dargestellt wird.

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>@ViewBag.Title - Inventar - muster ag</title>
  @Styles.Render("~/Content/css")
  @Scripts.Render("~/bundles/modernizr")
  @Scripts.Render("~/bundles/jquery")
</head>
<body>
  <div class="navbar navbar-inverse navbar-fixed-top">
    <div class="container">
      <div class="navbar-header">
        <button type="button" class="navbar-toggle" data-toggle="collapse"
data-target=".navbar-collapse">
          <span class="icon-bar"></span>
          <span class="icon-bar"></span>
          <span class="icon-bar"></span>
        </button>
        <span class="navbar-brand nav-title">Inventar</span>
      </div>
      @Html.Partial("Header")
    </div>
  </div>
  <div class="container body-content">
    @RenderBody()
    <hr />
    <footer>
      <p>&copy; @DateTime.Now.Year - muster ag - Inventar</p>
    </footer>
  </div>

  @Scripts.Render("~/bundles/bootstrap")
  @RenderSection("scripts", required: false)
</body>
</html>
```

Mit der `@Html.Partial("Header")` Methode wird der Header ins Layout File miteingebunden.

```

@using ch.muster.se.inv.web.Utilities

@if (User.Identity.IsAuthenticated)
{
    @* Der Benutzer muss eingeloggt sein, damit die Navigation angezeigt wird. *@
    <div class="navbar-collapse collapse">
        <ul class="nav navbar-nav">
            <li class="@Html.IsActive("Artikel")">@Html.ActionLink("Artikel", "Index",
"Artikel")</li>
            <li class="@Html.IsActive("Kategorie")">@Html.ActionLink("Kategorien",
"Index", "Kategorie")</li>
            <li class="@Html.IsActive("Raum")">@Html.ActionLink("Räume", "Index",
"Raum")</li>
            @if (this.User.IsInRole("Administrator"))
            {
                <li class="@Html.IsActive("Benutzer")">@Html.ActionLink("Benutzer",
"Index", "Benutzer")</li>
            }
            <li class="@Html.IsActive("FiBuKonto")">@Html.ActionLink("FiBu Konten",
"Index", "FiBuKonto")</li>
        </ul>
        <ul class="nav navbar-nav navbar-right">
            <li><a class="noLink">@User.Identity.Name</a></li>
            <li>@Html.ActionLink("Logout", "Logout", "Account")</li>
        </ul>
    </div>
}
else
{
    <div class="navbar-collapse collapse">
        @*<ul class="nav navbar-nav navbar-right">
            <li>@Html.ActionLink("Login", "Login", "Account")</li>
        </ul>*@
    </div>
}

<style>
    a.noLink:hover {
        color: #999999 !important;
    }
</style>

```

Utilities

Im Header.cshtml sieht man die Logik hinter der Navigation. Mit der `@if (User.Identity.IsAuthenticated)` Abfrage wird geprüft ob der Benutzer eingeloggt ist. Nebenbei wurde noch zusätzlich eine Methode `IsActive()` in der Klasse «Utilities» unter dem Verzeichnies «Utilities» geschrieben, sie prüft ob man auf der Seite des jeweiligen Navigationspunktes ist und gibt dem Navigationsitem im HTML die Klasse «active». Dies wurde mit einem Lösungsvorschlag auf Stack Overflow gelöst, der Link ist in den Quellen.

```
namespace ch.muster.se.inv.web.Utilities
{
    public static class Utilities
    {
        ...
        public static string IsActive(this HtmlHelper html, string control)
        {
            var routeData = html.ViewContext.RouteData;

            var routeControl = (string)routeData.Values["controller"];

            var returnActive = control == routeControl;

            return returnActive ? "active" : "";
        }
    }
}
```

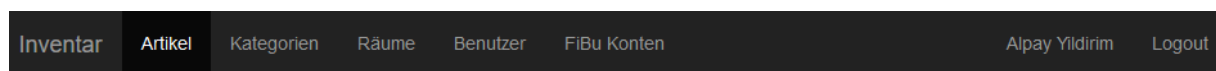


Abbildung 9: Navigation mit aktivem Item «Artikel» mit der Rolle «Administrator»

Artikel

Im Business Logic Layer gibt es den `ArtikelService` und im Data Access Layer ist der `ArtikelAccessor`, dabei wurden beide nach dem CRUD Prinzip aufgebaut.

```
public interface IArtikelService
{
    //Create
    Artikel CreateArtikel(Artikel Artikel);

    //Read
    Artikel GetArtikel(int? id);
    IEnumerable<Artikel> GetArtikels();
    //Update
    Artikel EditArtikel(Artikel Artikel);

    //Delete
    void DeleteArtikel(int? id);
}
```

Im `ArtikelService` sind die Methoden implementiert, mit `GetArtikels()` bekommt man alle Artikel. Bei `GetArtikel(id)` bekommt man den jeweiligen Artikel mit der ID. Mit `CreateArtikel` kann man Artikel erstellen und mit `EditArtikel`, den jeweiligen Artikel bearbeiten. `DeleteArtikel()` ist wie der Name schon sagt zum Löschen von Artikeln.

```
public class ArtikelService : IArtikelService
{
    private ArtikelAccessor _artikelAccessor;
```

```

public ArtikelService()
{
    _artikelAccessor = new ArtikelAccessor();
}

public IEnumerable<Artikel> GetArtikels()
{
    return _artikelAccessor.GetArtikels();
}

public Artikel GetArtikel(int? id)
{
    return _artikelAccessor.GetArtikel(id);
}

public Artikel CreateArtikel(Artikel artikel)
{
    return _artikelAccessor.SaveArtikel(artikel);
}

public Artikel EditArtikel(Artikel artikel)
{
    return _artikelAccessor.SaveArtikel(artikel);
}

public void DeleteArtikel(int? id)
{
    _artikelAccessor.DeleteArtikel(id);
}
}

```

Das ArtikelAccessor ist sehr ähnlich aufgebaut wie das vom Service. Die Kommunikation mit der Datenbank im Code wird mit dem Entity «db» geregelt. SaveArtikel(artikel) ist hierbei eine interessante Methode, sie kann zum Bearbeiten genutzt werden, aber auch zum Erstellen eines Artikels und dessen Speicherung. Dabei wird in der Methode geprüft ob der Artikel so in der Datenbank vorhanden ist. Falls der Artikel in der Datenbank vorhanden ist wird der jeweilige Datensatz geändert und gespeichert, wenn der Artikel nicht in der Datenbank vorhanden ist wird er neu erstellt.

```

public class ArtikelAccessor : IArtikelAccessor
{
    private Entities db = new Entities();

    public IEnumerable<Artikel> GetArtikels()
    {
        return db.Artikels.Include(a => a.Kategorie).Include(a =>
a.Nutzer).Include(a => a.Raum);
    }

    public Artikel GetArtikel(int? id)
    {
        return db.Artikels.Find(id);
    }

    public Artikel SaveArtikel(Artikel artikel)
    {
        if (db.Artikels.Any(a => a.ID == artikel.ID))
        {
            db.Entry(artikel).State = EntityState.Modified;
        }
        else

```



```

    {
        db.Artikels.Add(Artikel);
    }
    db.SaveChanges();

    return db.Artikels.Find(Artikel.ID);
}

public void DeleteArtikel(int? id)
{
    Artikel Artikel = db.Artikels.Find(id);
    db.Artikels.Remove(Artikel);
    db.SaveChanges();
}
}

```

AutoNumeric

Zum Erfassen des Preises und dessen Veränderung wurde die JavaScript Library AutoNumeric eingesetzt, diese Library ermöglicht es, dass der Preis richtig dargestellt wird und es für den Benutzer einfacher ist einen Wert einzugeben. Es wurden die schweizer Kultureinstellungen verwendet.

Beschaffungswert (in CHF)

3'005'000.25	CHF
--------------	-----

Abbildung 10: Preis Eingabe mit AutoNumeric.js (Create View Artikel)

@* Die JavaScript Library Autonumeric wurde eingesetzt, damit die Währung richtig angezeigt wird im Input-Feld. <http://autonumeric.org> *@

```

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
    @Scripts.Render("~/bundles/autonumeric")

    <script>
        new AutoNumeric('#currencyInput', {
            decimalCharacter: '.',
            digitGroupSeparator: '\'',
        });

        $.validator.methods.range = function (value, element, param) {
            var globalizedValue = value.replace(".", ",");
            return this.optional(element) || (globalizedValue >= param[0] &&
            globalizedValue <= param[1]);
        }

        $.validator.methods.number = function (value, element) {
            return this.optional(element) || /-?(?:\d+|\d{1,3}(?:[s\.,]\d{3})+)(?:[\.\,]\d+)?$/i.test(value);
        }

        $("#artikelForm").submit(function (event) {
            value_with_separators = $('#currencyInput').val();
            clean_value = value_with_separators.replace(/'/g, '');
            console.log(clean_value);
            $('#currencyInput').val(clean_value);
        });
    </script>
}

```

Suche

Die Suche ist im ArtikelController implementiert, weil sie nur dort benötigt wird. Dazu wurde ein separates Model erstellt für die Suchabfrage die «ArtikelSearchModel» heisst. Diese wird mit den angegebenen Parametern im Link abgefüllt. Die Parameter sind Nullable, das erkennt man am Fragezeichen (int?) nach dem Datentyp, dies bedeutet das der Parameter auch leer sein darf und es deswegen keine Fehlermeldung gibt. Ein Beispiel für ein Link kann so aussehen:

http://se.inv.muster.ch/?KategorieID=1&RaumID=&BenutzerID=4&FiBuKontoID=1&Search=ARB_00001&ArtikelYearsStart=2011&ArtikelYearsEnd=

Dabei sieht man, dass die ID der jeweiligen Kategorie, Raum, Benutzer oder FiBuKonto genommen wird zum Suchen, das Dropdown in der Suche für diese Felder gibt die ID weiter. Ebenfalls kann der Link weitergegeben werden um wieder benutzt zu werden, dies ist sehr nützlich bei Suchabfragen die öfters benötigt werden.

```
public ActionResult Index(int? KategorieID, int? FiBuKontoID, int? RaumID, int?
BenutzerID, int? ArtikelYearsStart, int? ArtikelYearsEnd, string Search, string
sortOrder, int page = 1, int pagesize = 25)
{
    // Das Searchmodel wird hier abgefüllt
    ArtikelSearchModel artikelSearchModel = new ArtikelSearchModel()
    {
        KategorieID = KategorieID,
        FiBuKontoID = FiBuKontoID,
        RaumID = RaumID,
        BenutzerID = BenutzerID,
        ArtikelYearsStart = ArtikelYearsStart,
        ArtikelYearsEnd = ArtikelYearsEnd,
        Search = Search
    };

    IEnumerable<Artikel> artikelList =
    _artikelService.GetArtikels(artikelSearchModel);

    Session["BackLink"] = "/Artikel?KategorieID=" + KategorieID +
"&FiBuKontoID=" + FiBuKontoID + "&RaumID=" + RaumID + "&BenutzerID=" + BenutzerID +
"&ArtikelYearsStart=" + ArtikelYearsStart + "&ArtikelYearsEnd=" + ArtikelYearsEnd +
"&Search=" + Search + "&sortOrder=" + sortOrder + "&page=" + page;

    ViewBag.KategorieID = new
SelectList(_kategorieService.GetKategorien().OrderBy(x => x.Bezeichnung), "ID",
"Bezeichnung", KategorieID);
    ViewBag.FiBuKontoID = new
SelectList(_fiBuKontoService.GetFiBuKontos().OrderBy(x => x.Nummer), "ID", "Nummer",
FiBuKontoID);
    ViewBag.RaumID = new SelectList(_raumService.GetRaums().OrderBy(x =>
x.Bezeichnung), "ID", "Bezeichnung", RaumID);
    ViewBag.BenutzerID = new
SelectList(_benutzerService.GetBenutzers().OrderBy(x => x.Fullname), "ID", "Fullname",
BenutzerID);
    ViewBag.ArtikelYearsStart = ArtikelYears();
    ViewBag.ArtikelYearsEnd = ArtikelYears();
    ...
    return View(new PagedList<Artikel>(artikelList.ToList(), page, pagesize));
}
```

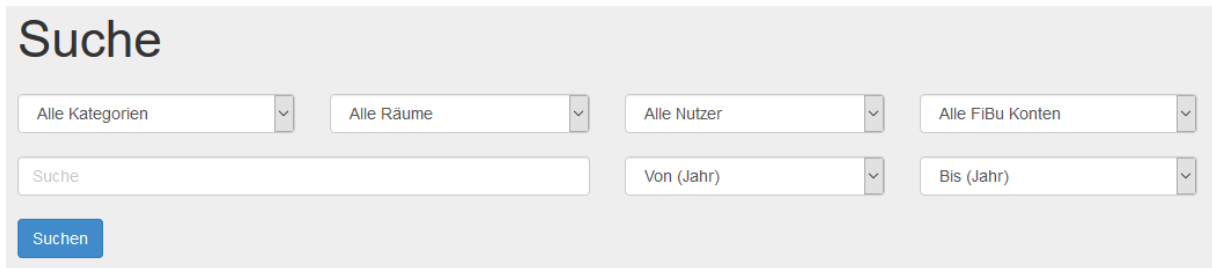
Eine Instanz von ArtikelSearchModel wird hierbei erstellt, die dann abgefüllt wird. Das erstellte ArtikelSearchModel wird dann verwendet beim Service «ArtikelService» und von dort aus geht es dann in den Accessor «ArtikelAccessor». Dort wird dann mit dem Entity Framework und mit Hilfe des erstellten ArtikelSearchModels eine Abfrage auf die Datenbank erstellt, die dann mit den Suchresultaten als Liste zurückgegeben wird. Mit .Contains(word) wird der Suchtext gesucht in der Inventarnummer, Bezeichnung und Beschreibung.

```
public IEnumerable<Artikel> GetArtikels(ArtikelSearchModel artikelSearchModel)
{
    IEnumerable<Artikel> artikellist = db.Artikels.Include(a =>
a.Kategorie).Include(a => a.Nutzer).Include(a => a.Raum);

    if (!String.IsNullOrEmpty(artikelSearchModel.Search))
    {
        artikelSearchModel.Search = artikelSearchModel.Search.ToLower();
        foreach (string word in artikelSearchModel.Search.Split(' '))
        {
            artikellist = artikellist.Where(a =>
a.InventarNummer.ToLower().Contains(word) || a.Bezeichnung.ToLower().Contains(word) ||
a.Beschreibung.ToLower().StartsWith(word));
        }
    }

    if (artikelSearchModel.KategorieID.HasValue)
    {
        artikellist = artikellist.Where(c =>
c.KategorieID.Equals(artikelSearchModel.KategorieID.Value));
    }
    if (artikelSearchModel.FiBuKontoID.HasValue)
    {
        artikellist = artikellist.Where(c =>
c.Kategorie.FiBuKontoID.Equals(artikelSearchModel.FiBuKontoID.Value));
    }
    if (artikelSearchModel.RaumID.HasValue)
    {
        artikellist = artikellist.Where(c =>
c.RaumID.Equals(artikelSearchModel.RaumID.Value));
    }
    if (artikelSearchModel.BenutzerID.HasValue)
    {
        artikellist = artikellist.Where(c =>
c.BenutzerIDNutzer.Equals(artikelSearchModel.BenutzerID.Value));
    }
    if (artikelSearchModel.ArtikelYearsStart.HasValue)
    {
        artikellist = artikellist.Where(c => c.Beschaffungsdatum.Date >= new
DateTime(int.Parse(artikelSearchModel.ArtikelYearsStart.Value.ToString()), 1, 1));
    }
    if (artikelSearchModel.ArtikelYearsEnd.HasValue)
    {
        artikellist = artikellist.Where(c => c.Beschaffungsdatum.Date <= new
DateTime(int.Parse((artikelSearchModel.ArtikelYearsEnd.Value + 1).ToString()), 1,
1)).ToList();
    }

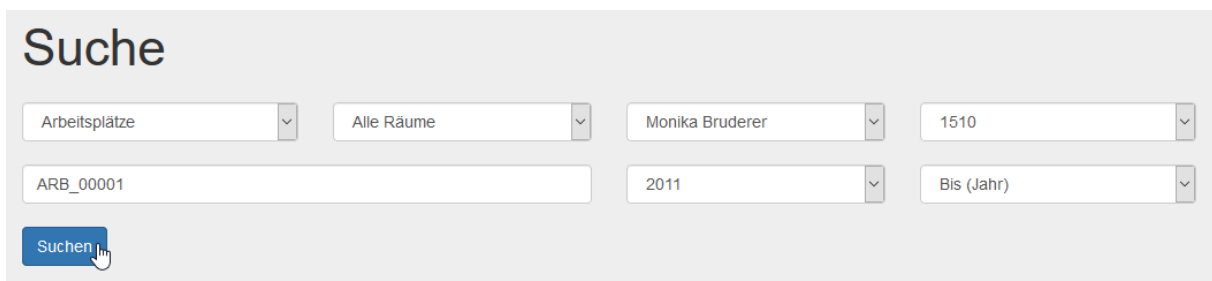
    return artikellist;
}
```



The screenshot shows a search interface titled 'Suche'. It contains four dropdown menus at the top, each with a downward arrow: 'Alle Kategorien', 'Alle Räume', 'Alle Nutzer', and 'Alle FiBu Konten'. Below these is a search text input field containing the placeholder 'Suche'. To the right of the text field are two more dropdown menus: 'Von (Jahr)' and 'Bis (Jahr)'. At the bottom left is a blue button labeled 'Suchen'.

Abbildung 11: Suche ohne ausgefüllte Felder

Insgesamt sieht dann die Suche so aus: es gibt vier Dropdown Listen für die Kategorien, Räume, Nutzer und FiBu Konten. Man kann einen Suchtext eingeben, der überprüft ob der jeweilige Text in der Inventarnummer, Beschreibung oder Bezeichnung vorkommt. Zusätzlich kann man nach dem Jahr filtern mit «Von» und «Bis», es reicht hierbei auch nur eine Auswahl anzugeben, dann werden alle Artikel bis und mit zu diesem Jahr angezeigt oder von und mit diesem Jahr.



The screenshot shows the same search interface as before, but with the fields filled. The first dropdown menu is set to 'Arbeitsplätze', the second to 'Alle Räume', the third to 'Monika Bruderer', and the fourth to '1510'. The search text input field contains 'ARB_00001'. The 'Von (Jahr)' dropdown is set to '2011' and the 'Bis (Jahr)' dropdown is set to 'Bis (Jahr)'. The blue 'Suchen' button is still at the bottom left.

Abbildung 12: Suche mit ausgefüllten Feldern

Mit dem Klicken auf den Suchbutton wird dann der Suchprozess durchgeführt, zusätzlich aber auch noch asynchron mit dem Auswählen eines Items in einem Dropdown.

PagedList

Die PagedList werden mit dem NuGet Package Manager hinzugefügt. Alle Listen im Frontend werden mit einer PagedList angezeigt in der pro Seite 25 Einträge sind. Die einzelnen Spalten können hierbei aufsteigend oder absteigend sortiert werden, in dem man auf den Titel der jeweiligen Spalte klickt.
































Wenn man auf die Details eines Items geht z.b beim Raum und dann wieder auf den zurück Link geht, kommt man wieder zurück zu der PagedList auf der man war mit der jeweiligen Page und der Sortierung die man ausgewählt hat.

z.b bei Raum: [http:// se.inv.muster.ch /Raum?&sortOrder=Bezeichnung&page=1](http://se.inv.muster.ch/Raum?&sortOrder=Bezeichnung&page=1)

Die Logik dahinter ist im Wesentlichen dieselbe wie bei der Index Methode mit den Such Parametern.

Raum

Raum erfassen

Lokalisierung	Bezeichnung 	
OG	Buchhaltung	  
EG	Büro Administration	  
OG	Büro Applikationsentwicklung	  
EG	Büro Leer	  
OG	Büro Schulung	  
EG	Büro Technik	  
UG	Lager Büromaterial	  
UG	Rechenzentrum	  
OG	Sitzungszimmer	  
EG	Technik Werkstatt	  

1

Abbildung 13: die PagedList auf der Index View von Raum, die Aufsteigend nach Bezeichnung sortiert wurde

Kategorien, Räume, Benutzer, FiBu Konten

Die Accessoren und Services, sowie die Views sind ähnlich aufgebaut wie beim Artikel. Der grösste Unterschied hierbei findet sich bei den Benutzern. Denn der BenutzerController hat ein Attribut «Authorize», mit dem Parameter 'Administrator'. So können nur die Benutzer mit der Rolle 'Administrator' auf die Views der Benutzer zugreifen. Damit ist sichergestellt das nur Benutzer mit der Rolle 'Administrator' Benutzer erstellen, bearbeiten und löschen können.

```
[Authorize(Roles = "Administrator")]
1 reference
public class BenutzerController : Controller
{
    IBenutzerService _benutzerService;
    IRolleService _rolleService;
```

Abbildung 14: BenutzerController Zugang auf Rolle 'Administrator' beschränken

Validierung & Fehlermeldungen

Bei der Eingabe falscher Benutzerdaten kommt eine Fehlermeldung und das Passwort muss erneut eingegeben werden. Dies wird mit einem Validation Summary realisiert, der an das Frontend zurückgegeben und schlussendlich angezeigt wird.

Login

Einloggen mit Windows Benutzerdaten

Ungültige Benutzerdaten

Benutzername

test

Passwort

Login

Abbildung 15: Fehlermeldung beim Login wegen ungültiger Benutzerdaten

Schon beim Eintippen sieht man auf den Erfassungsmasken der Models ob es so validiert werden kann. Grün bedeutet, dass das Feld so ausgefüllt werden kann, Rot hingegen heisst, dass das Feld falsch befüllt wurde oder bei Pflichtfeldern nichts steht, z.b ein Wort in einem Feld wo eine Zahl erwartet wird, wird nicht akzeptiert. Dies wurde mit einer Anpassung im JQuery Validate erreicht, wie es auf Stack Overflow empfohlen wurde, der Link dazu steht in den Quellen.

Erstellen

Artikel

Inventar Nummer

test 1

Bezeichnung

r

Beschreibung

testd

Kategorie

Das Feld "Kategorie" ist erforderlich.

Raum

Buchhaltung

Nutzer

Das Feld "Nutzer" ist erforderlich.

Beschaffungsdatum

06 . 03 . 2018

Beschaffungswert (in CHF)

500.00

CHF

Erstellen

Abbildung 16: Validierung beim Erstellen von einem Artikel

PDF Export

Das PDF wird mithilfe des NuGet Paketes «iTextSharp» generiert. Der PDF Export wird bei der Übersichtseite der Artikel gebraucht. Das PDF wird in der Methode «Download» im ArtikelController generiert.

Hierbei wurde auch darauf geachtet, dass der Name der Datei, das genaue Datum enthält. Dies ist praktisch zur Aufbewahrung und es ist dann ersichtlich, wie aktuell das PDF noch ist.

```
// Methode für den PDF Download eines Artikels
public ActionResult Download(int? KategorieID, int? FiBuKontoID, int? RaumID,
int? BenutzerID, int? ArtikelYearsStart, int? ArtikelYearsEnd, string Search)
{
    // Das Searchmodel wird hier abgefüllt
    ArtikelSearchModel artikelSearchModel = new ArtikelSearchModel()
    {
        KategorieID = KategorieID,
        FiBuKontoID = FiBuKontoID,
        RaumID = RaumID,
        BenutzerID = BenutzerID,
        ArtikelYearsStart = ArtikelYearsStart,
        ArtikelYearsEnd = ArtikelYearsEnd,
        Search = Search
    };

    IEnumerable<Artikel> artikelList =
_artikelService.GetArtikels(artikelSearchModel);

    Response.ContentEncoding = new System.Text.UTF8Encoding(); // UTF-8

    Response.ContentType = "application/pdf";

    FileContentResult fcr = new FileContentResult(RenderViewToPDF("Artikel",
"exportpdf", artikelList.ToArray()), "application/pdf");
    fcr.FileNameDownload = "LabelsInventar_" + GetTimestamp(DateTime.Now) +
".pdf";
    return fcr;
}
```

Im PDF werden die Artikel die auf der Übersichtseite der Artikel ersichtlich sind als Labels abgebildet. Das heisst es können auch PDF zu Artikeln erstellt werden die gesucht wurden. Deswegen ist im Code auch das ArtikelSearchModel, durch den Klick auf den «Labels erstellen in PDF» Button, wird das ArtikelSearchModel automatisch übernommen von der Suche. Das PDF kann dann ausgedruckt werden auf ein A4 Etiketten Blatt, die Etiketten können dann auf den Artikeln draufgeklebt/platziert werden.

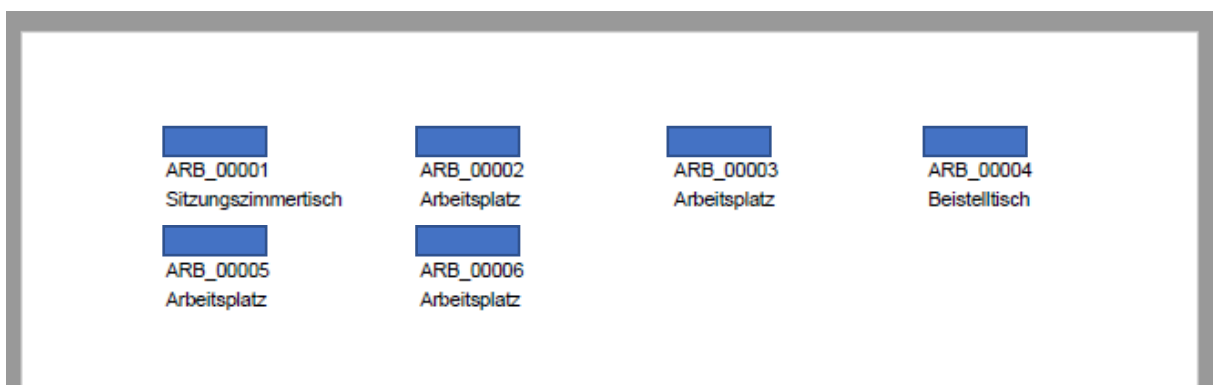


Abbildung 17: Generiertes PDF der Artikel

Excel Export

Das Excel wird mithilfe des NuGet Paketes «EPPlus» generiert. Der Excel Export ist ähnlich aufgebaut wie der PDF Export. Dabei wird auch die aktuelle Suche berücksichtigt, bei der Zusammenstellung der Excel Tabelle. Der Inhalt des PDFs wird hier mit UTF-8 kodiert, wie auch beim PDF.

UTF-8 ist wichtig für Zeichen wie: ü, ö, ä, die in der normalen Kodierung verwendet werden, die nicht vorhanden sind und deswegen falsch dargestellt werden, wenn nicht definiert wird, dass es ins UTF-8 (8-Bit UCS Transformation Format) kodiert werden muss.

```
// Diese Methode wird benötigt zum erstellen der Excel Liste
public ActionResult ArtikelExport(int? KategorieID, int? FiBuKontoID, int?
RaumID, int? BenutzerID, int? ArtikelYearsStart, int? ArtikelYearsEnd, string Search)
{
    // Das Searchmodel wird hier abgefüllt
    ArtikelSearchModel artikelSearchModel = new ArtikelSearchModel()
    {
        KategorieID = KategorieID,
        FiBuKontoID = FiBuKontoID,
        RaumID = RaumID,
        BenutzerID = BenutzerID,
        ArtikelYearsStart = ArtikelYearsStart,
        ArtikelYearsEnd = ArtikelYearsEnd,
        Search = Search
    };

    IEnumerable<Artikel> artikelList =
_artikelService.GetArtikels(artikelSearchModel);

    MemoryStream ms = ArtikelToExcel(artikelList.OrderBy(d =>
d.InventarNummer).ToList());
    ms.WriteTo(Response.OutputStream);
    Response.ContentType = "application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet";
    Response.ContentEncoding = new System.Text.UTF8Encoding(); // UTF-8
    Response.AddHeader("Content-Disposition",
"attachment;filename=InventarListe_" + GetTimestamp(DateTime.Now) + ".xlsx");
    Response.StatusCode = 200;
    Response.End();

    return null;
}
```

Der Dateiname der generierten Excel Datei beinhaltet auch das aktuelle Datum aus denselben Gründen wie beim PDF. Die Spaltenreihenfolge ist hierbei ein bisschen anders als bei der Übersichtsseite von Artikel. Bezeichnung und Beschreibung sind hier getrennt, sowie Raum und Stockwerk.

Inventar Nummer	Bezeichnung	Beschreibung	Kategorie	Stockwerk	Raum	FiBo Konto (Nummer)	Beschaffungsdatum	Beschaffungswert (in CHF)
ARB_00001	Sitzungszimmertisch	249x100x74 cm	Arbeitsplätze	OG	Sitzungszimmer	1510	9.12.2011	10'000.00
ARB_00002	Arbeitsplatz	180x80x72 cm	Arbeitsplätze	OG	Buchhaltung	1510	9.12.2011	732.00
ARB_00003	Arbeitsplatz	200x100x73 cm	Arbeitsplätze	EG	Büro Leer	1510	9.12.2011	524.00
ARB_00004	Beistelltisch	59.5x50x64.5 cm	Arbeitsplätze	OG	Buchhaltung	1510	9.12.2011	100.00
ARB_00005	Arbeitsplatz	180x90x75 cm	Arbeitsplätze	EG	Büro Technik	1510	9.12.2011	530.00
ARB_00006	Arbeitsplatz	180x90x75 cm	Arbeitsplätze	EG	Büro Technik	1510	9.12.2011	530.00

Abbildung 18: Generiertes Excel der Artikel

Anleitung

Zusätzlich wurde eine Anleitung erstellt, die dem Benutzer bei der Bedienung der internen Inventarverwaltung behilflich sein soll. In der Anleitung steht wie man Artikel erstellt, bearbeitet, löscht, betrachtet und sucht. Da dies bei den anderen Modellen ähnlich ist wie bei «Artikel», bin ich nur auf «Artikel», das wichtigste und am häufigsten verwendete Modell der Applikation eingegangen. Nebenbei findet man in der Anleitung noch Informationen zum PDF und Excel Export.

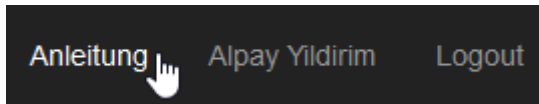


Abbildung 19: Anleitung in der Navigation

Der Link zur Anleitung ist dabei in der Navigation rechts platziert, so dass der Benutzer von jeder Seite aus darauf zugreifen kann.

Mehrsprachigkeit

Die Lösung wurde weitgehend übernommen aus dem Kundenprojekt «com.jmg.cc». Das Problem mit der möglichen Mehrsprachigkeit, die einmal benötigt werden könnte wurde schon im Voraus gelöst. Mithilfe eines HTML Helpers, der auch im Layout eingesetzt werden kann, ähnlich wie bei der Navigation mit `isActive()`. Z.b im Layout File das sich im «Shared» Verzeichnis befindet. In der Ressourcen Datei «String.resx» trägt man diejenigen Begriffe ein, die in mehreren Sprachen verfügbar sein sollte. Die Lokalisierung des Benutzers wird in der Datei «Localization» bestimmt, dies ist möglich durch die CultureSettings Information, die der Browser gibt. Die Globalisierung von eingetragenen Begriffen wird durch den Befehl `Resource()` ermöglicht, wie man am Code unten sieht.

```
<footer>
    <p>&copy; @DateTime.Now.Year - @Html.Resource("Company") - Inventar </p>
</footer>
```

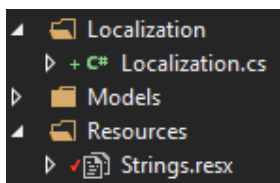


Abbildung 20: Das Ressourcen und Lokalisierungs File, die nötig sind für Mehrsprachigkeit

Design

Beim Design der Website wurde mit responsive Webdesign darauf geachtet, dass die Webapplikation auch auf dem Tablet bedienbar ist und gut dargestellt wird. Grundsätzlich wurde das schon mit Bootstrap erreicht.

Zusätzlich wurden noch Buttons mit Icons hinzugefügt, anstatt Text. Das sieht schöner aus und braucht weniger Platz.



Abbildung 21: Buttons mit Icons (Index View Artikel)

Kontrollieren

Das Kontrollieren ist die fünfte Phase der IPERKA-Methode.

Hier werden Korrekturen sowie gefundene Fehler beschrieben. Mit dem Testkonzept das in der Phase Planung erstellt wurde, wird die ganze Applikation getestet. Das Projekt wurde hierzu auf den Server ehost51 hochgeladen, die Tests werden unter dem Link <http://inv.se.muster.ch> durchgeführt.

Testprotokoll

Angaben PC & Software:

Name	Lenovo TC M900 10FM-000G
Prozessor	Intel(R) Core(TM) i7-6700T CPU @ 2.80 GHz
Ram	8 GB
Testumfeld / Browser	Firefox Browser (Version 59.0b11 (64-Bit))
Betriebssystem	Windows 10 Pro 64-Bit (Version 1607)

Testfall	Resultat	Datum	Tester	Bemerkung	Unterschrift
1	Passed	20.03.2018	A.Yildirim	Langsame Ausführung	A.Yildirim
2	Passed	20.03.2018	A.Yildirim	-	A.Yildirim
3	Passed	20.03.2018	A.Yildirim	-	A.Yildirim
4	Passed	20.03.2018	A.Yildirim	-	A.Yildirim
5	Passed	20.03.2018	A.Yildirim	-	A.Yildirim
6	Passed	20.03.2018	A.Yildirim	-	A.Yildirim
7	Passed	20.03.2018	A.Yildirim	Langsame Ausführung	A.Yildirim
8	Passed	20.03.2018	A.Yildirim	-	A.Yildirim
9	Passed	20.03.2018	A.Yildirim	-	A.Yildirim
10	Passed	20.03.2018	A.Yildirim	-	A.Yildirim
11	Passed	20.03.2018	A.Yildirim	-	A.Yildirim
12	Passed	20.03.2018	A.Yildirim	-	A.Yildirim

13	Passed	20.03.2018	A.Yildirim	-	A.Yildirim
14	Passed	20.03.2018	A.Yildirim	-	A.Yildirim
15	Passed	20.03.2018	A.Yildirim	-	A.Yildirim
16	Passed	20.03.2018	A.Yildirim	-	A.Yildirim
17	Passed	20.03.2018	A.Yildirim	-	A.Yildirim
18	Passed	20.03.2018	A.Yildirim	-	A.Yildirim
19	Passed	20.03.2018	A.Yildirim	-	A.Yildirim
20	Passed	20.03.2018	A.Yildirim	-	A.Yildirim
21	Passed	20.03.2018	A.Yildirim	-	A.Yildirim

Testbericht

Die Applikation reagierte langsam auf den ersten Testfall und den siebten. Das lag daran das die Applikation gerade neu auf den Server geladen wurde und noch über keinen Cache verfügte. Beim zweiten Versuch klappte alles hervorragend und ging deutlich schneller.

Insgesamt funktionierte alles wie erwartet und es traten keine Fehler auf.

Die Applikation kann so eingesetzt werden.

Reflexion

Das Informieren verlief gut, es war sehr deutlich was gefordert wurde in der Aufgabenstellung, nur bei vereinzelten Punkten musste ich nachfragen.

Das Planen verlief auch gut, das einzige was mir dort aufgefallen ist, ist das die Dokumentation mehr Zeit in Anspruch nimmt als erwartet.

Durch die gute Planung wusste ich beim Realisieren genau was ich zu tun hatte, über das ganze Projekt hinweg. Das grösste Problem was ich beim Realisieren hatte war eher das dokumentieren, den genauen Ablauf und die Funktionsweise zu beschreiben hat doch mehr Zeit benötigt als ich erwartet habe.

Beim Realisieren während dem Programmieren traten auch Fehler auf, diese konnte ich jedoch alle lösen.

Sehr zufrieden war ich beim Kontrollieren, als ich beim Testen gesehen habe das die ganze Applikation keine Fehler zurückgibt.

Rückblickend habe ich die Dokumentation eindeutig unterschätzt, es musste doch mehr Zeit in die Dokumentation investiert werden als erwartet, wie man es am Zeitplan auch erkennen kann. Glücklicherweise war ich deswegen nie in Zeitnot.

Für das nächste Projekt nimm ich mit, dass ich mehr Zeit in das Dokumentieren einplanen muss, da ich diesen Punkt doch recht unterschätzt habe.

Insgesamt muss ich sagen das ich sehr zufrieden bin mit dem Verlauf der IPA und dem daraus entstandenen Resultat, die Applikation wird eine grosse Bereicherung sein für die Verwaltung des Inventars und viel Zeit sparen.

Glossar

Begriff	Erklärung
ADO.NET	Ist ein Framework, mit dem Zugriffe auf Datenbanken wie SQL Server möglich sind.
Bootstrap	Ist ein CSS Framework das zur Gestaltung des Frontends einer Website dient. Es bietet viele Gestaltungsvorlagen wie Buttons, Tabellen und ein Grid System u.s.w.
Controller	Der Controller verbindet die Views mit dem Model und enthält Logik, die notwendig ist, damit die Applikation funktioniert.
CRUD	(Create/Read/Update/Delete), umfasst vier grundlegende Funktionen. Die für die Standardfunktionalität der Daten auf den Ansichtseiten sorgt. (Erstellen/Lesen/Bearbeiten/Löschen).
Dependency Injection	Ist eine Entwurfsarchitektur, welche die Referenzierungen der 3 Layer des Projektes regelt und es den Layern ermöglicht unabhängig voneinander zu funktionieren.
Globalisierung	Mit der Globalisierung von Stellen im Code erreicht man, dass die gleiche View in mehreren Ländern verwendet werden kann. Das Ziel der Globalisierung ist, dass der Inhalt für möglichst viele Nutzer verständlich ist. Dies erreicht man z.b mit Resource(«Begriff»), hier wird die Übersetzung vom Begriff geliefert, die der Nutzer wünscht. Das kann z.b Englisch, Deutsch u.s.w. sein.
Layer	In drei Layern ist das ganze Projekt verteilt, es gibt den Business Logic Layer, der für die Änderungen der Daten und den Presentation Layer zuständig ist. Der Data Access Layer, der den Zugriff auf die Datenbank regelt und den Presentation Layer, der zuständig für die Ausgabe und Darstellung ist.
LDAP	Steht für «Lightweight Directory Access Protocol», das Protokoll ist zuständig für den Zugriff der Applikation auf den Windows AD Server, für die Authentifizierung.
Lokalisierung	Lokalisierung ermöglicht es der Anwendung herauszufinden wer der Benutzer überhaupt ist und welche Sprache er spricht und lesen kann. Dies ist z.b möglich in dem die Applikation die Informationen liest, die der Nutzer mitgibt z.b die Sprachpräferenz (CultureInfo). Mit der Sprachpräferenz weiss die Applikation welche Sprache der Browser des Benutzers bevorzugt.
Model	Das Model repräsentiert die jeweilige Datenbanktabelle als Klasse, nur durch Models sollte der Zugriff auf die Datenbank erfolgen.
MVC	MVC ist ein Architekturmuster und steht für Model-View-Controller. MVC ermöglicht die einfache Wiederverwendung von einzelnen Komponenten und erleichtert auch das Anpassen einzelner Komponenten. Der Code ist klar und ersichtlich aufgeteilt in Layout, Logik und Modelle(Daten).
NuGet Paket	NuGet ist eine quelloffene Paketverwaltung für die Entwicklung unter .NET mit Visual Studio. Verschiedenste Libraries und Codeausschnitte werden angeboten, die über den Package Manager hinzugefügt werden können.
OWIN	OWIN bedeutet Open Web Interface for .NET. Es wird für die Autorisierung benötigt und ist der Standard für die Kommunikation

	zwischen dem IIS Webserver und der Webapplikation.
Razor	Razor ist eine Programmier Syntax im ASP.Net, mit dem in den Views zusätzliche Logik implementiert werden kann mittels C#-Programmiersprache, man erkennt Razor code am @ Zeichen vor der Anweisung, z.b @ViewBag.Title .
Responsive Webdesign	Responsive Webdesign ermöglicht eine gute Ansicht auf dem Destkop und auf mobilen Geräten. Für responsives Webdesign kann man z.b Bootstrap einsetzen oder direkt im CSS Änderungen vornehmen.
SQL	Eine Abkürzung für «Structured Query Language» ist eine Datenbanksprache, die verwendet wird um Datenbanken zu erstellen, bearbeiten und Daten abzufragen. Ein Beispiel für eine SQL Abfrage wäre «SELECT * FROM tabellen_name;», die alle Datensätze der jeweiligen Tabelle abfragt.
TFS	Der Team Foundation Server ist zuständig für die Versionsverwaltung und Speicherung von Projekten. Das ist vor allem nützlich wenn mehrere Entwickler an einem Projekt arbeiten. Er ist sehr gut geeignet für Backups, da man auf jede einzelne Version zurückgreifen kann.
Unity	Durch Unity ist eine einfache Umsetzung der Entwurfsarchitektur Dependency Injection möglich, es ist eine Library die per NuGet zu einem ASP.NET Projekt hinzugefügt werden kann.
View	Das Markup für die Ausgabe im Frontend ist in der View enthalten. Sie wird durch den Controller aufgerufen und heisst ins deutsche übersetzt «Ansicht».
Windows AD	Das Windows Active Directory ist der Verzeichnisdienst von den Microsoft Windows Servern. Auf der Windows AD sind die Benutzer gespeichert. Benötigt wird es für die Authentifizierung.
.NET	Das .NET Framework besteht aus einer Laufzeitumgebung (bekannt unter Common Language Runtime (CLR)), Klassenbibliotheken, Services und Programmierschnittstellen. Die Möglichkeiten der Verwendung sind vielfältig z.b die System Klassenbibliothek (System.Web, System.Collections, System.Data u.s.w.).

Abbildungsverzeichnis

Abbildung 1: Datenbankmodell im Visio	22
Abbildung 2: Die drei Layers (Schichten) im Visual Studio.....	35
Abbildung 3: Datenbank Diagramm im Visual Studio	36
Abbildung 4: Ordnerstruktur der Models und der ModelsPartial	37
Abbildung 5: Ordnerstruktur Services & Accessors	40
Abbildung 6: Startup.cs OWIN Konfiguration	42
Abbildung 7: FilterConfig.cs, festlegen des Filters für die Autorisierung.....	42
Abbildung 8: Account.cs mit der Login Methode.....	44
Abbildung 9: Navigation mit aktivem Item «Artikel» mit der Rolle «Administrator»	47
Abbildung 10: Preis Eingabe mit AutoNumeric.js (Create View Artikel).....	49
Abbildung 11: Suche ohne ausgefüllte Felder.....	52
Abbildung 12: Suche mit ausgefüllten Feldern	52
Abbildung 13: die PagedList auf der Index View von Raum, die Aufsteigend nach Bezeichnung sortiert wurde.....	53
Abbildung 14: BenutzerController Zugang auf Rolle 'Administrator' beschränken	53
Abbildung 15: Fehlermeldung beim Login wegen ungültiger Benutzerdaten	54
Abbildung 16: Validierung beim Erstellen von einem Artikel	54
Abbildung 17: Generiertes PDF der Artikel	55
Abbildung 18: Generiertes Excel der Artikel	56
Abbildung 19: Anleitung in der Navigation	57
Abbildung 20: Das Ressourcen und Lokalisierungs File, die nötig sind für Mehrsprachigkeit ..	57
Abbildung 21: Buttons mit Icons (Index View Artikel)	57

Quellenverzeichnis

<https://www.pkorg.ch/>

<http://www.pk-ag.ch/>

<https://github.com/devtrends/Unity.Mvc5>

https://en.wikipedia.org/wiki/Active_Directory

https://en.wikipedia.org/wiki/Team_Foundation_Server

https://en.wikipedia.org/wiki/Open_Web_Interface_for_.NET

https://en.wikipedia.org/wiki/Lightweight_Directory_Access_Protocol

<https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller>

https://en.wikipedia.org/wiki/.NET_Framework

https://en.wikipedia.org/wiki/Dependency_injection

<https://en.wikipedia.org/wiki/NuGet>

<https://de.wikipedia.org/wiki/SQL>

https://www.w3schools.com/asp/razor_syntax.asp

<http://benfoster.io/blog/aspnet-identity-stripped-bare-mvc-part-1>

<https://www.codeproject.com/Questions/526055/LDAPplususerplusAuthenticationplususingplusDirecto>

<https://docs.microsoft.com/en-us/aspnet/mvc/overview/getting-started/database-first-development/enhancing-data-validation>

<https://docs.microsoft.com/en-us/aspnet/mvc/overview/older-versions-1/getting-started-with-mvc/getting-started-with-mvc-part7>

<http://benfoster.io/blog/aspnet-identity-stripped-bare-mvc-part-1>

<http://www.benripley.com/development/javascript/asp-mvc-4-validation-with-bootstrap-3/>

<https://stackoverflow.com/questions/12107263/why-is-validationsummarytrue-displaying-an-empty-summary-for-property-errors>

<https://stackoverflow.com/questions/20410623/how-to-add-active-class-to-html-actionlink-in-asp-net-mvc>

<https://blogs.msdn.microsoft.com/dmx/2014/10/14/was-ist-eigentlich-dependency-injection-di/>

Anhang

Die Schrift des Anhanges wurde stark verkleinert auf die Schriftgrösse 7, sonst wäre die ganze Dokumentation über 200+ Seiten gross und schwer zum ausdrucken und binden.

Anleitung



ANLEITUNG

Inventarverwaltung

In dieser Einleitung werden die wichtigsten Funktionen
der internen Inventarverwaltung erklärt,
die unter dem Link [\[Link\]](#) erreichbar ist.

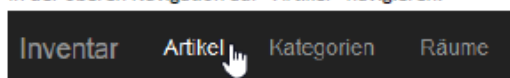
Verfasser: Alpay Yildirim
März, 2018

Inhalt

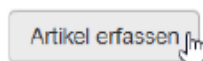
Artikel Erstellen	1
Artikel Bearbeiten	2
Artikel Betrachten	3
Artikel Löschen	4
Artikel Suchen	5
PDF Exportieren.....	6
Excel Exportieren.....	7

Artikel Erstellen

1. In der oberen Navigation auf «Artikel» navigieren.



2. Auf den Knopf «Artikel erfassen» klicken, dass sich oberhalb rechts der Artikel Tabelle befindet.



3. Alle benötigten Felder in der Erfassungsmaske ausfüllen, ein grüner Rahmen bedeutet, dass das Feld korrekt ausgefüllt wurde. Nach dem Ausfüllen auf den Knopf «Erstellen» klicken.

Erstellen

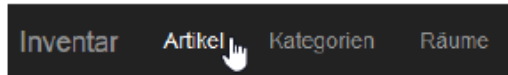
Artikel

Inventar Nummer	<input type="text" value="DRK_00001"/>
Bezeichnung	<input type="text" value="HP 500"/>
Beschreibung	<input type="text" value="Hauptdrucker"/>
Kategorie	<input type="text" value="Drucker"/>
Raum	<input type="text" value="Technik Werkstatt"/>
Nutzer	<input type="text" value="Hans Mustermann"/>
Beschaffungsdatum	<input type="text" value="01.01.2010"/>
Beschaffungswert (in CHF)	<input type="text" value="1'200.00"/> <input type="text" value="CHF"/>
<input type="button" value="Erstellen"/>	



[Zurück](#)

Artikel Bearbeiten

1. In der oberen Navigation auf «Artikel» navigieren.



2. In der Tabelle beim jeweiligen Artikel auf den «Bearbeiten Icon» klicken.

Inventar Nummer	Artikel	Kategorie	Raum	Nutzer	Fibu-Konto	Beschaffungsdatum	Beschaffungswert (in CHF)	
AR9_08001	Sitzungsleiterstisch (245x105x74 cm)	Arbeitsplätze	08 - Sitzungsleiter		1518	09.12.2011	18088.80	  
AR9_08002	Arbeitsplatz (150x80x72 cm)	Arbeitsplätze	08 - Buchhaltung		1518	09.12.2011	732.80	  
AR9_08003	Arbeitsplatz (200x105x73 cm)	Arbeitsplätze	08 - Büro Leiter		1518	09.12.2011	624.80	  
AR9_08004	Besprechst. (55.5x55x74.5 cm)	Arbeitsplätze	08 - Buchhaltung		1518	09.12.2011	186.80	  
AR9_08005	Arbeitsplatz (150x80x75 cm)	Arbeitsplätze	08 - Büro Technik		1518	09.12.2011	536.80	  
AR9_08006	Arbeitsplatz (150x80x75 cm)	Arbeitsplätze	08 - Büro Technik		1518	09.12.2011	536.80	  
DRK_00001	HP 500 (Hauptdrucker)	Drucker	08 - Technik Wartebst.		1508	01.01.2010	1280.80	  

1'200.00



3. Alle Felder die man Bearbeiten möchte in der Bearbeitungsmaske ändern, ein grüner Rahmen bedeutet, dass das Feld korrekt ausgefüllt wurde. Nach dem Bearbeiten auf den Knopf «Speichern» klicken.

Bearbeiten

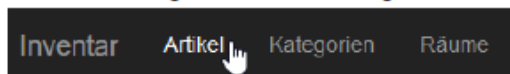
Artikel

Inventar Nummer	DRK_00001
Bezeichnung	HP 200
Beschreibung	Hauptdrucker
Kategorie	Drucker
Raum	Buchhaltung
Nutzer	Hans Mustermann
Beschaffungsdatum	22.03.2018
Beschaffungswert (in CHF)	1'200.00 CHF
<input type="button" value="Speichern"/>	

[Zurück](#)


Artikel Betrachten

1. In der oberen Navigation auf «Artikel» navigieren.



2. In der Tabelle beim jeweiligen Artikel auf den «Info Icon» klicken.

Inventar Nummer	Artikel	Kategorie	Raum	Nutzer	FiBu Konto	Beschaffungsdatum	Beschaffungswert (in CHF)	
ARB_0901	Sitzungsstierchen (24x18x74 cm)	Arbeitsstühle	OG - Sitzungsstimmer		1518	09.12.2011	16'800.00	
ARB_0902	Arbeitsplatz (180x90x72 cm)	Arbeitsplätze	OG - Buchhaltung		1518	09.12.2011	732.00	
ARB_0903	Arbeitsplatz (200x100x73 cm)	Arbeitsplätze	EG - Büro Leer		1518	09.12.2011	524.00	
ARB_0904	Beistelltisch (55.5x50x64.5 cm)	Arbeitsplätze	OG - Buchhaltung		1518	09.12.2011	100.00	
ARB_0905	Arbeitsplatz (180x90x75 cm)	Arbeitsplätze	EG - Büro Technik		1518	09.12.2011	538.00	
ARB_0906	Arbeitsplatz (180x90x75 cm)	Arbeitsplätze	EG - Büro Technik		1518	09.12.2011	538.00	
DRK_0001	HP 500 (Hauptdrucker)	Drucker	EG - Technik Werkstatt		1520	01.01.2010	1'200.00	

1'200.00   

3. Alle Informationen über den Artikel werden nun angezeigt.

Details

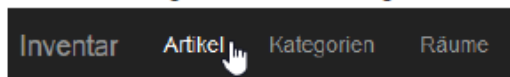
Artikel

Inventar Nummer	DRK_00001
Bezeichnung	HP 500
Beschreibung	Hauptdrucker
Bezeichnung	Drucker
Nutzer	Hans Mustermann
Raum	EG - Technik Werkstatt
FiBu Konto	1520
Beschaffungsdatum	01.01.2010
Beschaffungswert (in CHF)	CHF 1'200.00

[Bearbeiten](#) | [Zurück](#)

Artikel Löschen

1. In der oberen Navigation auf «Artikel» navigieren.



2. In der Tabelle beim jeweiligen Artikel auf den «Löschen Icon» klicken

Inventar Nummer	Artikel	Kategorie	Raum	Nutzer	FiBu Konto	Beschaffungsdatum	Beschaffungswert (in CHF)	
ARB_0001	Sitzungsleiterfach (248x188x74 cm)	Arbeitsplätze	OG - Sitzungsraum		1510	06.12.2011	1090.00	 
ARB_0002	Arbeitsplatz (188x88x73 cm)	Arbeitsplätze	OG - Buchhaltung		1510	06.12.2011	732.00	 
ARB_0003	Arbeitsplatz (208x188x73 cm)	Arbeitsplätze	EG - Büro Leer		1510	06.12.2011	524.00	 
ARB_0004	Beistelltisch (50.5x50x4.5 cm)	Arbeitsplätze	OG - Buchhaltung		1510	06.12.2011	106.00	 
ARB_0005	Arbeitsplatz (188x88x73 cm)	Arbeitsplätze	EG - Büro Technik		1510	06.12.2011	536.00	 
ARB_0006	Arbeitsplatz (188x88x73 cm)	Arbeitsplätze	EG - Büro Technik		1510	06.12.2011	536.00	 
DRK_0001	HP 500 (Hauptdrucker)	Drucker	EG - Technik Werkstatt		1520	01.01.2010	1'200.00	 

1'200.00  

3. Auf den Knopf «Löschen» klicken.

Löschen

Sind Sie sicher, dass Sie den Artikel löschen wollen?

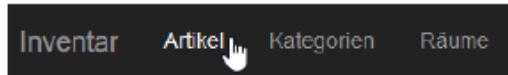
Artikel

Inventar Nummer DRK_00001
Bezeichnung HP 500
Beschreibung Hauptdrucker
Bezeichnung Drucker
Nutzer Hans Mustermann
Bezeichnung Technik Werkstatt
Lokalisierung EG
FiBu Konto 1520
Beschaffungsdatum 01.01.2010
Beschaffungswert (in CHF) CHF 1'200.00

Löschen | [Zurück](#)

Artikel Suchen

1. In der oberen Navigation auf «Artikel» navigieren.



2. In die Felder die Werte eintragen nach denen man suchen will und auf den Knopf «Suchen» klicken.

3. Suchresultate werden angezeigt und können ins Excel oder PDF exportiert werden.

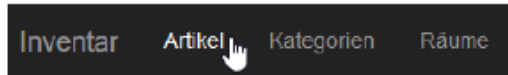
Inventar Nummer	Artikel	Kategorie	Raum	Nutzer	Fibu Konto	Beschaffungsdatum	Beschaffungswert (in CHF)
DRK_00001	HP 500 (Hauptdrucker)	Drucker	EG - Technik Werkstatt	Hans Muslermann	1528	31.01.2010	1'200.00

1

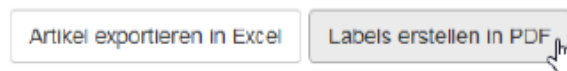
Artikel exportieren in Excel Labels erstellen in PDF

PDF Exportieren

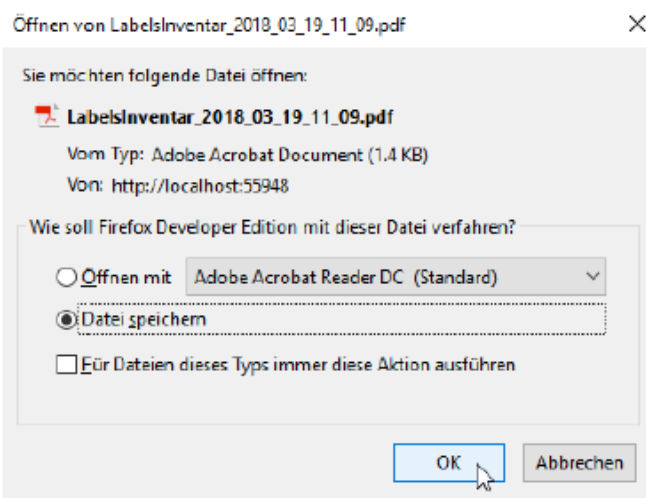
1. In der oberen Navigation auf «Artikel» navigieren.



2. Unten auf den Knopf «Labels erstellen in PDF» klicken.

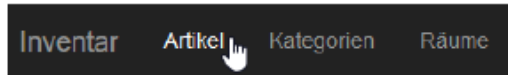


3. Die PDF Datei anschliessend speichern.



Excel Exportieren

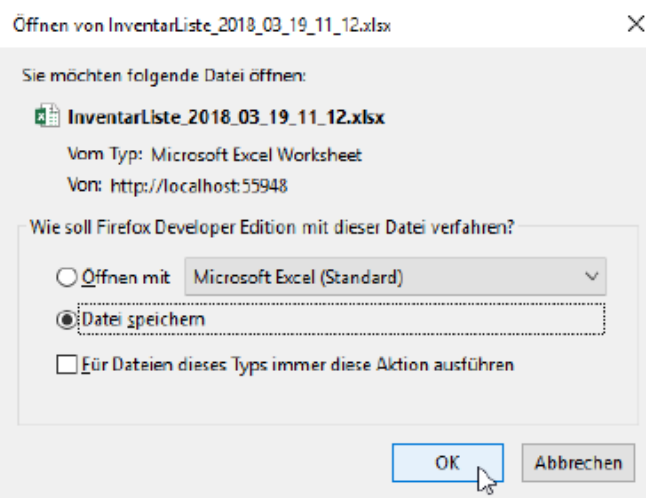
1. In der oberen Navigation auf «Artikel» navigieren.



2. Unten auf den Knopf «Artikel exportieren in Excel» klicken.



3. Die Excel Datei anschliessend speichern.



MSSQL

Dbo.Artikel

```
USE [ch.muster.se.inv]
GO
```

```
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
```

```
CREATE TABLE [dbo].[Artikel](
    [ID] [int] IDENTITY(1,1) NOT NULL,
    [KategorieID] [int] NOT NULL,
    [RaumID] [int] NOT NULL,
    [BenutzerIDNutzer] [int] NOT NULL,
    [InventarNummer] [nvarchar](50) NOT NULL,
    [Bezeichnung] [nvarchar](50) NOT NULL,
    [Beschreibung] [nvarchar](255) NULL,
    [Beschaffungswert] [decimal](18, 2) NOT NULL,
    [Beschaffungsdatum] [date] NOT NULL,
    CONSTRAINT [PK_Artikel] PRIMARY KEY CLUSTERED
(
    [ID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO
```

Dbo.Benutzer

```
USE [ch.muster.se.inv]
GO
```

```
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
```

```
CREATE TABLE [dbo].[Benutzer](
    [ID] [int] IDENTITY(1,1) NOT NULL,
    [RolleID] [int] NOT NULL,
    [Benutzername] [nvarchar](50) NOT NULL,
    [Vorname] [nvarchar](50) NOT NULL,
    [Name] [nvarchar](50) NOT NULL,
    [EMail] [nvarchar](255) NOT NULL,
    [TelefonIntern] [nvarchar](50) NULL,
    CONSTRAINT [PK_Benutzer] PRIMARY KEY CLUSTERED
(
    [ID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO
```

Dbo.FiBuKonto

```
USE [ch.muster.se.inv]
GO
```

```
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
```

```
CREATE TABLE [dbo].[FiBuKonto](
    [ID] [int] IDENTITY(1,1) NOT NULL,
    [Nummer] [nvarchar](50) NOT NULL,
    [Bezeichnung] [nvarchar](255) NOT NULL,
    CONSTRAINT [PK_FiBuKonto] PRIMARY KEY CLUSTERED
(
    [ID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO
```

Dbo.Kategorie

```
USE [ch.muster.se.inv]
GO
```

```
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
```

```
CREATE TABLE [dbo].[Kategorie](
    [ID] [int] IDENTITY(1,1) NOT NULL,
    [FiBuKontoID] [int] NOT NULL,
    [Bezeichnung] [nvarchar](255) NOT NULL,
    CONSTRAINT [PK_Kategorie] PRIMARY KEY CLUSTERED
(
    [ID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO
```

Dbo.Raum

```
USE [ch.muster.se.inv]
GO
```

```
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
```

```
CREATE TABLE [dbo].[Raum](
    [ID] [int] IDENTITY(1,1) NOT NULL,
    [Lokalisierung] [nvarchar](50) NOT NULL,
    [Bezeichnung] [nvarchar](255) NOT NULL,
    CONSTRAINT [PK_Raum] PRIMARY KEY CLUSTERED
(
    [ID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO
```

Dbo.Rolle

```
USE [ch.muster.se.inv]
GO
```

```
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
```

```
CREATE TABLE [dbo].[Rolle](
    [ID] [int] IDENTITY(1,1) NOT NULL,
    [Name] [nvarchar](50) NULL,
    CONSTRAINT [PK_Rolle] PRIMARY KEY CLUSTERED
(
    [ID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO
```

C#

```
ch.muster.se.inv.bll.Interfaces.IArtikelService
```

```
using ch.muster.se.inv.dal.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace ch.muster.se.inv.bll.Interfaces
{
    public interface IArtikelService
```

```

    {
        //Create
        Artikel CreateArtikel(Artikel Artikel);

        //Read
        Artikel GetArtikel(int? id);
        IEnumerable<Artikel> GetArtikels();
        IEnumerable<Artikel> GetArtikels(ArtikelSearchModel artikelSearchModel);
        //Update
        Artikel EditArtikel(Artikel Artikel);

        //Delete
        void DeleteArtikel(int? id);
    }
}

```

```
ch.muster.se.inv.bll.Interfaces.IBenutzerService
```

```

using ch.muster.se.inv.dal.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ch.muster.se.inv.bll.Interfaces
{
    public interface IBenutzerService
    {
        //Create
        Benutzer CreateBenutzer(Benutzer Benutzer);

        //Read
        Benutzer GetBenutzer(int? id);
        IEnumerable<Benutzer> GetBenutzers();
        Benutzer GetBenutzerByBenutzername(string Benutzername);

        //Update
        Benutzer EditBenutzer(Benutzer Benutzer);

        //Delete
        void DeleteBenutzer(int? id);
    }
}

```

```
ch.muster.se.inv.bll.Interfaces.IFiBuKontoService
```

```

using ch.muster.se.inv.dal.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ch.muster.se.inv.bll.Interfaces
{
    public interface IBenutzerService
    {
        //Create
        Benutzer CreateBenutzer(Benutzer Benutzer);

        //Read
        Benutzer GetBenutzer(int? id);
        IEnumerable<Benutzer> GetBenutzers();
        Benutzer GetBenutzerByBenutzername(string Benutzername);

        //Update
        Benutzer EditBenutzer(Benutzer Benutzer);

        //Delete
        void DeleteBenutzer(int? id);
    }
}

```

```
ch.muster.se.inv.bll.Interfaces.IKategorieService
```

```

using ch.muster.se.inv.dal.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ch.muster.se.inv.bll.Interfaces
{
    public interface IKategorieService
    {
        //Create

```

```

        Kategorie CreateKategorie(Kategorie Kategorie);

        //Read
        Kategorie GetKategorie(int? id);
        IEnumerable<Kategorie> GetKategories();

        //Update
        Kategorie EditKategorie(Kategorie Kategorie);

        //Delete
        void DeleteKategorie(int? id);
    }
}

```

```
ch.muster.se.inv.bll.Interfaces.IRaumService
```

```

using ch.muster.se.inv.dal.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ch.muster.se.inv.bll.Interfaces
{
    public interface IRaumService
    {
        //Create
        Raum CreateRaum(Raum Raum);

        //Read
        Raum GetRaum(int? id);
        IEnumerable<Raum> GetRaums();

        //Update
        Raum EditRaum(Raum Raum);

        //Delete
        void DeleteRaum(int? id);
    }
}

```

```
ch.muster.se.inv.bll.Interfaces.IRolleService
```

```

using ch.muster.se.inv.dal.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ch.muster.se.inv.bll.Interfaces
{
    public interface IRolleService
    {
        //Create
        Rolle CreateRolle(Rolle Rolle);

        //Read
        Rolle GetRolle(int? id);
        IEnumerable<Rolle> GetRollen();

        //Update
        Rolle EditRolle(Rolle Rolle);

        //Delete
        void DeleteRolle(int? id);
    }
}

```

```
ch.muster.se.inv.bll.Services.ArtikelService
```

```

using ch.muster.se.inv.bll.Interfaces;
using ch.muster.se.inv.dal.Accessors;
using ch.muster.se.inv.dal.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ch.muster.se.inv.bll.Services
{
    public class ArtikelService : IArtikelService
    {
        private ArtikelAccessor _artikelAccessor;

        public ArtikelService()

```

```

    {
        _artikelAccessor = new ArtikelAccessor();
    }

    public IEnumerable<Artikel> GetArtikels()
    {
        return _artikelAccessor.GetArtikels();
    }

    public IEnumerable<Artikel> GetArtikels(ArtikelSearchModel artikelSearchModel)
    {
        return _artikelAccessor.GetArtikels(artikelSearchModel);
    }

    public Artikel GetArtikel(int? id)
    {
        return _artikelAccessor.GetArtikel(id);
    }

    public Artikel CreateArtikel(Artikel artikel)
    {
        return _artikelAccessor.SaveArtikel(artikel);
    }

    public Artikel EditArtikel(Artikel artikel)
    {
        return _artikelAccessor.SaveArtikel(artikel);
    }

    public void DeleteArtikel(int? id)
    {
        _artikelAccessor.DeleteArtikel(id);
    }
}

```

```
ch.muster.se.inv.bll.Services.BenutzerService
```

```

using ch.muster.se.inv.bll.Interfaces;
using ch.muster.se.inv.dal.Accessors;
using ch.muster.se.inv.dal.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

```

```
namespace ch.muster.se.inv.bll.Services
```

```

{
    public class BenutzerService : IBenutzerService
    {
        private BenutzerAccessor _benutzerAccessor;

        public BenutzerService()
        {
            _benutzerAccessor = new BenutzerAccessor();
        }

        public IEnumerable<Benutzer> GetBenutzers()
        {
            return _benutzerAccessor.GetBenutzers();
        }

        public Benutzer GetBenutzer(int? id)
        {
            return _benutzerAccessor.GetBenutzer(id);
        }

        public Benutzer GetBenutzerByBenutzername(string Benutzername)
        {
            try
            {
                return _benutzerAccessor.GetBenutzerByBenutzername(Benutzername);
            }
            catch (Exception ex)
            {
                return null;
            }
        }

        public Benutzer CreateBenutzer(Benutzer benutzer)
        {
            return _benutzerAccessor.SaveBenutzer(benutzer);
        }

        public Benutzer EditBenutzer(Benutzer benutzer)
        {
            return _benutzerAccessor.SaveBenutzer(benutzer);
        }
    }
}

```

```

    }

    public void DeleteBenutzer(int? id)
    {
        _benutzerAccessor.DeleteBenutzer(id);
    }
}

ch.muster.se.inv.bll.Services.FiBuKontoService
using ch.muster.se.inv.bll.Interfaces;
using ch.muster.se.inv.dal.Accessors;
using ch.muster.se.inv.dal.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ch.muster.se.inv.bll.Services
{
    public class FiBuKontoService : IFiBuKontoService
    {
        private FiBuKontoAccessor _fiBuKontoAccessor;

        public FiBuKontoService()
        {
            _fiBuKontoAccessor = new FiBuKontoAccessor();
        }

        public IEnumerable<FiBuKonto> GetFiBuKontos()
        {
            return _fiBuKontoAccessor.GetFiBuKontos();
        }

        public FiBuKonto GetFiBuKonto(int? id)
        {
            return _fiBuKontoAccessor.GetFiBuKonto(id);
        }

        public FiBuKonto CreateFiBuKonto(FiBuKonto FiBuKonto)
        {
            return _fiBuKontoAccessor.SaveFiBuKonto(FiBuKonto);
        }

        public FiBuKonto EditFiBuKonto(FiBuKonto FiBuKonto)
        {
            return _fiBuKontoAccessor.SaveFiBuKonto(FiBuKonto);
        }

        public void DeleteFiBuKonto(int? id)
        {
            _fiBuKontoAccessor.DeleteFiBuKonto(id);
        }
    }
}

```

```

ch.muster.se.inv.bll.Services.KategorieService
using ch.muster.se.inv.bll.Interfaces;
using ch.muster.se.inv.dal.Accessors;
using ch.muster.se.inv.dal.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ch.muster.se.inv.bll.Services
{
    public class KategorieService : IKategorieService
    {
        private KategorieAccessor _kategorieAccessor;

        public KategorieService()
        {
            _kategorieAccessor = new KategorieAccessor();
        }

        public IEnumerable<Kategorie> GetKategories()
        {
            return _kategorieAccessor.GetKategories();
        }

        public Kategorie GetKategorie(int? id)

```

```

    {
        return _kategorieAccessor.GetKategorie(id);
    }

    public Kategorie CreateKategorie(Kategorie Kategorie)
    {
        return _kategorieAccessor.SaveKategorie(Kategorie);
    }

    public Kategorie EditKategorie(Kategorie Kategorie)
    {
        return _kategorieAccessor.SaveKategorie(Kategorie);
    }

    public void DeleteKategorie(int? id)
    {
        _kategorieAccessor.DeleteKategorie(id);
    }
}

```

```

ch.muster.se.inv.bll.Services.RaumService
using ch.muster.se.inv.bll.Interfaces;
using ch.muster.se.inv.dal.Accessors;
using ch.muster.se.inv.dal.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ch.muster.se.inv.bll.Services
{
    public class RaumService : IRaumService
    {
        private RaumAccessor _raumAccessor;

        public RaumService()
        {
            _raumAccessor = new RaumAccessor();
        }

        public IEnumerable<Raum> GetRaums()
        {
            return _raumAccessor.GetRaums();
        }

        public Raum GetRaum(int? id)
        {
            return _raumAccessor.GetRaum(id);
        }

        public Raum CreateRaum(Raum Raum)
        {
            return _raumAccessor.SaveRaum(Raum);
        }

        public Raum EditRaum(Raum Raum)
        {
            return _raumAccessor.SaveRaum(Raum);
        }

        public void DeleteRaum(int? id)
        {
            _raumAccessor.DeleteRaum(id);
        }
    }
}

```

```

ch.muster.se.inv.bll.Services.RolleService
using ch.muster.se.inv.bll.Interfaces;
using ch.muster.se.inv.dal.Accessors;
using ch.muster.se.inv.dal.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ch.muster.se.inv.bll.Services
{
    public class RolleService : IRolleService
    {

```

```

        private RolleAccessor _rolleAccessor;

        public RolleService()
        {
            _rolleAccessor = new RolleAccessor();
        }

        public IEnumerable<Rolle> GetRolles()
        {
            return _rolleAccessor.GetRolles();
        }

        public Rolle GetRolle(int? id)
        {
            return _rolleAccessor.GetRolle(id);
        }

        public Rolle CreateRolle(Rolle Rolle)
        {
            return _rolleAccessor.SaveRolle(Rolle);
        }

        public Rolle EditRolle(Rolle Rolle)
        {
            return _rolleAccessor.SaveRolle(Rolle);
        }

        public void DeleteRolle(int? id)
        {
            _rolleAccessor.DeleteRolle(id);
        }
    }
}

```

ch.muster.se.inv.dal.Accessors.ArtikelAccessor

```

using ch.muster.se.inv.dal.Interfaces;
using ch.muster.se.inv.dal.Models;
using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ch.muster.se.inv.dal.Accessors
{
    public class ArtikelAccessor : IArtikelAccessor
    {
        private Entities db = new Entities();

        public IEnumerable<Artikel> GetArtikels()
        {
            return db.Artikels.Include(a => a.Kategorie).Include(a => a.Nutzer).Include(a => a.Raum);
        }

        public Artikel GetArtikel(int? id)
        {
            return db.Artikels.Find(id);
        }

        public IEnumerable<Artikel> GetArtikels(ArtikelSearchModel artikelSearchModel)
        {
            IEnumerable<Artikel> artikellist = db.Artikels.Include(a => a.Kategorie).Include(a => a.Nutzer).Include(a
=> a.Raum);

            if (!String.IsNullOrEmpty(artikelSearchModel.Search))
            {
                artikelSearchModel.Search = artikelSearchModel.Search.ToLower();
                foreach (string word in artikelSearchModel.Search.Split(' '))
                {
                    artikellist = artikellist.Where(a => a.InventarNummer.ToLower().Contains(word) ||
a.Bezeichnung.ToLower().Contains(word) || a.Beschreibung.ToLower().StartsWith(word));
                }
            }

            if (artikelSearchModel.KategorieID.HasValue)
            {
                artikellist = artikellist.Where(c => c.KategorieID.Equals(artikelSearchModel.KategorieID.Value));
            }
            if (artikelSearchModel.FiBuKontoID.HasValue)
            {
                artikellist = artikellist.Where(c =>
c.Kategorie.FiBuKontoID.Equals(artikelSearchModel.FiBuKontoID.Value));
            }
        }
    }
}

```



```

    }
    if (artikelSearchModel.RaumID.HasValue)
    {
        artikellist = artikellist.Where(c => c.RaumID.Equals(artikelSearchModel.RaumID.Value));
    }
    if (artikelSearchModel.BenutzerID.HasValue)
    {
        artikellist = artikellist.Where(c => c.BenutzerIDNutzer.Equals(artikelSearchModel.BenutzerID.Value));
    }
    if (artikelSearchModel.ArtikelYearsStart.HasValue)
    {
        artikellist = artikellist.Where(c => c.Beschaffungsdatum.Date >= new
DateTime(int.Parse(artikelSearchModel.ArtikelYearsStart.Value.ToString()), 1, 1));
    }
    if (artikelSearchModel.ArtikelYearsEnd.HasValue)
    {
        artikellist = artikellist.Where(c => c.Beschaffungsdatum.Date <= new
DateTime(int.Parse((artikelSearchModel.ArtikelYearsEnd.Value + 1).ToString()), 1, 1)).ToList();
    }

    return artikellist;
}

public Artikel SaveArtikel(Artikel Artikel)
{
    if (db.Artikels.Any(a => a.ID == Artikel.ID))
    {
        db.Entry(Artikel).State = EntityState.Modified;
    }
    else
    {
        db.Artikels.Add(Artikel);
    }
    db.SaveChanges();

    return db.Artikels.Find(Artikel.ID);
}

public void DeleteArtikel(int? id)
{
    Artikel Artikel = db.Artikels.Find(id);
    db.Artikels.Remove(Artikel);
    db.SaveChanges();
}
}
}

```

ch.muster.se.inv.dal.Accessors.BenutzerAccessor

```

using ch.muster.se.inv.dal.Interfaces;
using ch.muster.se.inv.dal.Models;
using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ch.muster.se.inv.dal.Accessors
{
    public class BenutzerAccessor : IBenutzerAccessor
    {
        private Entities db = new Entities();

        public IEnumerable<Benutzer> GetBenutzers()
        {
            return db.Benutzers.Include(b => b.Rolle);
        }

        public Benutzer GetBenutzer(int? id)
        {
            return db.Benutzers.Find(id);
        }

        public Benutzer GetBenutzerByBenutzername(string Benutzername)
        {
            return db.Benutzers.Where(b => b.Benutzername == Benutzername).First(); // Es nimmt das Erste, was den
jeweiligen Benutzernamen hat, es gibt nicht zwei AD-Benutzer die den gleichen Benutzernamen haben
        }

        public Benutzer SaveBenutzer(Benutzer Benutzer)
        {
            if (db.Benutzers.Any(a => a.ID == Benutzer.ID))
            {
                db.Entry(Benutzer).State = EntityState.Modified;
            }
        }
    }
}

```

```

    }
    else
    {
        db.Benutzers.Add(Benutzer);
    }
    db.SaveChanges();

    return db.Benutzers.Find(Benutzer.ID);
}

public void DeleteBenutzer(int? id)
{
    Benutzer Benutzer = db.Benutzers.Find(id);
    db.Benutzers.Remove(Benutzer);
    db.SaveChanges();
}
}
}

```

```

ch.muster.se.inv.dal.Accessors.FiBuKontoAccessor
using ch.muster.se.inv.dal.Interfaces;
using ch.muster.se.inv.dal.Models;
using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ch.muster.se.inv.dal.Accessors
{
    public class FiBuKontoAccessor : IFiBuKontoAccessor
    {
        private Entities db = new Entities();

        public IEnumerable<FiBuKonto> GetFiBuKontos()
        {
            return db.FiBuKontoes;
        }

        public FiBuKonto GetFiBuKonto(int? id)
        {
            return db.FiBuKontoes.Find(id);
        }

        public FiBuKonto SaveFiBuKonto(FiBuKonto FiBuKonto)
        {
            if (db.FiBuKontoes.Any(a => a.ID == FiBuKonto.ID))
            {
                db.Entry(FiBuKonto).State = EntityState.Modified;
            }
            else
            {
                db.FiBuKontoes.Add(FiBuKonto);
            }
            db.SaveChanges();

            return db.FiBuKontoes.Find(FiBuKonto.ID);
        }

        public void DeleteFiBuKonto(int? id)
        {
            FiBuKonto FiBuKonto = db.FiBuKontoes.Find(id);
            db.FiBuKontoes.Remove(FiBuKonto);
            db.SaveChanges();
        }
    }
}

```

```

ch.muster.se.inv.dal.Accessors.KategorieAccessor
using ch.muster.se.inv.dal.Interfaces;
using ch.muster.se.inv.dal.Models;
using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ch.muster.se.inv.dal.Accessors
{
    public class FiBuKontoAccessor : IFiBuKontoAccessor
    {

```

```

private Entities db = new Entities();

public IEnumerable<FiBuKonto> GetFiBuKontos()
{
    return db.FiBuKontoes;
}

public FiBuKonto GetFiBuKonto(int? id)
{
    return db.FiBuKontoes.Find(id);
}

public FiBuKonto SaveFiBuKonto(FiBuKonto FiBuKonto)
{
    if (db.FiBuKontoes.Any(a => a.ID == FiBuKonto.ID))
    {
        db.Entry(FiBuKonto).State = EntityState.Modified;
    }
    else
    {
        db.FiBuKontoes.Add(FiBuKonto);
    }
    db.SaveChanges();

    return db.FiBuKontoes.Find(FiBuKonto.ID);
}

public void DeleteFiBuKonto(int? id)
{
    FiBuKonto FiBuKonto = db.FiBuKontoes.Find(id);
    db.FiBuKontoes.Remove(FiBuKonto);
    db.SaveChanges();
}
}
}

```

```

ch.muster.se.inv.dal.Accessors.RaumAccessor
using ch.muster.se.inv.dal.Interfaces;
using ch.muster.se.inv.dal.Models;
using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ch.muster.se.inv.dal.Accessors
{
    public class RaumAccessor : IRaumAccessor
    {
        private Entities db = new Entities();

        public IEnumerable<Raum> GetRaums()
        {
            return db.Raums;
        }

        public Raum GetRaum(int? id)
        {
            return db.Raums.Find(id);
        }

        public Raum SaveRaum(Raum Raum)
        {
            if (db.Raums.Any(a => a.ID == Raum.ID))
            {
                db.Entry(Raum).State = EntityState.Modified;
            }
            else
            {
                db.Raums.Add(Raum);
            }
            db.SaveChanges();

            return db.Raums.Find(Raum.ID);
        }

        public void DeleteRaum(int? id)
        {
            Raum Raum = db.Raums.Find(id);
            db.Raums.Remove(Raum);
            db.SaveChanges();
        }
    }
}

```

```
}
```

```
ch.muster.se.inv.dal.Accessors.RolleAccessor
using ch.muster.se.inv.dal.Interfaces;
using ch.muster.se.inv.dal.Models;
using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ch.muster.se.inv.dal.Accessors
{
    public class RolleAccessor : IRolleAccessor
    {
        private Entities db = new Entities();

        public IEnumerable<Rolle> GetRolles()
        {
            return db.Rolles;
        }

        public Rolle GetRolle(int? id)
        {
            return db.Rolles.Find(id);
        }

        public Rolle SaveRolle(Rolle Rolle)
        {
            if (db.Rolles.Any(a => a.ID == Rolle.ID))
            {
                db.Entry(Rolle).State = EntityState.Modified;
            }
            else
            {
                db.Rolles.Add(Rolle);
            }
            db.SaveChanges();

            return db.Rolles.Find(Rolle.ID);
        }

        public void DeleteRolle(int? id)
        {
            Rolle Rolle = db.Rolles.Find(id);
            db.Rolles.Remove(Rolle);
            db.SaveChanges();
        }
    }
}

ch.muster.se.inv.dal.Interfaces.IArtikelAccessor
using ch.muster.se.inv.dal.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ch.muster.se.inv.dal.Interfaces
{
    public interface IArtikelAccessor
    {
        //Create & Update
        Artikel SaveArtikel(Artikel Artikel);

        //Read
        IEnumerable<Artikel> GetArtikels();
        IEnumerable<Artikel> GetArtikels(ArtikelSearchModel artikelSearchModel);
        Artikel GetArtikel(int? id);

        //Delete
        void DeleteArtikel(int? id);
    }
}
```

```
ch.muster.se.inv.dal.Interfaces.IBenutzerAccessor
using ch.muster.se.inv.dal.Models;
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ch.muster.se.inv.dal.Interfaces
{
    public interface IBenutzerAccessor
    {
        //Create & Update
        Benutzer SaveBenutzer(Benutzer Benutzer);

        //Read
        IEnumerable<Benutzer> GetBenutzers();
        Benutzer GetBenutzer(int? id);
        Benutzer GetBenutzerByBenutzername(string Benutzername);

        //Delete
        void DeleteBenutzer(int? id);
    }
}
```

```
ch.muster.se.inv.dal.Interfaces.IFiBuKontoAccessor
using ch.muster.se.inv.dal.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ch.muster.se.inv.dal.Interfaces
{
    public interface IFiBuKontoAccessor
    {
        //Create & Update
        FiBuKonto SaveFiBuKonto(FiBuKonto FiBuKonto);

        //Read
        IEnumerable<FiBuKonto> GetFiBuKontos();
        FiBuKonto GetFiBuKonto(int? id);

        //Delete
        void DeleteFiBuKonto(int? id);
    }
}
```

```
ch.muster.se.inv.dal.Interfaces.IKategorieAccessor
using ch.muster.se.inv.dal.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ch.muster.se.inv.dal.Interfaces
{
    public interface IKategorieAccessor
    {
        //Create & Update
        Kategorie SaveKategorie(Kategorie Kategorie);

        //Read
        IEnumerable<Kategorie> GetKategories();
        Kategorie GetKategorie(int? id);

        //Delete
        void DeleteKategorie(int? id);
    }
}
```

```
ch.muster.se.inv.dal.Interfaces.IRaumAccessor
using ch.muster.se.inv.dal.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ch.muster.se.inv.dal.Interfaces
```

```

{
    public interface IRaumAccessor
    {
        //Create & Update
        Raum SaveRaum(Raum Raum);

        //Read
        IEnumerable<Raum> GetRaums();
        Raum GetRaum(int? id);

        //Delete
        void DeleteRaum(int? id);
    }
}

```

```
ch.muster.se.inv.dal.Interfaces.IRolleAccessor
```

```

using ch.muster.se.inv.dal.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

```

```
namespace ch.muster.se.inv.dal.Interfaces
```

```

{
    public interface IRaumAccessor
    {
        //Create & Update
        Raum SaveRaum(Raum Raum);

        //Read
        IEnumerable<Raum> GetRaums();
        Raum GetRaum(int? id);

        //Delete
        void DeleteRaum(int? id);
    }
}

```

```
ch.muster.se.inv.dal.Models.ArtikelSearchModel
```

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

```

```
namespace ch.muster.se.inv.dal.Models
```

```

{
    // Ein SearchModel ist nützlich für grössere Suchabfragen.
    // https://stackoverflow.com/questions/33153932/filter-search-using-multiple-fields-asp-net-mvc
    public class ArtikelSearchModel
    {
        public int? KategorieID { get; set; }
        public int? FiBuKontoID { get; set; }
        public int? RaumID { get; set; }
        public int? BenutzerID { get; set; }
        public int? ArtikelYearsStart { get; set; }
        public int? ArtikelYearsEnd { get; set; }
        public string Search { get; set; }
    }
}

```

```
ch.muster.se.inv.dal.Models.Artikel
```

```

using System;
using System.ComponentModel.DataAnnotations;

```

```
namespace ch.muster.se.inv.dal.Models
```

```

{
    // Die Metadaten der jeweiligen Properties in Artikel sind in der Klasse ArtikelMeta angegeben.
    // Zusätzliche Properties für Artikel wie z.B Preis werden hier angegeben und nicht in der Datenbank.
    [MetadataType(typeof(ArtikelMeta))]
    public partial class Artikel
    {
        public string Preis
        {
            get
            {
                return Beschaffungswert.ToString("N");
            }
        }
    }
}

```

```

    }

    [Display(Name = "Artikel")]
    public string Gegenstand
    {
        get
        {
            return Bezeichnung + " (" + Beschreibung + ")";
        }
    }

    [Display(Name = "Raum")]
    public string Ort
    {
        get
        {
            return Raum.Lokalisierung + " - " + Raum.Bezeichnung;
        }
    }
}

public class ArtikelMeta
{
    [Required]
    [Display(Name = "Kategorie")]
    public int KategorieID { get; set; }

    [Required]
    [Display(Name = "Raum")]
    public int RaumID { get; set; }

    [Required]
    [Display(Name = "Nutzer")]
    public int BenutzerIDNutzer { get; set; }

    [Required]
    [Display(Name = "Inventar Nummer")]
    public string InventarNummer { get; set; }

    [Required]
    [Display(Name = "Bezeichnung")]
    public string Bezeichnung { get; set; }

    [Display(Name = "Beschreibung")]
    public string Beschreibung { get; set; }

    [Required]
    [DataType(DataType.Currency)]
    [Display(Name = "Beschaffungswert (in CHF)")]
    public Nullable Beschaffungswert { get; set; }

    [Display(Name = "Beschaffungsdatum")]
    [Required]
    [DataType(DataType.Date)]
    [DisplayFormat(DataFormatString = "{0:dd.MM.yyyy}")]
    public Nullable<System.DateTime> Beschaffungsdatum { get; set; }
}
}

```

ch.muster.se.inv.dal.Models.Benutzer

```

using System;
using System.ComponentModel.DataAnnotations;

namespace ch.muster.se.inv.dal.Models
{
    // Die Metadaten der jeweiligen Properties in Benutzer sind in der Klasse BenutzerMeta angegeben.
    // Zusätzliche Properties für Benutzer wie z.b Preis werden hier angegeben und nicht in der Datenbank.
    [MetadataType(typeof(BenutzerMeta))]
    public partial class Benutzer
    {
        public string Fullname
        {
            get
            {
                return Vorname + " " + Name;
            }
        }
    }

    public class BenutzerMeta
    {
        [Required]
        [Display(Name = "Rolle")]
        public int RolleID { get; set; }
    }
}

```

```

    [Required]
    [Display(Name = "Benutzername")]
    public string Benutzername { get; set; }

    [Required]
    [Display(Name = "Vorname")]
    public string Vorname { get; set; }

    [Required]
    [Display(Name = "Name")]
    public string Name { get; set; }

    [Required(ErrorMessage = "Die E-Mail Adresse wird benötigt")]
    [Display(Name = "E-Mail")]
    [EmailAddress(ErrorMessage = "Invalide Email Adresse")]
    public string EMail { get; set; }

    [Display(Name = "Telefon Intern")]
    public string TelefonIntern { get; set; }
}

```

ch.muster.se.inv.dal.Models.FiBuKonto

```

using System;
using System.ComponentModel.DataAnnotations;

namespace ch.muster.se.inv.dal.Models
{
    // Die Metadaten der jeweiligen Properties in FiBuKonto sind in der Klasse FiBuKonMaxeta angegeben.
    // Zusätzliche Properties für FiBuKonto wie z.b Preis werden hier angegeben und nicht in der Datenbank.
    [MetadataType(typeof(FiBuKontoMeta))]
    public partial class FiBuKonto
    {
    }

    public class FiBuKontoMeta
    {
        [Required]
        [Display(Name = "FiBu Konto")]
        public string Nummer { get; set; }

        [Required]
        [Display(Name = "Bezeichnung")]
        public string Bezeichnung { get; set; }
    }
}

```

ch.muster.se.inv.dal.Models.Kategorie

```

using System;
using System.ComponentModel.DataAnnotations;

namespace ch.muster.se.inv.dal.Models
{
    // Die Metadaten der jeweiligen Properties in Kategorie sind in der Klasse KategorieMeta angegeben.
    // Zusätzliche Properties für Kategorie wie z.b Preis werden hier angegeben und nicht in der Datenbank.
    [MetadataType(typeof(KategorieMeta))]
    public partial class Kategorie
    {
    }

    public class KategorieMeta
    {
        [Required]
        [Display(Name = "FiBu Konto")]
        public int FiBuKontoID { get; set; }

        [Required]
        [Display(Name = "Bezeichnung")]
        public string Bezeichnung { get; set; }
    }
}

```

ch.muster.se.inv.dal.Models.Raum

```

using System;
using System.ComponentModel.DataAnnotations;

namespace ch.muster.se.inv.dal.Models
{
    // Die Metadaten der jeweiligen Properties in Raum sind in der Klasse RaumMeta angegeben.

```



```

// Zusätzliche Properties für Raum wie z.b Preis werden hier angegeben und nicht in der Datenbank.
[MetadataType(typeof(RaumMeta))]
public partial class Raum
{
}

public class RaumMeta
{
    [Required]
    [Display(Name = "Lokalisierung")]
    public string Lokalisierung { get; set; }

    [Required]
    [Display(Name = "Bezeichnung")]
    public string Bezeichnung { get; set; }
}

```

```

ch.muster.se.inv.dal.Models.Rolle
using System;
using System.ComponentModel.DataAnnotations;

namespace ch.muster.se.inv.dal.Models
{
    // Die Metadaten der jeweiligen Properties in Rolle sind in der Klasse RolleMeta angegeben.
    // Zusätzliche Properties für Rolle wie z.b Preis werden hier angegeben und nicht in der Datenbank.
    [MetadataType(typeof(RolleMeta))]
    public partial class Rolle
    {
    }

    public class RolleMeta
    {
        [Required]
        [Display(Name = "Rolle")]
        public string Name { get; set; }
    }
}

```

```

ch.muster.se.inv.web.BundleConfig
using System.Web;
using System.Web.Optimization;

namespace ch.muster.se.inv.web
{
    public class BundleConfig
    {
        // For more information on bundling, visit http://go.microsoft.com/fwlink/?LinkId=301862
        public static void RegisterBundles(BundleCollection bundles)
        {
            bundles.Add(new ScriptBundle("~/bundles/jquery").Include(
                "~/Scripts/jquery-{version}.js"));

            bundles.Add(new ScriptBundle("~/bundles/jqueryval").Include(
                "~/Scripts/jquery.validate*"));

            // Use the development version of Modernizr to develop with and learn from. Then, when you're
            // ready for production, use the build tool at http://modernizr.com to pick only the tests you need.
            bundles.Add(new ScriptBundle("~/bundles/modernizr").Include(
                "~/Scripts/modernizr-*"));

            bundles.Add(new ScriptBundle("~/bundles/bootstrap").Include(
                "~/Scripts/bootstrap.js",
                "~/Scripts/respond.js"));

            bundles.Add(new ScriptBundle("~/bundles/autonumeric").Include(
                "~/Scripts/autoNumeric.js"));

            bundles.Add(new StyleBundle("~/Content/css").Include(
                "~/Content/bootstrap.css",
                "~/Content/site.css"));
        }
    }
}

```

```

ch.muster.se.inv.web.FilterConfig
using System.Web;
using System.Web.Mvc;

```

```

namespace ch.muster.se.inv.web
{
    public class FilterConfig
    {
        public static void RegisterGlobalFilters(GlobalFilterCollection filters)
        {
            filters.Add(new HandleErrorAttribute());
            // Mit diesem Filter wird angegeben das alle Controller nur im eingeloggten Zustand zugreifbar sein
            dürfen.
            filters.Add(new AuthorizeAttribute());
        }
    }
}

```

```

ch.muster.se.inv.web.RouteConfig
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Routing;

namespace ch.muster.se.inv.web
{
    public class RouteConfig
    {
        public static void RegisterRoutes(RouteCollection routes)
        {
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

            routes.MapRoute(
                name: "Default",
                url: "{controller}/{action}/{id}",
                defaults: new { controller = "Artikel", action = "Index", id = UrlParameter.Optional }
            );
        }
    }
}

```

```

ch.muster.se.inv.web.UnityConfig
using System.Web.Mvc;
using Microsoft.Practices.Unity;
using Unity.Mvc5;
using ch.muster.se.inv.dal.Interfaces;
using ch.muster.se.inv.dal.Accessors;
using ch.muster.se.inv.bll.Interfaces;
using ch.muster.se.inv.bll.Services;

namespace ch.muster.se.inv.web
{
    public static class UnityConfig
    {
        public static void RegisterComponents()
        {
            var container = new UnityContainer();

            // DAL
            container.RegisterType<IArtikelAccessor, ArtikelAccessor>();
            container.RegisterType<IBenutzerAccessor, BenutzerAccessor>();
            container.RegisterType<IFiBuKontoAccessor, FiBuKontoAccessor>();
            container.RegisterType<IKategorieAccessor, KategorieAccessor>();
            container.RegisterType<IRaumAccessor, RaumAccessor>();
            container.RegisterType<IRolleAccessor, RolleAccessor>();

            // BLL
            container.RegisterType<IArtikelService, ArtikelService>();
            container.RegisterType<IBenutzerService, BenutzerService>();
            container.RegisterType<IFiBuKontoService, FiBuKontoService>();
            container.RegisterType<IKategorieService, KategorieService>();
            container.RegisterType<IRaumService, RaumService>();
            container.RegisterType<IRolleService, RolleService>();

            DependencyResolver.SetResolver(new UnityDependencyResolver(container));
        }
    }
}

```

```

ch.muster.se.inv.web.Controllers.AccountController
using ch.muster.se.inv.bll.Interfaces;
using ch.muster.se.inv.dal.Models;
using ch.muster.se.inv.web.Models;
using System;
using System.Collections.Generic;
using System.Linq;

```

```

using System.Security.Claims;
using System.Web;
using System.Web.Mvc;

namespace ch.muster.se.inv.web.Controllers
{
    // Zum Teil wurden Methoden aus den Klassen der jeweiligen Tutorials übernommen für die Implementierung von OWIN
    // und Role Authentication
    // http://benfoster.io/blog/aspnet-identity-stripped-bare-mvc-part-1
    // http://benfoster.io/blog/asp-net-identity-role-claims

    [AllowAnonymous]
    public class AccountController : Controller
    {
        IBenutzerService _benutzerService;

        public AccountController(IBenutzerService benutzerService)
        {
            _benutzerService = benutzerService;
        }

        public ActionResult Index()
        {
            return RedirectToAction("Login");
        }

        public ActionResult Login(string returnUrl)
        {
            Account model = new Account
            {
                ReturnUrl = returnUrl
            };

            return View(model);
        }

        [HttpPost]
        public ActionResult Login(Account model)
        {
            if (!ModelState.IsValid)
            {
                return View(model);
            }

            // Es wird hier überprüft ob die Login Daten korrekt sind
            if (model.Login())
            {
                // Hier wird abgefragt ob es in der Benutzer Tabelle schon einen Benutzer gibt mit dem jeweiligen
                // Benutzernamen, falls nicht, wird ein null zurückgegeben
                Benutzer benutzer = _benutzerService.GetBenutzerByBenutzername(model.Username);

                // Wenn der Benutzer nicht in der Datenbank vorhanden ist, wird er auf das hingewiesen
                if (benutzer == null)
                {
                    // Der Benutzer wird darauf hingewiesen, das er einen Account braucht
                    ModelState.AddModelError(string.Empty, "Login fehlgeschlagen, bitte überprüfen Sie ihre
Eingabe.");

                    return View(model);
                }

                // Dieser Codeabschnitt wurde übernommen aus dem oben genannten Tutorial und gemäss dem in der
                // Dokumentation angegebenen File angepasst
                var identity = new ClaimsIdentity(new[] {
                    new Claim(ClaimTypes.Name, model.FullName),
                    new Claim(ClaimTypes.NameIdentifier, model.Username),
                    new Claim(ClaimTypes.Role, benutzer.Rolle.Name) // Die Rolle wird hier in der Identity
                    gespeichert, vorübergehend bis die Sitzung beendet ist
                }, "ApplicationCookie");

                var ctx = Request.GetOwinContext();
                var authManager = ctx.Authentication;

                authManager.SignIn(identity);

                return Redirect(GetRedirectUrl(model.ReturnUrl));
            }

            ModelState.AddModelError("", "Ungültige Benutzerdaten");
            return View(model);
        }

        private string GetRedirectUrl(string returnUrl)
        {
            if (string.IsNullOrEmpty(returnUrl) || !Url.IsLocalUrl(returnUrl))

```

```

        {
            return "/";
        }

        return returnUrl;
    }

    public ActionResult Logout()
    {
        var ctx = Request.GetOwinContext();
        var authManager = ctx.Authentication;

        authManager.SignOut("ApplicationCookie");
        return RedirectToAction("Login", "Account");
    }
}
}

```

ch.muster.se.inv.web.Controllers.ArtikelController

```

using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Net;
using System.Web;
using System.Web.Mvc;
using ch.muster.se.inv.dal.Models;
using ch.muster.se.inv.bll.Interfaces;
using System.IO;
using System.Drawing;
using PagedList;
using PagedList.Mvc;
using OfficeOpenXml;
using iTextSharp.text;
using iTextSharp.text.pdf;
using iTextSharp.tool.xml;
using iTextSharp.tool.xml.html;
using iTextSharp.tool.xml.parser;
using iTextSharp.tool.xml.pipeline.css;
using iTextSharp.tool.xml.pipeline.end;
using iTextSharp.tool.xml.pipeline.html;

namespace ch.muster.se.inv.web.Controllers
{
    public class ArtikelController : Controller
    {
        IArtikelService _artikelService;
        IKategorieService _kategorieService;
        IBenutzerService _benutzerService;
        IRaumService _raumService;
        IFiBuKontoService _fiBuKontoService;

        public ArtikelController(IArtikelService artikelService, IKategorieService kategorieService, IBenutzerService
benutzerService, IRaumService raumService, IFiBuKontoService fiBuKontoService)
        {
            _artikelService = artikelService;
            _kategorieService = kategorieService;
            _benutzerService = benutzerService;
            _raumService = raumService;
            _fiBuKontoService = fiBuKontoService;
        }

        // Das ist die Übersicht über alle Artikel, die Parameter sind optional und werden für den Suchfilter
        benötigt
        // Die PagedList habe ich nach diesem Tutorial aufgebaut
        // http://www.itprotoday.com/web-development/aspnet-mvc-paging-done-perfectly
        public ActionResult Index(int? KategorieID, int? FiBuKontoID, int? RaumID, int? BenutzerID, int?
ArtikelYearsStart, int? ArtikelYearsEnd, string Search, string sortOrder, int page = 1, int pagesize = 25)
        {
            // Das Searchmodel wird hier abgefüllt
            ArtikelSearchModel artikelSearchModel = new ArtikelSearchModel()
            {
                KategorieID = KategorieID,
                FiBuKontoID = FiBuKontoID,
                RaumID = RaumID,
                BenutzerID = BenutzerID,
                ArtikelYearsStart = ArtikelYearsStart,
                ArtikelYearsEnd = ArtikelYearsEnd,
                Search = Search
            };

            IEnumerable<Artikel> artikellist = _artikelService.GetArtikels(artikelSearchModel);

```

```

        ViewBag.exportLink = "/Artikel/ArtikelExport?KategorieID=" + KategorieID + "&FiBuKontoID=" + FiBuKontoID
+ "&RaumID=" + RaumID + "&BenutzerID=" + BenutzerID + "&ArtikelYearsStart=" + ArtikelYearsStart + "&ArtikelYearsEnd="
+ ArtikelYearsEnd + "&Search=" + Search;

        ViewBag.exportLinkPDF = "/Artikel/Download?KategorieID=" + KategorieID + "&FiBuKontoID=" + FiBuKontoID +
"&RaumID=" + RaumID + "&BenutzerID=" + BenutzerID + "&ArtikelYearsStart=" + ArtikelYearsStart + "&ArtikelYearsEnd=" +
ArtikelYearsEnd + "&Search=" + Search;

        Session["BackLink"] = "/Artikel?KategorieID=" + KategorieID + "&FiBuKontoID=" + FiBuKontoID + "&RaumID="
+ RaumID + "&BenutzerID=" + BenutzerID + "&ArtikelYearsStart=" + ArtikelYearsStart + "&ArtikelYearsEnd=" +
ArtikelYearsEnd + "&Search=" + Search + "&sortOrder=" + sortOrder + "&page=" + page;

        ViewBag.KategorieID = new SelectList(_kategorieService.GetKategorien().OrderBy(x => x.Bezeichnung), "ID",
"Bezeichnung", KategorieID);
        ViewBag.FiBuKontoID = new SelectList(_fiBuKontoService.GetFiBuKontos().OrderBy(x => x.Nummer), "ID",
"Nummer", FiBuKontoID);
        ViewBag.RaumID = new SelectList(_raumService.GetRaums().OrderBy(x => x.Bezeichnung), "ID", "Bezeichnung",
RaumID);
        ViewBag.BenutzerID = new SelectList(_benutzerService.GetBenutzers().OrderBy(x => x.Fullname), "ID",
"Fullname", BenutzerID);
        ViewBag.ArtikelYearsStart = ArtikelYears();
        ViewBag.ArtikelYearsEnd = ArtikelYears();

        //Sortieren der Tabelle nach dem Beispiel im angegebenen Tutorial
        //https://www.itworld.com/article/2956575/development/how-to-sort-search-and-paginate-tables-in-asp-net-
mvc-5.html

        ViewBag.InventarNummerSortParam = sortOrder == "InventarNummer" ? "InventarNummer_desc" :
"InventarNummer";
        ViewBag.GegenstandSortParam = sortOrder == "Gegenstand" ? "Gegenstand_desc" : "Gegenstand";
        ViewBag.KategorieSortParam = sortOrder == "Kategorie" ? "Kategorie_desc" : "Kategorie";
        ViewBag.OrtSortParam = sortOrder == "Ort" ? "Ort_desc" : "Ort";
        ViewBag.NutzerSortParam = sortOrder == "Nutzer" ? "Nutzer_desc" : "Nutzer";
        ViewBag.FiBuKontoSortParam = sortOrder == "FiBuKonto" ? "FiBuKonto_desc" : "FiBuKonto";
        ViewBag.BeschaffungsdatumSortParam = sortOrder == "Beschaffungsdatum" ? "Beschaffungsdatum_desc" :
"Beschaffungsdatum";
        ViewBag.BeschaffungswertSortParam = sortOrder == "Beschaffungswert" ? "Beschaffungswert_desc" :
"Beschaffungswert";

        ViewBag.CurrentSort = sortOrder;

        switch (sortOrder)
        {
            case "InventarNummer":
                artikelList = artikelList.OrderBy(x => x.InventarNummer);
                break;
            case "InventarNummer_desc":
                artikelList = artikelList.OrderByDescending(x => x.InventarNummer);
                break;
            case "Gegenstand":
                artikelList = artikelList.OrderBy(x => x.Gegenstand);
                break;
            case "Gegenstand_desc":
                artikelList = artikelList.OrderByDescending(x => x.Gegenstand);
                break;
            case "Kategorie":
                artikelList = artikelList.OrderBy(x => x.Kategorie.Bezeichnung);
                break;
            case "Kategorie_desc":
                artikelList = artikelList.OrderByDescending(x => x.Kategorie.Bezeichnung);
                break;
            case "Ort":
                artikelList = artikelList.OrderBy(x => x.Ort);
                break;
            case "Ort_desc":
                artikelList = artikelList.OrderByDescending(x => x.Ort);
                break;
            case "Nutzer":
                artikelList = artikelList.OrderBy(x => x.Nutzer.Fullname);
                break;
            case "Nutzer_desc":
                artikelList = artikelList.OrderByDescending(x => x.Nutzer.Fullname);
                break;
            case "FiBuKonto":
                artikelList = artikelList.OrderBy(x => x.Kategorie.FiBuKonto.Nummer);
                break;
            case "FiBuKonto_desc":
                artikelList = artikelList.OrderByDescending(x => x.Kategorie.FiBuKonto.Nummer);
                break;
            case "Beschaffungsdatum":
                artikelList = artikelList.OrderBy(x => x.Beschaffungsdatum);
                break;
            case "Beschaffungsdatum_desc":
                artikelList = artikelList.OrderByDescending(x => x.Beschaffungsdatum);
                break;
            case "Beschaffungswert":

```

```

        artikelList = artikelList.OrderBy(x => x.Beschaffungswert);
        break;
    case "Beschaffungswert_desc":
        artikelList = artikelList.OrderByDescending(x => x.Beschaffungswert);
        break;
    default:
        break;
}

ViewBag.ExportButtons = artikelList.Count() == 0 ? "disabled" : ""; // Wenn es keine Einträge in der
artikelList hat, sind die Buttons deaktiviert

return View(new PagedList<Artikel>(artikelList.ToList(), page, pagesize));
}

// Diese Methode wird benötigt zum erstellen der Excel Liste
public ActionResult ArtikelExport(int? KategorieID, int? FiBuKontoID, int? RaumID, int? BenutzerID, int?
ArtikelYearsStart, int? ArtikelYearsEnd, string Search)
{
    // Das Searchmodel wird hier abgefüllt
    ArtikelSearchModel artikelSearchModel = new ArtikelSearchModel()
    {
        KategorieID = KategorieID,
        FiBuKontoID = FiBuKontoID,
        RaumID = RaumID,
        BenutzerID = BenutzerID,
        ArtikelYearsStart = ArtikelYearsStart,
        ArtikelYearsEnd = ArtikelYearsEnd,
        Search = Search
    };

    IEnumerable<Artikel> artikelList = _artikelService.GetArtikels(artikelSearchModel);

    MemoryStream ms = ArtikelToExcel(artikelList.OrderBy(d => d.InventarNumber).ToList());
    ms.WriteTo(Response.OutputStream);
    Response.ContentType = "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet";
    Response.ContentEncoding = new System.Text.UTF8Encoding(); // UTF-8
    Response.AddHeader("Content-Disposition", "attachment;filename=InventarListe_" +
GetTimestamp(DateTime.Now) + ".xlsx");
    Response.StatusCode = 200;
    Response.End();

    return null;
}

// Diese Methode liefert das Excel File
internal MemoryStream ArtikelToExcel(List<Artikel> ArtikelList)
{
    MemoryStream Result = new MemoryStream();
    ExcelPackage pack = new ExcelPackage();
    ExcelWorksheet ws = pack.Workbook.Worksheets.Add("Inventar");

    int row = 1;

    ws.Cells[row, 1].Value = "Inventar Nummer";
    ws.Cells[row, 2].Value = "Bezeichnung";
    ws.Cells[row, 3].Value = "Beschreibung";
    ws.Cells[row, 4].Value = "Kategorie";
    ws.Cells[row, 5].Value = "Stockwerk";
    ws.Cells[row, 6].Value = "Raum";
    ws.Cells[row, 7].Value = "Nutzer";
    ws.Cells[row, 8].Value = "FiBo Konto (Nummer)";
    ws.Cells[row, 9].Value = "Beschaffungsdatum";
    ws.Cells[row, 10].Value = "Beschaffungswert (in CHF)";

    ws.Row(row).Style.Font.Size = 9;
    ws.Row(row).Style.Font.Bold = true;
    ws.Row(row).Style.Font.Color.SetColor(Color.FromArgb(255, 255, 255));
    ws.Row(row).Height = 18;
    ws.Row(row).Style.VerticalAlignment = OfficeOpenXml.Style.ExcelVerticalAlignment.Center;
    ws.Row(row).Style.Fill.PatternType = OfficeOpenXml.Style.ExcelFillStyle.Solid;
    ws.Row(row).Style.Fill.BackgroundColor.SetColor(Color.FromArgb(51, 101, 155));

    row++;

    foreach (Artikel artikel in ArtikelList)
    {
        ws.Cells[row, 1].Value = artikel.InventarNumber;
        ws.Cells[row, 2].Value = artikel.Bezeichnung;
        ws.Cells[row, 3].Value = artikel.Beschreibung;
        ws.Cells[row, 4].Value = artikel.Kategorie.Bezeichnung;
        ws.Cells[row, 5].Value = artikel.Raum.Lokalisierung;
        ws.Cells[row, 6].Value = artikel.Raum.Bezeichnung;
        ws.Cells[row, 7].Value = artikel.Nutzer.Fullname;
        ws.Cells[row, 8].Value = artikel.Kategorie.FiBuKonto.Nummer.ToString() + " ";
    }
}

```

```

        ws.Cells[row, 9].Value = artikel.Beschaffungsdatum;
        ws.Cells[row, 9].Style.NumberFormat.Format = @"d/m/yyyy";
        ws.Cells[row, 10].Value = artikel.Beschaffungswert;
        ws.Cells[row, 10].Style.NumberFormat.Format = "#,##0.00;(#,##0.00)";

        ws.Row(row).Style.Font.Size = 9;
        ws.Row(row).Height = 18;
        ws.Row(row).Style.VerticalAlignment = OfficeOpenXml.Style.ExcelVerticalAlignment.Center;

        row++;
    }

    ws.Cells.AutoFitColumns(1, 80);

    pack.SaveAs(Result);
    return Result;
}

// Methode für den PDF Download eines Artikels
public ActionResult Download(int? KategorieID, int? FiBuKontoID, int? RaumID, int? BenutzerID, int?
ArtikelYearsStart, int? ArtikelYearsEnd, string Search)
{
    // Das Searchmodel wird hier abgefüllt
    ArtikelSearchModel artikelSearchModel = new ArtikelSearchModel()
    {
        KategorieID = KategorieID,
        FiBuKontoID = FiBuKontoID,
        RaumID = RaumID,
        BenutzerID = BenutzerID,
        ArtikelYearsStart = ArtikelYearsStart,
        ArtikelYearsEnd = ArtikelYearsEnd,
        Search = Search
    };

    IEnumerable<Artikel> artikellist = _artikelService.GetArtikels(artikelSearchModel);

    Response.ContentEncoding = new System.Text.UTF8Encoding(); // UTF-8

    Response.ContentType = "application/pdf";

    FileContentResult fcr = new FileContentResult(RenderViewToPDF("Artikel", "exportpdf",
artikellist.ToArray()), "application/pdf");
    fcr.FileName = "LabelsInventar_" + GetTimestamp(DateTime.Now) + ".pdf";
    return fcr;
}

private byte[] RenderViewToPDF(string ControllerName, string ViewName, IEnumerable<Artikel> artikellist)
{
    var test = ViewName;

    return RenderHtmlToPDF(RenderViewToHTML(ControllerName, ViewName, artikellist));
}

protected string RenderViewToHTML(string ControllerName, string ViewName, object Model)
{
    string Html = string.Empty;
    ViewData.Model = Model;

    if (!string.IsNullOrEmpty(ControllerName))
    {
        ControllerContext.RouteData.Values["controller"] = ControllerName;
    }

    using (var sw = new StringWriter())
    {
        ViewEngineResult vr = ViewEngines.Engines.FindPartialView(ControllerContext, ViewName);
        vr.View.Render(new ViewContext(ControllerContext, vr.View, ViewData, TempData, sw), sw);
        vr.ViewEngine.ReleaseView(ControllerContext, vr.View);
        Html += sw.GetStringBuilder().ToString();
    }

    return Html;
}

protected byte[] RenderHtmlToPDF(string Html)
{
    // http://stackoverflow.com/questions/36180131/using-itextsharp-xmlworker-to-convert-html-to-pdf-and-
write-text-vertically
    // http://stackoverflow.com/questions/20488045/change-default-font-and-fontsize-in-pdf-using-itextsharp

    Document document = new Document(PageSize.A4, 50f, 30f, 40f, 90f);

    if (Html.Contains("class=\"landscape\""))
    {
        document.SetPageSize(iTextSharp.text.PageSize.A4.Rotate());
    }
}

```

```

        MemoryStream stream = new MemoryStream();
        TextReader reader = new StringReader(Html);
        PdfWriter writer = PdfWriter.GetInstance(document, stream);
        document.AddTitle("muster ag");

        XMLWorkerFontProvider fonts = new XMLWorkerFontProvider();
        CssApppliers appliers = new CssApppliersImpl(fonts);

        HtmlPipelineContext context = new HtmlPipelineContext(appliers);
        context.SetAcceptUnknown(true);
        context.SetTagFactory(Tags.GetHtmlTagProcessorFactory());

        PdfWriterPipeline pdfpipeline = new PdfWriterPipeline(document, writer);
        HtmlPipeline htmlpipeline = new HtmlPipeline(context, pdfpipeline);

        var resolver = XMLWorkerHelper.GetInstance().GetDefaultCssResolver(false);
        resolver.AddCssFile(Server.MapPath("~/Content/inv.pdf.css"), true);
        CssResolverPipeline csspipeline = new CssResolverPipeline(resolver, htmlpipeline);

        XMLWorker worker = new XMLWorker(csspipeline, true);
        XMLParser parser = new XMLParser(worker);

        document.Open();
        parser.Parse(reader);
        worker.Close();
        document.Close();

        return stream.ToArray();
    }

    // Diese Methode gibt einen Zeitstempel zurück
    public static String GetTimestamp(DateTime value)
    {
        return value.ToString("yyyy_MM_dd_HH_mm");
    }

    public SelectList ArtikelYears()
    {
        IEnumerable<int> artikelListDates = _artikelService.GetArtikels().OrderBy(x =>
x.Beschaffungsdatum).Select(d => d.Beschaffungsdatum.Year).Distinct().ToList();

        List<String> datelistYear = new List<String>();

        foreach (int artikelDate in artikelListDates)
        {
            datelistYear.Add(artikelDate.ToString());
        }

        var min = artikelListDates.Min();
        var max = artikelListDates.Max();
        var difference = max - min;

        for (int i = 1; i < difference; i++)
        {
            datelistYear.Add((min + i).ToString());
        }

        SelectList artikelYears = new SelectList(datelistYear.Distinct().OrderBy(d => d).ToList());

        return artikelYears;
    }

    // Hier sieht man die Details zu einem jeweiligen Artikel
    public ActionResult Details(int? id)
    {
        if (id == null)
        {
            return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
        }
        Artikel artikel = _artikelService.GetArtikel(id);
        if (artikel == null)
        {
            return HttpNotFound();
        }
        return View(artikel);
    }

    // Diese Methode ist zum Erstellen eines Artikels nötig
    public ActionResult Create()
    {
        ViewBag.KategorieID = new SelectList(_kategorieService.GetKategories().OrderBy(x => x.Bezeichnung), "ID",
"Bezeichnung");
        ViewBag.BenutzerIDNutzer = new SelectList(_benutzerService.GetBenutzers().OrderBy(x => x.Fullname), "ID",
"Fullname");
    }

```



```

        ViewBag.RaumID = new SelectList(_raumService.GetRaums().OrderBy(x => x.Bezeichnung), "ID",
"Bezeichnung");
        return View();
    }

    // Diese Methode ist für den Post zuständig, die zum Erstellen eines Artikels gebraucht wird
    [HttpPost]
    [ValidateAntiForgeryToken]
    public ActionResult Create([Bind(Include =
"ID,KategorieID,RaumID,BenutzerIDNutzer,InventarNummer,Bezeichnung,Beschreibung,Beschaffungswert,Beschaffungsdatum")]
Artikel artikel)
    {
        if (ModelState.IsValid)
        {
            _artikelService.CreateArtikel(artikel);
            return RedirectToAction("Index");
        }

        ViewBag.KategorieID = new SelectList(_kategorieService.GetKategorien().OrderBy(x => x.Bezeichnung), "ID",
"Bezeichnung", artikel.KategorieID);
        ViewBag.BenutzerIDNutzer = new SelectList(_benutzerService.GetBenutzers().OrderBy(x => x.Fullname), "ID",
"Fullname", artikel.BenutzerIDNutzer);
        ViewBag.RaumID = new SelectList(_raumService.GetRaums().OrderBy(x => x.Bezeichnung), "ID", "Bezeichnung",
artikel.RaumID);
        return View(artikel);
    }

    // Diese Methode wird zum Editieren eines Artikels benötigt
    public ActionResult Edit(int? id)
    {
        if (id == null)
        {
            return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
        }
        Artikel artikel = _artikelService.GetArtikel(id);
        if (artikel == null)
        {
            return HttpNotFound();
        }
        ViewBag.KategorieID = new SelectList(_kategorieService.GetKategorien().OrderBy(x => x.Bezeichnung), "ID",
"Bezeichnung", artikel.KategorieID);
        ViewBag.BenutzerIDNutzer = new SelectList(_benutzerService.GetBenutzers().OrderBy(x => x.Fullname), "ID",
"Fullname", artikel.BenutzerIDNutzer);
        ViewBag.RaumID = new SelectList(_raumService.GetRaums().OrderBy(x => x.Bezeichnung), "ID", "Bezeichnung",
artikel.RaumID);
        return View(artikel);
    }

    // Diese Methode ist für den Post zuständig, die zum Editieren eines Artikels gebraucht wird
    [HttpPost]
    [ValidateAntiForgeryToken]
    public ActionResult Edit([Bind(Include =
"ID,KategorieID,RaumID,BenutzerIDNutzer,InventarNummer,Bezeichnung,Beschreibung,Beschaffungswert,Beschaffungsdatum")]
Artikel artikel)
    {
        if (ModelState.IsValid)
        {
            _artikelService.EditArtikel(artikel);
            return RedirectToAction("Index");
        }
        ViewBag.KategorieID = new SelectList(_kategorieService.GetKategorien().OrderBy(x => x.Bezeichnung), "ID",
"Bezeichnung", artikel.KategorieID);
        ViewBag.BenutzerIDNutzer = new SelectList(_benutzerService.GetBenutzers().OrderBy(x => x.Fullname), "ID",
"Fullname", artikel.BenutzerIDNutzer);
        ViewBag.RaumID = new SelectList(_raumService.GetRaums().OrderBy(x => x.Bezeichnung), "ID", "Bezeichnung",
artikel.RaumID);
        return View(artikel);
    }

    // Diese Methode wird zum Löschen eines Artikels benötigt
    public ActionResult Delete(int? id)
    {
        if (id == null)
        {
            return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
        }
        Artikel artikel = _artikelService.GetArtikel(id);
        if (artikel == null)
        {
            return HttpNotFound();
        }
        return View(artikel);
    }

    // Diese Methode ist für den Post zuständig, der zum Löschen eines Artikels gebraucht wird
    [HttpPost, ActionName("Delete")]

```

```

        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            _artikelService.DeleteArtikel(id);
            return RedirectToAction("Index");
        }
    }
}

ch.muster.se.inv.web.Controllers.BenutzerController
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Net;
using System.Web;
using System.Web.Mvc;
using ch.muster.se.inv.dal.Models;
using ch.muster.se.inv.bll.Interfaces;
using PagedList;

namespace ch.muster.se.inv.web.Controllers
{
    [Authorize(Roles = "Administrator")]
    public class BenutzerController : Controller
    {
        IBenutzerService _benutzerService;
        IRolleService _rolleService;

        public BenutzerController(IBenutzerService benutzerService, IRolleService rolleService)
        {
            _benutzerService = benutzerService;
            _rolleService = rolleService;
        }

        // Das ist die Übersicht über alle Benutzer
        public ActionResult Index(string sortOrder, int page = 1, int pagesize = 25)
        {
            var benutzers = _benutzerService.GetBenutzers();

            Session["BackLinkBenutzer"] = "/Benutzer?&sortOrder=" + sortOrder + "&page=" + page;

            //Sortieren der Tabelle nach dem Beispiel im angegebenen Tutorial
            //https://www.itworld.com/article/2956575/development/how-to-sort-search-and-paginate-tables-in-asp-net-
            ViewBag.BenutzernameSortParam = sortOrder == "Benutzername" ? "Benutzername_desc" : "Benutzername";
            ViewBag.VornameSortParam = sortOrder == "Vorname" ? "Vorname_desc" : "Vorname";
            ViewBag.NameSortParam = sortOrder == "Name" ? "Name_desc" : "Name";
            ViewBag.EmailSortParam = sortOrder == "EMail" ? "EMail_desc" : "EMail";
            ViewBag.TelefonInternSortParam = sortOrder == "TelefonIntern" ? "TelefonIntern_desc" : "TelefonIntern";
            ViewBag.RolleSortParam = sortOrder == "Rolle" ? "Rolle_desc" : "Rolle";

            ViewBag.CurrentSort = sortOrder;

            switch (sortOrder)
            {
                case "Benutzername":
                    benutzers = benutzers.OrderBy(x => x.Benutzername);
                    break;
                case "Benutzername_desc":
                    benutzers = benutzers.OrderByDescending(x => x.Benutzername);
                    break;
                case "Vorname":
                    benutzers = benutzers.OrderBy(x => x.Vorname);
                    break;
                case "Vorname_desc":
                    benutzers = benutzers.OrderByDescending(x => x.Vorname);
                    break;
                case "Name":
                    benutzers = benutzers.OrderBy(x => x.Name);
                    break;
                case "Name_desc":
                    benutzers = benutzers.OrderByDescending(x => x.Name);
                    break;
                case "EMail":
                    benutzers = benutzers.OrderBy(x => x.EMail);
                    break;
                case "EMail_desc":
                    benutzers = benutzers.OrderByDescending(x => x.EMail);
                    break;
                case "TelefonIntern":
                    benutzers = benutzers.OrderBy(x => x.TelefonIntern);
                    break;
                case "TelefonIntern_desc":

```

```

        benutzers = benutzers.OrderByDescending(x => x.TelefonIntern);
        break;
    case "Rolle":
        benutzers = benutzers.OrderBy(x => x.Rolle.Name);
        break;
    case "Rolle_desc":
        benutzers = benutzers.OrderByDescending(x => x.Rolle.Name);
        break;
    default:
        break;
    }

    return View(new PagedList<Benutzer>(benutzers.ToList(), page, pagesize));
}

// Hier sieht man die Details zu einem jeweiligen Benutzer
public ActionResult Details(int? id)
{
    if (id == null)
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }
    Benutzer benutzer = _benutzerService.GetBenutzer(id);
    if (benutzer == null)
    {
        return HttpNotFound();
    }
    return View(benutzer);
}

// Diese Methode ist zum Erstellen eines Benutzers nötig
public ActionResult Create()
{
    ViewBag.RolleID = new SelectList(_rolleService.GetRollen(), "ID", "Name");
    return View();
}

// Diese Methode ist für den Post zuständig, die zum Erstellen eines Benutzers gebraucht wird
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Create([Bind(Include = "ID,RolleID,Benutzername,Vorname,Name,EMail,TelefonIntern")]
Benutzer benutzer)
{
    if (ModelState.IsValid)
    {
        _benutzerService.CreateBenutzer(benutzer);
        return RedirectToAction("Index");
    }

    ViewBag.RolleID = new SelectList(_rolleService.GetRollen(), "ID", "Name", benutzer.RolleID);
    return View(benutzer);
}

// Diese Methode wird zum Editieren eines Benutzers benötigt
public ActionResult Edit(int? id)
{
    if (id == null)
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }
    Benutzer benutzer = _benutzerService.GetBenutzer(id);
    if (benutzer == null)
    {
        return HttpNotFound();
    }
    ViewBag.RolleID = new SelectList(_rolleService.GetRollen(), "ID", "Name", benutzer.RolleID);
    return View(benutzer);
}

// Diese Methode ist für den Post zuständig, die zum Editieren eines Benutzers gebraucht wird
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Edit([Bind(Include = "ID,RolleID,Benutzername,Vorname,Name,EMail,TelefonIntern")]
Benutzer benutzer)
{
    if (ModelState.IsValid)
    {
        _benutzerService.EditBenutzer(benutzer);
        return RedirectToAction("Index");
    }
    ViewBag.RolleID = new SelectList(_rolleService.GetRollen(), "ID", "Name", benutzer.RolleID);
    return View(benutzer);
}

// Diese Methode wird zum Löschen eines Benutzers benötigt
public ActionResult Delete(int? id)

```

```

    {
        if (id == null)
        {
            return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
        }
        Benutzer benutzer = _benutzerService.GetBenutzer(id);
        if (benutzer == null)
        {
            return HttpNotFound();
        }
        return View(benutzer);
    }

    // Diese Methode ist für den Post zuständig, der zum Löschen eines Benutzers gebraucht wird
    [HttpPost, ActionName("Delete")]
    [ValidateAntiForgeryToken]
    public ActionResult DeleteConfirmed(int id)
    {
        _benutzerService.DeleteBenutzer(id);
        return RedirectToAction("Index");
    }
}
}

ch.muster.se.inv.web.Controllers.FiBuKontoController
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Net;
using System.Web;
using System.Web.Mvc;
using ch.muster.se.inv.dal.Models;
using ch.muster.se.inv.bll.Interfaces;
using PagedList;

namespace ch.muster.se.inv.web.Controllers
{
    public class FiBuKontoController : Controller
    {
        IFiBuKontoService _fiBuKontoService;

        public FiBuKontoController(IFiBuKontoService fiBuKontoService)
        {
            _fiBuKontoService = fiBuKontoService;
        }

        // Das ist die Übersicht über alle FiBuKonten
        public ActionResult Index(string sortOrder, int page = 1, int pagesize = 25)
        {
            var fibukontos = _fiBuKontoService.GetFiBuKontos();

            Session["BackLinkFiBuKonto"] = "/FiBuKonto?&sortOrder=" + sortOrder + "&page=" + page;

            //Sortieren der Tabelle nach dem Beispiel im angegebenen Tutorial
            //https://www.itworld.com/article/2956575/development/how-to-sort-search-and-paginate-tables-in-asp-net-
            ViewBag.NummerSortParam = sortOrder == "Nummer" ? "Nummer_desc" : "Nummer";
            ViewBag.BezeichnungSortParam = sortOrder == "Bezeichnung" ? "Bezeichnung_desc" : "Bezeichnung";

            ViewBag.CurrentSort = sortOrder;

            switch (sortOrder)
            {
                case "Nummer":
                    fibukontos = fibukontos.OrderBy(x => x.Nummer);
                    break;
                case "Nummer_desc":
                    fibukontos = fibukontos.OrderByDescending(x => x.Nummer);
                    break;
                case "Bezeichnung":
                    fibukontos = fibukontos.OrderBy(x => x.Bezeichnung);
                    break;
                case "Bezeichnung_desc":
                    fibukontos = fibukontos.OrderByDescending(x => x.Bezeichnung);
                    break;
                default:
                    break;
            }

            return View(new PagedList<FiBuKonto>(fibukontos.ToList(), page, pagesize));
        }

        // Hier sieht man die Details zu einem jeweiligen FiBuKonto

```

```

public ActionResult Details(int? id)
{
    if (id == null)
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }
    FiBuKonto fiBuKonto = _fiBuKontoService.GetFiBuKonto(id);
    if (fiBuKonto == null)
    {
        return HttpNotFound();
    }
    return View(fiBuKonto);
}

// Diese Methode ist zum Erstellen eines FiBuKontos nötig
public ActionResult Create()
{
    return View();
}

// Diese Methode ist für den Post zuständig, die zum Erstellen eines FiBuKontos gebraucht wird
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Create([Bind(Include = "ID,Nummer,Bezeichnung")] FiBuKonto fiBuKonto)
{
    if (ModelState.IsValid)
    {
        _fiBuKontoService.CreateFiBuKonto(fiBuKonto);
        return RedirectToAction("Index");
    }

    return View(fiBuKonto);
}

// Diese Methode wird zum Editieren eines FiBuKontos benötigt
public ActionResult Edit(int? id)
{
    if (id == null)
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }
    FiBuKonto fiBuKonto = _fiBuKontoService.GetFiBuKonto(id);
    if (fiBuKonto == null)
    {
        return HttpNotFound();
    }
    return View(fiBuKonto);
}

// Diese Methode ist für den Post zuständig, die zum Editieren eines FiBuKontos gebraucht wird
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Edit([Bind(Include = "ID,Nummer,Bezeichnung")] FiBuKonto fiBuKonto)
{
    if (ModelState.IsValid)
    {
        _fiBuKontoService.EditFiBuKonto(fiBuKonto);
        return RedirectToAction("Index");
    }
    return View(fiBuKonto);
}

// Diese Methode wird zum Löschen eines FiBuKontos benötigt
public ActionResult Delete(int? id)
{
    if (id == null)
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }
    FiBuKonto fiBuKonto = _fiBuKontoService.GetFiBuKonto(id);
    if (fiBuKonto == null)
    {
        return HttpNotFound();
    }
    return View(fiBuKonto);
}

// Diese Methode ist für den Post zuständig, der zum Löschen eines FiBuKontos gebraucht wird
[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
public ActionResult DeleteConfirmed(int id)
{
    _fiBuKontoService.DeleteFiBuKonto(id);
    return RedirectToAction("Index");
}

```

```

    }
}

ch.muster.se.inv.web.Controllers.KategorieController
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Net;
using System.Web;
using System.Web.Mvc;
using ch.muster.se.inv.dal.Models;
using ch.muster.se.inv.bll.Interfaces;
using PagedList;

namespace ch.muster.se.inv.web.Controllers
{
    public class KategorieController : Controller
    {
        IKategorieService _kategorieService;
        IFiBuKontoService _fiBuKontoService;

        public KategorieController(IKategorieService kategorieService, IFiBuKontoService fiBuKontoService)
        {
            _kategorieService = kategorieService;
            _fiBuKontoService = fiBuKontoService;
        }

        // Das ist die Übersicht über alle Kategorien
        public ActionResult Index(string sortOrder, int page = 1, int pageSize = 25)
        {
            var kategorien = _kategorieService.GetKategorien();

            Session["BackLinkKategorie"] = "/Kategorie?&sortOrder=" + sortOrder + "&page=" + page;

            //Sortieren der Tabelle nach dem Beispiel im angegebenen Tutorial
            //https://www.itworld.com/article/2956575/development/how-to-sort-search-and-paginate-tables-in-asp-net-
            ViewBag.BezeichnungSortParam = sortOrder == "Bezeichnung" ? "Bezeichnung_desc" : "Bezeichnung";
            ViewBag.FiBuKontoSortParam = sortOrder == "FiBuKonto" ? "FiBuKonto_desc" : "FiBuKonto";

            ViewBag.CurrentSort = sortOrder;

            switch (sortOrder)
            {
                case "Bezeichnung":
                    kategorien = kategorien.OrderBy(x => x.Bezeichnung);
                    break;
                case "Bezeichnung_desc":
                    kategorien = kategorien.OrderByDescending(x => x.Bezeichnung);
                    break;
                case "FiBuKonto":
                    kategorien = kategorien.OrderBy(x => x.FiBuKonto.Nummer);
                    break;
                case "FiBuKonto_desc":
                    kategorien = kategorien.OrderByDescending(x => x.FiBuKonto.Nummer);
                    break;
                default:
                    break;
            }

            return View(new PagedList<Kategorie>(kategorien.ToList(), page, pageSize));
        }

        // Hier sieht man die Details zu einer jeweiligen Kategorie
        public ActionResult Details(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Kategorie kategorie = _kategorieService.GetKategorie(id);
            if (kategorie == null)
            {
                return HttpNotFound();
            }
            return View(kategorie);
        }

        // Diese Methode ist zum Erstellen einer Kategorie nötig
        public ActionResult Create()
        {
            ViewBag.FiBuKontoID = new SelectList(_fiBuKontoService.GetFiBuKontos(), "ID", "Nummer");
            return View();
        }
    }
}

```

```

    }

    // Diese Methode ist für den Post zuständig, die zum Erstellen einer Kategorie gebraucht wird
    [HttpPost]
    [ValidateAntiForgeryToken]
    public ActionResult Create([Bind(Include = "ID,FiBuKontoID,Bezeichnung")] Kategorie kategorie)
    {
        if (ModelState.IsValid)
        {
            _kategorieService.CreateKategorie(kategorie);
            return RedirectToAction("Index");
        }

        ViewBag.FiBuKontoID = new SelectList(_fiBuKontoService.GetFiBuKontos(), "ID", "Nummer",
kategorie.FiBuKontoID);
        return View(kategorie);
    }

    // Diese Methode wird zum Editieren einer Kategorie benötigt
    public ActionResult Edit(int? id)
    {
        if (id == null)
        {
            return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
        }
        Kategorie kategorie = _kategorieService.GetKategorie(id);
        if (kategorie == null)
        {
            return HttpNotFound();
        }
        ViewBag.FiBuKontoID = new SelectList(_fiBuKontoService.GetFiBuKontos(), "ID", "Nummer",
kategorie.FiBuKontoID);
        return View(kategorie);
    }

    // Diese Methode ist für den Post zuständig, die zum Editieren einer Kategorie gebraucht wird
    [HttpPost]
    [ValidateAntiForgeryToken]
    public ActionResult Edit([Bind(Include = "ID,FiBuKontoID,Bezeichnung")] Kategorie kategorie)
    {
        if (ModelState.IsValid)
        {
            _kategorieService.EditKategorie(kategorie);
            return RedirectToAction("Index");
        }
        ViewBag.FiBuKontoID = new SelectList(_fiBuKontoService.GetFiBuKontos(), "ID", "Nummer",
kategorie.FiBuKontoID);
        return View(kategorie);
    }

    // Diese Methode wird zum Löschen einer Kategorie benötigt
    public ActionResult Delete(int? id)
    {
        if (id == null)
        {
            return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
        }
        Kategorie kategorie = _kategorieService.GetKategorie(id);
        if (kategorie == null)
        {
            return HttpNotFound();
        }
        return View(kategorie);
    }

    // Diese Methode ist für den Post zuständig, der zum Löschen einer Kategorie gebraucht wird
    [HttpPost, ActionName("Delete")]
    [ValidateAntiForgeryToken]
    public ActionResult DeleteConfirmed(int id)
    {
        _kategorieService.DeleteKategorie(id);
        return RedirectToAction("Index");
    }
}

```

ch.muster.se.inv.web.Controllers.RaumController

```

using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Net;
using System.Web;
using System.Web.Mvc;

```

```

using ch.muster.se.inv.dal.Models;
using ch.muster.se.inv.bll.Interfaces;
using PagedList;

namespace ch.muster.se.inv.web.Controllers
{
    public class RaumController : Controller
    {
        IRaumService _raumService;

        public RaumController(IRaumService raumService)
        {
            _raumService = raumService;
        }

        // Das ist die Übersicht über alle Räume
        public ActionResult Index(string sortOrder, int page = 1, int pagesize = 25)
        {
            var raums = _raumService.GetRaums();

            Session["BackLinkRaum"] = "/Raum?sortOrder=" + sortOrder + "&page=" + page;

            //Sortieren der Tabelle nach dem Beispiel im angegebenen Tutorial
            //https://www.itworld.com/article/2956575/development/how-to-sort-search-and-paginate-tables-in-asp-net-
            ViewBag.LokalisierungSortParam = sortOrder == "Lokalisierung" ? "Lokalisierung_desc" : "Lokalisierung";
            ViewBag.BezeichnungSortParam = sortOrder == "Bezeichnung" ? "Bezeichnung_desc" : "Bezeichnung";

            ViewBag.CurrentSort = sortOrder;

            switch (sortOrder)
            {
                case "Lokalisierung":
                    raums = raums.OrderBy(x => x.Lokalisierung);
                    break;
                case "Lokalisierung_desc":
                    raums = raums.OrderByDescending(x => x.Lokalisierung);
                    break;
                case "Bezeichnung":
                    raums = raums.OrderBy(x => x.Bezeichnung);
                    break;
                case "Bezeichnung_desc":
                    raums = raums.OrderByDescending(x => x.Bezeichnung);
                    break;
                default:
                    break;
            }

            return View(new PagedList<Raum>(raums.ToList(), page, pagesize));
        }

        // Hier sieht man die Details zu einem jeweiligen Raum
        public ActionResult Details(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Raum raum = _raumService.GetRaum(id);
            if (raum == null)
            {
                return HttpNotFound();
            }
            return View(raum);
        }

        // Diese Methode ist zum Erstellen eines Raumes nötig
        public ActionResult Create()
        {
            IEnumerable<SelectListItem> lokalisierungItems = new SelectList(
                new List<SelectListItem>
                {
                    new SelectListItem { Text = "UG", Value = "UG"},
                    new SelectListItem { Text = "EG", Value = "EG"},
                    new SelectListItem { Text = "OG", Value = "OG"},
                }, "Value", "Text");

            ViewData["lokalisierungItems"] = lokalisierungItems;

            return View();
        }

        // Diese Methode ist für den Post zuständig, die zum Erstellen eines Raumes gebraucht wird
        [HttpPost]
        [ValidateAntiForgeryToken]

```



```

public ActionResult Create([Bind(Include = "ID,Lokalisierung,Bezeichnung")] Raum raum)
{
    if (ModelState.IsValid)
    {
        if (raum.Lokalisierung == null)
        {
            ModelState.AddModelError(string.Empty, "Die Lokalisierung muss angegeben werden."); // Der
            Benutzer wird darauf hingewiesen, das die Lokalisierung angegeben werden muss.

            IEnumerable<SelectListItem> lokalisierungItems = new SelectList(
                new List<SelectListItem>
                {
                    new SelectListItem { Text = "UG", Value = "UG"},
                    new SelectListItem { Text = "EG", Value = "EG"},
                    new SelectListItem { Text = "OG", Value = "OG"},
                }, "Value", "Text");

            ViewData["lokalisierungItems"] = lokalisierungItems;

            return View(raum);
        }
        _raumService.CreateRaum(raum);
        return RedirectToAction("Index");
    }

    return View(raum);
}

// Diese Methode wird zum Editieren eines Raumes benötigt
public ActionResult Edit(int? id)
{
    if (id == null)
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }
    Raum raum = _raumService.GetRaum(id);
    if (raum == null)
    {
        return HttpNotFound();
    }

    IEnumerable<SelectListItem> lokalisierungItems = new SelectList(
        new List<SelectListItem>
        {
            new SelectListItem { Text = "UG", Value = "UG", Selected = raum.Lokalisierung.Equals("ug",
                StringComparison.InvariantCultureIgnoreCase) ? true : false },
            new SelectListItem { Text = "EG", Value = "EG", Selected = raum.Lokalisierung.Equals("eg",
                StringComparison.InvariantCultureIgnoreCase) ? true : false },
            new SelectListItem { Text = "OG", Value = "OG", Selected = raum.Lokalisierung.Equals("og",
                StringComparison.InvariantCultureIgnoreCase) ? true : false },
        }, "Value", "Text");

    ViewData["lokalisierungItems"] = lokalisierungItems;

    return View(raum);
}

// Diese Methode ist für den Post zuständig, die zum Editieren eines Raumes gebraucht wird
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Edit([Bind(Include = "ID,Lokalisierung,Bezeichnung")] Raum raum)
{
    if (ModelState.IsValid)
    {
        _raumService.EditRaum(raum);
        return RedirectToAction("Index");
    }

    return View(raum);
}

// Diese Methode wird zum Löschen eines Raumes benötigt
public ActionResult Delete(int? id)
{
    if (id == null)
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }
    Raum raum = _raumService.GetRaum(id);
    if (raum == null)
    {
        return HttpNotFound();
    }

    return View(raum);
}

```

```

        // Diese Methode ist für den Post zuständig, der zum Löschen eines Raumes gebraucht wird
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            _raumService.DeleteRaum(id);
            return RedirectToAction("Index");
        }
    }
}

ch.muster.se.inv.web.Localization
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Globalization;
using System.Linq;
using System.Threading;
using System.Web;
using System.Web.Mvc;
using System.Web.Routing;

// Weitgehend übernommen aus dem Kundenprojekt com.jmg.cc

namespace ch.muster.se.inv.web.Localization
{
    public static class LocalizedExtensionMethods
    {
        public static IHtmlString Resource(this HtmlHelper helper, string key)
        {
            string result = ch.muster.se.inv.web.Resources.Strings.ResourceManager.GetString(key,
                Thread.CurrentThread.CurrentCulture);
            if (!string.IsNullOrEmpty(result))
            {
                return new HtmlString(result);
            }
            return new HtmlString("<span style=\"color:#FF0000;\">" + key + "</span>");
        }
    }

    public class LocalizationFilter : ActionFilterAttribute
    {
        // filters.Add(new LocalizationFilter());
        // [LocalizationFilter]

        public override void OnActionExecuting(ActionExecutingContext ctx)
        {
            if (ctx.RouteData.Values["controller"].ToString().ToLower() != "file" &&
                ctx.RouteData.Values["controller"].ToString().ToLower() != "export" && ctx.RouteData.Values["lang"] == null)
            {
                List<string> SiteLang = new List<string>() { "de-ch", "en-us", "fr-ch", "it-ch" };
                ctx.RouteData.Values["lang"] = SiteLang[0];

                if (HttpContext.Current.Request.UserLanguages != null)
                {
                    foreach (string Lang in HttpContext.Current.Request.UserLanguages)
                    {
                        if (SiteLang.Any(a => a.Substring(0, 2) == Lang.Substring(0, 2)))
                        {
                            ctx.RouteData.Values["lang"] = SiteLang.FirstOrDefault(f => f.Substring(0, 2) ==
                                Lang.Substring(0, 2));
                            break;
                        }
                    }
                }

                var qs = HttpContext.Current.Request.QueryString;
                var test = qs.AllKeys.Aggregate(new RouteValueDictionary(ctx.RouteData.Values), (rvd, k) => {
                    rvd.Add(k, qs[k]); return rvd; });
                ctx.Result = new RedirectToRouteResult(test);
            }
        }
    }

    public class LocalizedControllerActivator : IControllerActivator
    {
        private string _DefaultLanguage = "en";

        public IController Create(RequestContext requestContext, Type controllerType)
        {
            //Get the {language} parameter in the RouteData
            string lang = requestContext.RouteData.Values["lang"] != null ?
                requestContext.RouteData.Values["lang"].ToString() : _DefaultLanguage;

```

```

        if (lang != _DefaultLanguage)
        {
            try
            {
                Thread.CurrentThread.CurrentCulture =
                Thread.CurrentThread.CurrentUICulture = new CultureInfo(lang);
            }
            catch (Exception e)
            {
                throw new NotSupportedException(String.Format("ERROR: Invalid language code '{0}'.", lang));
            }
        }

        return DependencyResolver.Current.GetService(controllerType) as IController;
    }
}

public class MyLocalizationProvider : DataAnnotationsModelMetadataProvider
{
    protected override ModelMetadata CreateMetadata(
        IEnumerable<Attribute> attributes,
        Type containerType,
        Func<object> modelAccessor,
        Type modelType,
        string propertyName)
    {
        string sKey = string.Empty;
        string sLocalizedText = string.Empty;

        HttpContext.Current.Application.Lock();
        foreach (var attr in attributes)
        {
            if (attr != null)
            {
                string typeName = attr.GetType().Name;
                string attrAppKey = string.Empty;

                if (typeName.Equals("DisplayAttribute"))
                {
                    sKey = ((DisplayAttribute)attr).Name;

                    if (!string.IsNullOrEmpty(sKey))
                    {
                        attrAppKey = string.Format("{0}-{1}-{2}",
                            containerType.Name, propertyName, typeName);
                        if (HttpContext.Current.Application[attrAppKey] == null)
                        {
                            HttpContext.Current.Application[attrAppKey] = sKey;
                        }
                        else
                        {
                            sKey = HttpContext.Current.Application[attrAppKey].ToString();
                        }
                    }

                    sLocalizedText = ch.muster.se.inv.web.Resources.Strings.ResourceManager.GetString(sKey,
                        Thread.CurrentThread.CurrentCulture);
                    if (string.IsNullOrEmpty(sLocalizedText))
                    {
                        sLocalizedText = sKey;
                    }
                    //else
                    //{
                    //    sLocalizedText = "### " + sKey + " ###";
                    //}

                    ((DisplayAttribute)attr).Name = sLocalizedText;
                }
            }
            else if (attr is ValidationAttribute)
            {
                sKey = ((ValidationAttribute)attr).ErrorMessage;

                if (!string.IsNullOrEmpty(sKey))
                {
                    attrAppKey = string.Format("{0}-{1}-{2}",
                        containerType.Name, propertyName, typeName);
                    if (HttpContext.Current.Application[attrAppKey] == null)
                    {
                        HttpContext.Current.Application[attrAppKey] = sKey;
                    }
                    else
                    {
                        sKey = HttpContext.Current.Application[attrAppKey].ToString();
                    }
                }
            }
        }
    }
}

```

```

        sLocalizedText = ch.muster.se.inv.web.Resources.Strings.ResourceManager.GetString(sKey,
Thread.CurrentThread.CurrentCulture);
        if (string.IsNullOrEmpty(sLocalizedText))
        {
            sLocalizedText = sKey;
        }

        ((ValidationAttribute)attr).ErrorMessage = sLocalizedText;
    }
}
}
}
}
}
}
}
}
}

ch.muster.se.inv.web.Utilities
using System;
using System.Collections.Generic;
using System.Collections.Specialized;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Routing;

namespace ch.muster.se.inv.web.Utilities
{
    // Dieser Code wurde von dem Tutorial genommen und weitgehend angepasst, siehe Doku
    // http://www.codingeverything.com/2014/05/mvcbootstrapactivenavbar.html
    // Diese HtmlHelper Klasse ist dafür zuständig, dass das Navigationsitem active ist wenn man sich auf der
    // jeweiligen Seite dazu befindet

    public static class Utilities
    {
        public static string IsActive(this HtmlHelper html,
            string control)
        {
            var routeData = html.ViewContext.RouteData;

            var routeControl = (string)routeData.Values["controller"];

            var returnActive = control == routeControl;

            return returnActive ? "active" : "";
        }

        public static IHtmlString SortIdentifier(this HtmlHelper html, string sortOrder, string field)
        {
            if (string.IsNullOrEmpty(sortOrder) || (sortOrder.Trim() != field && sortOrder.Replace("_desc",
            "").Trim() != field)) return null;

            string glyph = "glyphicon glyphicon-chevron-up";
            if (sortOrder.ToLower().Contains("desc"))
            {
                glyph = "glyphicon glyphicon-chevron-down";
            }

            var span = new TagBuilder("span");
            span.Attributes["class"] = glyph;

            return MvcHtmlString.Create(span.ToString());
        }

        public static RouteValueDictionary ToRouteValueDictionary(this NameValueCollection collection, string newKey,
            string newValue)
        {
            var routeValueDictionary = new RouteValueDictionary();
            foreach (var key in collection.AllKeys)
            {
                if (key == null) continue;
                if (routeValueDictionary.ContainsKey(key))
                    routeValueDictionary.Remove(key);

                routeValueDictionary.Add(key, collection[key]);
            }
            if (string.IsNullOrEmpty(newValue))
            {
                routeValueDictionary.Remove(newKey);
            }
            else
            {
                routeValueDictionary.Add(newKey, newValue);
            }
        }
    }
}

```

```

        if (routeValueDictionary.ContainsKey(newKey))
            routeValueDictionary.Remove(newKey);

        routeValueDictionary.Add(newKey, newValue);
    }
    return routeValueDictionary;
}
}
}

ch.muster.se.inv.web.MvcApplication
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Optimization;
using System.Web.Routing;

namespace ch.muster.se.inv.web
{
    public class MvcApplication : System.Web.HttpApplication
    {
        protected void Application_Start()
        {
            AreaRegistration.RegisterAllAreas();
            UnityConfig.RegisterComponents();
            FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
            RouteConfig.RegisterRoutes(RouteTable.Routes);
            BundleConfig.RegisterBundles(BundleTable.Bundles);
        }
    }
}

```

```

ch.muster.se.inv.web.Startup
using Microsoft.Owin;
using Microsoft.Owin.Security.Cookies;
using Owin;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Security.Claims;
using System.Web;
using System.Web.Helpers;

namespace ch.muster.se.inv.web
{
    public class Startup
    {
        public void Configuration(IAppBuilder app)
        {
            app.UseCookieAuthentication(new CookieAuthenticationOptions
            {
                AuthenticationType = "ApplicationCookie",
                LoginPath = new PathString("/Account/Login"),

                // Wenn der User 30 Minuten Inaktiv ist, wird er automatisch ausgeloggt
                ExpireTimeSpan = new TimeSpan(0, 30, 0),
                SlidingExpiration = true
            });

            // Diese Codezeile wurde aus dem File kopiert das in der Dokumentation angegebenen ist
            AntiForgeryConfig.UniqueClaimTypeIdentifier = ClaimTypes.NameIdentifier;
        }
    }
}

```

Views (cshtml)

```

Account.Login
@using ch.muster.se.inv.web.Models
@model Account
@{
    ViewBag.Title = "Login";
}

@* Das Meiste dieser View wurde automatisch generiert *@

<h2>@ViewBag.Title</h2>
<div class="row">
    <div class="col-md-8">
        <section id="loginForm">

```

```

        using (Html.BeginForm("Login", "Account", new { returnUrl = Model.ReturnUrl }, FormMethod.Post, new {
@class = "form-horizontal", role = "form" }))
        {
            @Html.AntiForgeryToken()
            <h4>Einloggen mit Windows Benutzerdaten</h4>
            <hr />
            // Codeabschnitt für diese Error Meldung wurde grösstenteils von hier übernommen
            // https://stackoverflow.com/questions/13867307/show-validationsummary-mvc3-as-alert-error-bootstrap
            if (ViewData.ModelState.Any(x => x.Value.Errors.Any()))
            {
                <div class="alert alert-danger" role="alert">
                    <a class="close" data-dismiss="alert">x</a>
                    @foreach (var modelError in Html.ViewData.ModelState.SelectMany(keyValuePair =>
keyValuePair.Value.Errors))
                    {
                        <p>@modelError.ErrorMessage</p>
                    }
                </div>
            }
            <div class="form-group">
                @Html.LabelFor(m => m.Username, new { @class = "col-md-2 control-label" })
                <div class="col-md-10">
                    @Html.TextBoxFor(m => m.Username, new { @class = "form-control" })
                    @Html.ValidationMessageFor(m => m.Username, "", new { @class = "text-danger" })
                </div>
            </div>
            <div class="form-group">
                @Html.LabelFor(m => m.Password, new { @class = "col-md-2 control-label" })
                <div class="col-md-10">
                    @Html.PasswordFor(m => m.Password, new { @class = "form-control" })
                    @Html.ValidationMessageFor(m => m.Password, "", new { @class = "text-danger" })
                </div>
            </div>
            <div class="form-group">
                <div class="col-md-offset-2 col-md-10">
                    <input type="submit" value="Login" class="btn btn-primary" />
                </div>
            </div>
        }
    </section>
</div>
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}

Artikel.Create
@model ch.muster.se.inv.dal.Models.Artikel

@{
    ViewBag.Title = "Create";
}

<h2>Erstellen</h2>

using (Html.BeginForm(null, null, FormMethod.Post, new { id = "artikelForm" }))
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>Artikel</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        <div class="form-group">
            @Html.LabelFor(model => model.InventarNummer, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.InventarNummer, new { htmlAttributes = new { @class = "form-control" } })
            </div>
            @Html.ValidationMessageFor(model => model.InventarNummer, "", new { @class = "text-danger" })
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Bezeichnung, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Bezeichnung, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Bezeichnung, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Beschreibung, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">

```

```

        @Html.EditorFor(model => model.Beschreibung, new { htmlAttributes = new { @class = "form-control" } })
    })
    @Html.ValidationMessageFor(model => model.Beschreibung, "", new { @class = "text-danger" })
</div>
</div>

<div class="form-group">
    @Html.LabelFor(model => model.KategorieID, htmlAttributes: new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        @Html.DropDownList("KategorieID", null, "", htmlAttributes: new { @class = "form-control" })
        @Html.ValidationMessageFor(model => model.KategorieID, "", new { @class = "text-danger" })
    </div>
</div>

<div class="form-group">
    @Html.LabelFor(model => model.RaumID, htmlAttributes: new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        @Html.DropDownList("RaumID", null, "", htmlAttributes: new { @class = "form-control" })
        @Html.ValidationMessageFor(model => model.RaumID, "", new { @class = "text-danger" })
    </div>
</div>

<div class="form-group">
    @Html.LabelFor(model => model.BenutzerIDNutzer, htmlAttributes: new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        @Html.DropDownList("BenutzerIDNutzer", null, "", htmlAttributes: new { @class = "form-control" })
        @Html.ValidationMessageFor(model => model.BenutzerIDNutzer, "", new { @class = "text-danger" })
    </div>
</div>

<div class="form-group">
    @Html.LabelFor(model => model.Beschaffungsdatum, htmlAttributes: new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        @Html.EditorFor(model => model.Beschaffungsdatum, new { htmlAttributes = new { @class = "form-control" } })
        @Html.ValidationMessageFor(model => model.Beschaffungsdatum, "", new { @class = "text-danger" })
    </div>
</div>

<div class="form-group">
    @Html.LabelFor(model => model.Beschaffungswert, htmlAttributes: new { @class = "control-label col-md-2" })
    <div class="input-group currencyField">
        @Html.EditorFor(model => model.Beschaffungswert, new { htmlAttributes = new { @class = "form-control currency", @id = "currencyInput" } })
        <span class="input-group-addon">CHF</span>
    </div>
    @Html.ValidationMessageFor(model => model.Beschaffungswert, "", new { @class = "currencyDanger text-danger" })
</div>

<div class="form-group">
    <div class="col-md-offset-2 col-md-10">
        <input type="submit" value="Erstellen" class="btn btn-default" />
    </div>
</div>
</div>
}

<div>
    @if (Session["BackLink"] != null)
    {
        <a href="@Session["BackLink"]">Zurück</a>
    }
    else
    {
        @Html.ActionLink("Zurück", "Index")
    }
</div>

@* Die JavaScript Library Autonumeric wurde eingesetzt, damit die Währung richtig angezeigt wird im Input-Feld.
http://autonumeric.org *@

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
    @Scripts.Render("~/bundles/autonumeric")

    <script>
        new AutoNumeric('#currencyInput', {
            decimalCharacter: '.',
            digitGroupSeparator: '\',
        });

        $.validator.methods.range = function (value, element, param) {

```

```

        var globalizedValue = value.replace(".", "");
        return this.optional(element) || (globalizedValue >= param[0] && globalizedValue <= param[1]);
    }

    $.validator.methods.number = function (value, element) {
        return this.optional(element) || /^-?(?:\d+|\d{1,3}(?:[\s\.,]\d{3})+)(?:[\s\.,]\d+)?$/i.test(value);
    }

    $("#artikelForm").submit(function (event) {
        value_with_separators = $('#currencyInput').val();
        clean_value = value_with_separators.replace(/'/g, '');
        console.log(clean_value);
        $('#currencyInput').val(clean_value);
    });
</script>
}

```

Artikel.Delete

```
@model ch.muster.se.inv.dal.Models.Artikel
```

```

@{
    ViewBag.Title = "Delete";
}

```

```

<h2>Löschen</h2>

<h3>Sind Sie sicher, dass Sie den Artikel löschen wollen?</h3>
<div>
    <h4>Artikel</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.InventarNummer)
        </dt>
        <dd>
            @Html.DisplayFor(model => model.InventarNummer)
        </dd>
        <dt>
            @Html.DisplayNameFor(model => model.Bezeichnung)
        </dt>
        <dd>
            @Html.DisplayFor(model => model.Bezeichnung)
        </dd>
        <dt>
            @Html.DisplayNameFor(model => model.Beschreibung)
        </dt>
        <dd>
            @Html.DisplayFor(model => model.Beschreibung)
        </dd>
        <dt>
            @Html.DisplayNameFor(model => model.Kategorie.Bezeichnung)
        </dt>
        <dd>
            @Html.DisplayFor(model => model.Kategorie.Bezeichnung)
        </dd>
        <dt>
            @Html.DisplayNameFor(model => model.Nutzer)
        </dt>
        <dd>
            @Html.DisplayFor(model => model.Nutzer.Fullname)
        </dd>
        <dt>
            @Html.DisplayNameFor(model => model.Raum.Bezeichnung)
        </dt>
        <dd>
            @Html.DisplayFor(model => model.Raum.Bezeichnung)
        </dd>
        <dt>
            @Html.DisplayNameFor(model => model.Raum.Lokalisierung)
        </dt>
        <dd>
            @Html.DisplayFor(model => model.Raum.Lokalisierung)
        </dd>
    </dl>

```



```

        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Kategorie.FiBuKonto.Nummer)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Kategorie.FiBuKonto.Nummer)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Beschaffungsdatum)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Beschaffungsdatum)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Beschaffungswert)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Beschaffungswert)
        </dd>
    </dl>

    @using (Html.BeginForm())
    {
        @Html.AntiForgeryToken()

        <div class="form-actions no-color">
            <input type="submit" value="Löschen" class="btn btn-default" /> |
            @if (Session["BackLink"] != null)
            {
                <a href="@Session["BackLink"]">Zurück</a>
            }
            else
            {
                @Html.ActionLink("Zurück", "Index")
            }
        </div>
    }
</div>

```

Artikel.Details

```
@model ch.muster.se.inv.dal.Models.Artikel
```

```

@{
    ViewBag.Title = "Details";
}

<h2>Details</h2>

<div>
    <h4>Artikel</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.InventarNummer)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.InventarNummer)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Bezeichnung)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Bezeichnung)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Beschreibung)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Beschreibung)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Kategorie.Bezeichnung)
        </dt>
    </dl>

```

```

        <dd>
            @Html.DisplayFor(model => model.Kategorie.Bezeichnung)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Nutzer)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Nutzer.Fullname)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Ort)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Ort)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Kategorie.FiBuKonto.Nummer)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Kategorie.FiBuKonto.Nummer)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Beschaffungsdatum)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Beschaffungsdatum)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Beschaffungswert)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Beschaffungswert)
        </dd>
    </dl>
</div>
<p>
    @Html.ActionLink("Bearbeiten", "Edit", new { id = Model.ID }) |
    @if (Session["BackLink"] != null)
    {
        <a href="@Session["BackLink"]">Zurück</a>
    }
    else
    {
        @Html.ActionLink("Zurück", "Index")
    }
</p>
</div>

Artikel.Edit
@model ch.muster.se.inv.dal.Models.Artikel

@{
    ViewBag.Title = "Edit";
}

<h2>Bearbeiten</h2>

@using (Html.BeginForm(null, null, FormMethod.Post, new { id = "artikelForm" }))
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>Artikel</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        @Html.HiddenFor(model => model.ID)

        <div class="form-group">
            @Html.LabelFor(model => model.InventarNummer, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.InventarNummer, new { htmlAttributes = new { @class = "form-control" } })
            </div>
            @Html.ValidationMessageFor(model => model.InventarNummer, "", new { @class = "text-danger" })
        </div>
    </div>
}

```

```

</div>

<div class="form-group">
    @Html.LabelFor(model => model.Bezeichnung, htmlAttributes: new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        @Html.EditorFor(model => model.Bezeichnung, new { htmlAttributes = new { @class = "form-control" } })
        @Html.ValidationMessageFor(model => model.Bezeichnung, "", new { @class = "text-danger" })
    </div>
</div>

<div class="form-group">
    @Html.LabelFor(model => model.Beschreibung, htmlAttributes: new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        @Html.EditorFor(model => model.Beschreibung, new { htmlAttributes = new { @class = "form-control" } })
        @Html.ValidationMessageFor(model => model.Beschreibung, "", new { @class = "text-danger" })
    </div>
</div>

<div class="form-group">
    @Html.LabelFor(model => model.KategorieID, htmlAttributes: new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        @Html.DropDownList("KategorieID", null, "", htmlAttributes: new { @class = "form-control" })
        @Html.ValidationMessageFor(model => model.KategorieID, "", new { @class = "text-danger" })
    </div>
</div>

<div class="form-group">
    @Html.LabelFor(model => model.RaumID, htmlAttributes: new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        @Html.DropDownList("RaumID", null, "", htmlAttributes: new { @class = "form-control" })
        @Html.ValidationMessageFor(model => model.RaumID, "", new { @class = "text-danger" })
    </div>
</div>

<div class="form-group">
    @Html.LabelFor(model => model.BenutzerIDNutzer, htmlAttributes: new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        @Html.DropDownList("BenutzerIDNutzer", null, "", htmlAttributes: new { @class = "form-control" })
        @Html.ValidationMessageFor(model => model.BenutzerIDNutzer, "", new { @class = "text-danger" })
    </div>
</div>

<div class="form-group">
    @Html.LabelFor(model => model.Beschaffungsdatum, htmlAttributes: new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        @Html.TextBoxFor(model => model.Beschaffungsdatum, new { Value =
Model.Beschaffungsdatum.ToString("yyyy-MM-dd"), Class = "form-control", Type = "Date" })
        @Html.ValidationMessageFor(model => model.Beschaffungsdatum, "", new { @class = "text-danger" })
    </div>
</div>

<div class="form-group">
    @Html.LabelFor(model => model.Beschaffungswert, htmlAttributes: new { @class = "control-label col-md-2" })
    <div class="input-group currencyField">
        @Html.EditorFor(model => model.Beschaffungswert, new { htmlAttributes = new { @class = "form-control
currency", @id = "currencyInput" } })
        <span class="input-group-addon">CHF</span>
    </div>
    @Html.ValidationMessageFor(model => model.Beschaffungswert, "", new { @class = "currencyDanger text-
danger" })
</div>

<div class="form-group">
    <div class="col-md-offset-2 col-md-10">
        <input type="submit" value="Speichern" class="btn btn-default" />
    </div>
</div>
</div>
}

<div>
    @if (Session["BackLink"] != null)
    {
        <a href="@Session["BackLink"]">Zurück</a>
    }
    else
    {
        @Html.ActionLink("Zurück", "Index")
    }
</div>

@section Scripts {

```

```

@Scripts.Render("~/bundles/jqueryval")
@Scripts.Render("~/bundles/autonumeric")

<script>
    new AutoNumeric('#currencyInput', {
        decimalCharacter: '.',
        digitGroupSeparator: '\',
    });

    $.validator.methods.range = function (value, element, param) {
        var globalizedValue = value.replace(".", ",");
        return this.optional(element) || (globalizedValue >= param[0] && globalizedValue <= param[1]);
    }

    $.validator.methods.number = function (value, element) {
        return this.optional(element) || /-?(?:\d+|\d{1,3}(?:[\s\.,]\d{3})+)(?:[\.\,]\d+)?$/i.test(value);
    }

    $("#artikelForm").submit(function (event) {
        value_with_separators = $('#currencyInput').val();
        clean_value = value_with_separators.replace(/'/g, '');
        console.log(clean_value);
        $('#currencyInput').val(clean_value);
    });
</script>
}

```

Artikel.ExportPDF

```

@model IEnumerable<ch.muster.se.inv.dal.Models.Artikel>
@using ch.muster.se.inv.dal.Models

<table border="0" cellspacing="0" width="100%" style="page-break-inside: avoid; page-break-after: always;">
    <tr>
        <td colspan="3">
            <table border="0" cellspacing="0" width="100%" style="margin-botMax:60px;">
                @if
                {
                    int articlePos = 0; // Die jetzige Position des Artikels in der Liste
                }
                @for (int i = 0; i < Model.Count(); i += 4) // Schleife Für die Zeilen
                {
                    <tr>
                        @for (int ii = 0; ii < 4; ii++) // Schleife für die 4 Zellen pro Zeile
                        {
                            if (articlePos < Model.Count())
                            {
                                Artikel a = Model.ToList()[articlePos];
                                <td valign="top" width="25%" style="padding:8px 0px;">
                                    <p style="padding-left:30px;">
                                        <b>muster ag</b><br />
                                        @a.InventarNummer<br />
                                        @a.Bezeichnung
                                    </p>
                                </td>
                            }
                            else // Leere Zeile ohne Inhalt, falls es weniger als 4 Einträge hat
                            {
                                <td valign="top" width="25%" style="padding:8px 0px;">
                                    <p style="padding-left:30px;">
                                </p>
                                </td>
                            }
                            articlePos++;
                        }
                    </tr>
                }
            </table>
        </td>
    </tr>
</table>

```

Artikel.Index

```

@model PagedList<ch.muster.se.inv.dal.Models.Artikel>
@using ch.muster.se.inv.web.Utilities
@using PagedList;
@using PagedList.Mvc;

@{
    ViewBag.Title = "Artikel";

    string currentSort = ViewBag.CurrentSort;
    if (string.IsNullOrEmpty(currentSort))
    {

```

```

        currentSort = "date_desc";
    }
}
</script>
<script>
$(document).ready(function () {
    $('#filterForm select').change(function () {
        document.forms[0].submit();
    })
})
</script>

<div class="jumbotron">
    <h1 class="pageTitle">Suche</h1>

    @using (Html.BeginForm("Index", "Artikel", FormMethod.Get, new { id = "filterform" }))
    {
        <p>
            <div class="col-md-3 no-padding-left">
                @Html.DropDownList("KategorieID", null, "Alle Kategorien", htmlAttributes: new { @class = "form-control col-xs-12" })
            </div>
            <div class="col-md-3">
                @Html.DropDownList("RaumID", null, "Alle Räume", htmlAttributes: new { @class = "form-control col-xs-12" })
            </div>
            <div class="col-md-3">
                @Html.DropDownList("BenutzerID", null, "Alle Nutzer", htmlAttributes: new { @class = "form-control col-xs-12" })
            </div>
            <div class="col-md-3 no-padding-right">
                @Html.DropDownList("FiBuKontoID", null, "Alle FiBu Konten", htmlAttributes: new { @class = "form-control col-xs-12" })
            </div>
            <div class="col-md-6 searchFields no-padding-left">
                <a>@Html.TextBox("Search", null, new { id = "searchbox", placeholder = "Suche", @class = "form-control search-box col-xs-12" })</a>
            </div>
            <div class="col-md-3 searchFields">
                @Html.DropDownList("ArtikelYearsStart", null, "Von (Jahr)", htmlAttributes: new { @class = "form-control col-xs-12" })
            </div>
            <div class="col-md-3 searchFields no-padding-right">
                @Html.DropDownList("ArtikelYearsEnd", null, "Bis (Jahr)", htmlAttributes: new { @class = "form-control col-xs-12" })
            </div>
        </p>
        <p>
            <input type="submit" value="Suchen" class="btn btn-primary btn-filter" />
        </p>
    }
</div>
<p class="divCreateBtn">
    @Html.ActionLink("Artikel erfassen", "Create", null, new { @class = "btn btn-default btn-create-artikel" })
</p>
@if (Model.ToList().Count != 0)
{
    <table class="table tablesorter">
        <tr>
            <th>
                @Html.ActionLink("Inventar Nummer", "Index", Request.QueryString.ToRouteValueDictionary("sortOrder", (string)ViewBag.InventarNummerSortParam))
                @Html.SortIdentifier(currentSort, "InventarNummer")
            </th>
            <th>
                @Html.ActionLink("Artikel", "Index", Request.QueryString.ToRouteValueDictionary("sortOrder", (string)ViewBag.GegenstandSortParam))
                @Html.SortIdentifier(currentSort, "Gegenstand")
            </th>
            <th>
                @Html.ActionLink("Kategorie", "Index", Request.QueryString.ToRouteValueDictionary("sortOrder", (string)ViewBag.KategorieSortParam))
                @Html.SortIdentifier(currentSort, "Kategorie")
            </th>
            <th>
                @Html.ActionLink("Raum", "Index", Request.QueryString.ToRouteValueDictionary("sortOrder", (string)ViewBag.OrtSortParam))
                @Html.SortIdentifier(currentSort, "Ort")
            </th>
            <th>
                @Html.ActionLink("Nutzer", "Index", Request.QueryString.ToRouteValueDictionary("sortOrder", (string)ViewBag.NutzerSortParam))
                @Html.SortIdentifier(currentSort, "Nutzer")
            </th>
            <th>
                @Html.ActionLink("FiBu Konto", "Index", Request.QueryString.ToRouteValueDictionary("sortOrder", (string)ViewBag.FiBuKontoSortParam))
            </th>
        </tr>
    </table>
}

```

```

        @Html.SortIdentifier(currentSort, "FiBuKonto")
    </th>
    <th>
        @Html.ActionLink("Beschaffungsdatum", "Index",
Request.QueryString.ToRouteValueDictionary("sortOrder", (string)ViewBag.BeschaffungsdatumSortParam))
        @Html.SortIdentifier(currentSort, "Beschaffungsdatum")
    </th>
    <th>
        @Html.ActionLink("Beschaffungswert (in CHF)", "Index",
Request.QueryString.ToRouteValueDictionary("sortOrder", (string)ViewBag.BeschaffungswertSortParam))
        @Html.SortIdentifier(currentSort, "Beschaffungswert")
    </th>
    <th></th>
</tr>
@foreach (var item in Model)
{
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.InventarNumber)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Gegenstand)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Kategorie.Bezeichnung)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Ort)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Nutzer.Fullname)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Kategorie.FiBuKonto.Nummer)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Beschaffungsdatum)
        </td>
        <td style="text-align: right">
            @Html.DisplayFor(modelItem => item.Preis)
        </td>
        <td>
            <div class="btn-group">
                <a class="btn btn-default btn-xs" href="/Artikel/Edit/@item.ID">
                    <span class="glyphicon glyphicon-pencil" aria-hidden="true"></span>
                </a>
                <a class="btn btn-default btn-xs" href="/Artikel/Details/@item.ID">
                    <span class="glyphicon glyphicon-info-sign" aria-hidden="true"></span>
                </a>
                <a class="btn btn-default btn-xs" href="/Artikel/Delete/@item.ID">
                    <span class="glyphicon glyphicon-trash" aria-hidden="true"></span>
                </a>
            </div>
        </td>
    </tr>
}
</table>
}
else
{
    <div class="alert alert-danger" role="alert">
        Keine Ergebnisse zu den angegeben Suchkriterien verfügbar
    </div>
}

@Html.PagedListPager(Model, page => Url.Action("Index", new
{
    page = page,
    sortOrder = ViewBag.currentSort
}))
<p>
    <a href="@ViewBag.exportLink" class="btn btn-default @ViewBag.ExportButtons">Artikel exportieren in Excel</a>

    <a href="@ViewBag.exportLinkPDF" class="btn btn-default @ViewBag.ExportButtons">Labels erstellen in PDF</a>
</p>

Benutzer.Create
@model ch.muster.se.inv.dal.Models.Benutzer

@{
    ViewBag.Title = "Create";
}

<h2>Erstellen</h2>

```

```

@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>Benutzer</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })

        <div class="form-group">
            @Html.LabelFor(model => model.Benutzername, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Benutzername, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Benutzername, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Vorname, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Vorname, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Vorname, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Name, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Name, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Name, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Email, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Email, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Email, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.TelefonIntern, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.TelefonIntern, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.TelefonIntern, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.RolleID, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("RolleID", null, "", htmlAttributes: new { @class = "form-control" })
                @Html.ValidationMessageFor(model => model.RolleID, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Erstellen" class="btn btn-default" />
            </div>
        </div>
    </div>
}

<div>
    @if (Session["BackLinkBenutzer"] != null)
    {
        <a href="@Session["BackLinkBenutzer"]">Zurück</a>
    }
    else
    {
        @Html.ActionLink("Zurück", "Index")
    }
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}

```

Benutzer.Delete

@model ch.muster.se.inv.dal.Models.Benutzer

08.09.2018

Seite 119 von 134

```

@{
    ViewBag.Title = "Delete";
}

<h2>Löschen</h2>

<h3>Sind Sie sicher, dass Sie das löschen wollen?</h3>
<div>
    <h4>Benutzer</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.Benutzername)
        </dt>
        <dd>
            @Html.DisplayFor(model => model.Benutzername)
        </dd>
        <dt>
            @Html.DisplayNameFor(model => model.Vorname)
        </dt>
        <dd>
            @Html.DisplayFor(model => model.Vorname)
        </dd>
        <dt>
            @Html.DisplayNameFor(model => model.Name)
        </dt>
        <dd>
            @Html.DisplayFor(model => model.Name)
        </dd>
        <dt>
            @Html.DisplayNameFor(model => model.Email)
        </dt>
        <dd>
            @Html.DisplayFor(model => model.Email)
        </dd>
        <dt>
            @Html.DisplayNameFor(model => model.TelefonIntern)
        </dt>
        <dd>
            @Html.DisplayFor(model => model.TelefonIntern)
        </dd>
        <dt>
            @Html.DisplayNameFor(model => model.Rolle.Name)
        </dt>
        <dd>
            @Html.DisplayFor(model => model.Rolle.Name)
        </dd>
    </dl>
    @using (Html.BeginForm())
    {
        @Html.AntiForgeryToken()

        <div class="form-actions no-color">
            <input type="submit" value="Löschen" class="btn btn-default" /> |
            @if (Session["BackLinkBenutzer"] != null)
            {
                <a href="@Session["BackLinkBenutzer"]">Zurück</a>
            }
            else
            {
                @Html.ActionLink("Zurück", "Index")
            }
        </div>
    }
</div>

```

Benutzer.Details

```
@model ch.muster.se.inv.dal.Models.Benutzer
```

```

@{
    ViewBag.Title = "Details";
}

```



```

<h2>Details</h2>

<div>
  <h4>Benutzer</h4>
  <hr />
  <dl class="dl-horizontal">
    <dt>
      @Html.DisplayNameFor(model => model.Benutzername)
    </dt>

    <dd>
      @Html.DisplayFor(model => model.Benutzername)
    </dd>

    <dt>
      @Html.DisplayNameFor(model => model.Vorname)
    </dt>

    <dd>
      @Html.DisplayFor(model => model.Vorname)
    </dd>

    <dt>
      @Html.DisplayNameFor(model => model.Name)
    </dt>

    <dd>
      @Html.DisplayFor(model => model.Name)
    </dd>

    <dt>
      @Html.DisplayNameFor(model => model.EMail)
    </dt>

    <dd>
      @Html.DisplayFor(model => model.EMail)
    </dd>

    <dt>
      @Html.DisplayNameFor(model => model.TelefonIntern)
    </dt>

    <dd>
      @Html.DisplayFor(model => model.TelefonIntern)
    </dd>

    <dt>
      @Html.DisplayNameFor(model => model.Rolle.Name)
    </dt>

    <dd>
      @Html.DisplayFor(model => model.Rolle.Name)
    </dd>
  </dl>
</div>
<p>
  @Html.ActionLink("Bearbeiten", "Edit", new { id = Model.ID }) |
  @if (Session["BackLinkBenutzer"] != null)
  {
    <a href="@Session["BackLinkBenutzer"]">Zurück</a>
  }
  else
  {
    @Html.ActionLink("Zurück", "Index")
  }
</p>

Benutzer.Edit
@model ch.muster.se.inv.dal.Models.Benutzer

@{
  ViewBag.Title = "Edit";
}

<h2>Bearbeiten</h2>

@using (Html.BeginForm())
{
  @Html.AntiForgeryToken()

  <div class="form-horizontal">
    <h4>Benutzer</h4>
    <hr />

```

```

        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        @Html.HiddenFor(model => model.ID)

        <div class="form-group">
            @Html.LabelFor(model => model.Benutzername, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Benutzername, new { htmlAttributes = new { @class = "form-control" } })
            </div>
            @Html.ValidationMessageFor(model => model.Benutzername, "", new { @class = "text-danger" })
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Vorname, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Vorname, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Vorname, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Name, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Name, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Name, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Email, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Email, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Email, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.TelefonIntern, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.TelefonIntern, new { htmlAttributes = new { @class = "form-control" } })
            </div>
            @Html.ValidationMessageFor(model => model.TelefonIntern, "", new { @class = "text-danger" })
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.RolleID, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("RolleID", null, "", htmlAttributes: new { @class = "form-control" })
                @Html.ValidationMessageFor(model => model.RolleID, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Speichern" class="btn btn-default" />
            </div>
        </div>
    </div>

    <div>
        @if (Session["BackLinkBenutzer"] != null)
        {
            <a href="@Session["BackLinkBenutzer"]">Zurück</a>
        }
        else
        {
            @Html.ActionLink("Zurück", "Index")
        }
    </div>

    @section Scripts {
        @Scripts.Render("~/bundles/jqueryval")
    }

    Benutzer.Index
    @model PagedList<ch.muster.se.inv.dal.Models.Benutzer>
    @using ch.muster.se.inv.web.Utilities
    @using PagedList;
    @using PagedList.Mvc;

    @{
        ViewBag.Title = "Benutzer";

        string currentSort = ViewBag.CurrentSort;
        if (string.IsNullOrEmpty(currentSort))

```

```

    {
        currentSort = "date_desc";
    }
}

<h2>Benutzer</h2>

<p>
    @Html.ActionLink("Benutzer erfassen", "Create", null, new { @class = "btn btn-default" })
</p>
<table class="table">
    <tr>
        <th>
            @Html.ActionLink("Benutzername", "Index", Request.QueryString.ToRouteValueDictionary("sortOrder",
(string)ViewBag.BenutzernameSortParam))
            @Html.SortIdentifier(currentSort, "Benutzername")
        </th>
        <th>
            @Html.ActionLink("Vorname", "Index", Request.QueryString.ToRouteValueDictionary("sortOrder",
(string)ViewBag.VornameSortParam))
            @Html.SortIdentifier(currentSort, "Vorname")
        </th>
        <th>
            @Html.ActionLink("Name", "Index", Request.QueryString.ToRouteValueDictionary("sortOrder",
(string)ViewBag.NameSortParam))
            @Html.SortIdentifier(currentSort, "Name")
        </th>
        <th>
            @Html.ActionLink("E-Mail", "Index", Request.QueryString.ToRouteValueDictionary("sortOrder",
(string)ViewBag.EmailSortParam))
            @Html.SortIdentifier(currentSort, "Email")
        </th>
        <th>
            @Html.ActionLink("Telefon Intern", "Index", Request.QueryString.ToRouteValueDictionary("sortOrder",
(string)ViewBag.TelefonInternSortParam))
            @Html.SortIdentifier(currentSort, "TelefonIntern")
        </th>
        <th>
            @Html.ActionLink("Rolle", "Index", Request.QueryString.ToRouteValueDictionary("sortOrder",
(string)ViewBag.RolleSortParam))
            @Html.SortIdentifier(currentSort, "Rolle")
        </th>
    </tr>
    <tr>
        @foreach (var item in Model)
        {
            <td>
                @Html.DisplayFor(modelItem => item.Benutzername)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Vorname)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Name)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Email)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.TelefonIntern)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Rolle.Name)
            </td>
            <td>
                <div class="btn-group">
                    <a class="btn btn-default btn-xs" href="/Benutzer/Edit/@item.ID">
                        <span class="glyphicon glyphicon-pencil" aria-hidden="true"></span>
                    </a>
                    <a class="btn btn-default btn-xs" href="/Benutzer/Details/@item.ID">
                        <span class="glyphicon glyphicon-info-sign" aria-hidden="true"></span>
                    </a>
                    <a class="btn btn-default btn-xs" href="/Benutzer/Delete/@item.ID">
                        <span class="glyphicon glyphicon-trash" aria-hidden="true"></span>
                    </a>
                </div>
            </td>
        </tr>
    }
</table>

@Html.PagedListPager(Model, page => Url.Action("Index", new
{

```

```

        page = page,
        sortOrder = ViewBag.currentSort
    )))

```

FiBuKonto.Create

```
@model ch.muster.se.inv.dal.Models.FiBuKonto
```

```

@{
    ViewBag.Title = "Create";
}

```

```
<h2>Erstellen</h2>
```

```
@using (Html.BeginForm())
```

```

{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>FiBu Konto</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        <div class="form-group">
            @Html.LabelFor(model => model.Nummer, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Nummer, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Nummer, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Bezeichnung, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Bezeichnung, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Bezeichnung, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Erstellen" class="btn btn-default" />
            </div>
        </div>
    </div>

    <div>
        @if (Session["BackLinkFiBuKonto"] != null)
        {
            <a href="@Session["BackLinkFiBuKonto"]">Zurück</a>
        }
        else
        {
            @Html.ActionLink("Zurück", "Index")
        }
    </div>

```

```

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}

```

FiBuKonto.Delete

```
@model ch.muster.se.inv.dal.Models.FiBuKonto
```

```

@{
    ViewBag.Title = "Delete";
}

```

```
<h2>Löschen</h2>
```

```
<h3>Sind Sie sicher, dass Sie das löschen wollen?</h3>
```

```

<div>
    <h4>FiBu Konto</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.Nummer)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Nummer)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Bezeichnung)

```

```

        </dt>

        <dd>
            @Html.DisplayFor(model => model.Bezeichnung)
        </dd>

    </dl>

    @using (Html.BeginForm())
    {
        @Html.AntiForgeryToken()

        <div class="form-actions no-color">
            <input type="submit" value="Löschen" class="btn btn-default" /> |
            @if (Session["BackLinkFiBuKonto"] != null)
            {
                <a href="@Session["BackLinkFiBuKonto"]">Zurück</a>
            }
            else
            {
                @Html.ActionLink("Zurück", "Index")
            }
        </div>
    }
</div>

```

FiBuKonto.Details

```

@model ch.muster.se.inv.dal.Models.FiBuKonto

@{
    ViewBag.Title = "Details";
}

<h2>Details</h2>

<div>
    <h4>FiBu Konto</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.Nummer)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Nummer)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Bezeichnung)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Bezeichnung)
        </dd>
    </dl>
</div>

<p>
    @Html.ActionLink("Bearbeiten", "Edit", new { id = Model.ID }) |
    @if (Session["BackLinkFiBuKonto"] != null)
    {
        <a href="@Session["BackLinkFiBuKonto"]">Zurück</a>
    }
    else
    {
        @Html.ActionLink("Zurück", "Index")
    }
</p>

```

FiBuKonto.Edit

```

@model ch.muster.se.inv.dal.Models.FiBuKonto

@{
    ViewBag.Title = "Edit";
}

<h2>Bearbeiten</h2>

@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>FiBu Konto</h4>

```

```

<hr />
@Html.ValidationSummary(true, "", new { @class = "text-danger" })
@Html.HiddenFor(model => model.ID)

<div class="form-group">
    @Html.LabelFor(model => model.Nummer, htmlAttributes: new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        @Html.EditorFor(model => model.Nummer, new { htmlAttributes = new { @class = "form-control" } })
        @Html.ValidationMessageFor(model => model.Nummer, "", new { @class = "text-danger" })
    </div>
</div>

<div class="form-group">
    @Html.LabelFor(model => model.Bezeichnung, htmlAttributes: new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        @Html.EditorFor(model => model.Bezeichnung, new { htmlAttributes = new { @class = "form-control" } })
        @Html.ValidationMessageFor(model => model.Bezeichnung, "", new { @class = "text-danger" })
    </div>
</div>

<div class="form-group">
    <div class="col-md-offset-2 col-md-10">
        <input type="submit" value="Speichern" class="btn btn-default" />
    </div>
</div>
</div>
}

<div>
    @if (Session["BackLinkFiBuKonto"] != null)
    {
        <a href="@Session["BackLinkFiBuKonto"]">Zurück</a>
    }
    else
    {
        @Html.ActionLink("Zurück", "Index")
    }
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}

FiBuKonto.Index
@model PagedList<ch.muster.se.inv.dal.Models.FiBuKonto>
@using ch.muster.se.inv.web.Utilities
@using PagedList;
@using PagedList.Mvc;

@{
    ViewBag.Title = "FiBu Konto";

    string currentSort = ViewBag.CurrentSort;
    if (string.IsNullOrEmpty(currentSort))
    {
        currentSort = "date_desc";
    }
}

<h2>FiBu Konto</h2>

<p>
    @Html.ActionLink("FiBu Konto erfassen", "Create", null, new { @class = "btn btn-default" })
</p>
<table class="table">
    <tr>
        <th>
            @Html.ActionLink("FiBu Konto", "Index", Request.QueryString.ToRouteValueDictionary("sortOrder",
                (string)ViewBag.NummerSortParam))
            @Html.SortIdentifier(currentSort, "Nummer")
        </th>
        <th>
            @Html.ActionLink("Bezeichnung", "Index", Request.QueryString.ToRouteValueDictionary("sortOrder",
                (string)ViewBag.BezeichnungSortParam))
            @Html.SortIdentifier(currentSort, "Bezeichnung")
        </th>
        <th></th>
    </tr>

    @foreach (var item in Model)
    {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.Nummer)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Bezeichnung)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Datum)
            </td>
        </tr>
    }
}

```

```

        @Html.DisplayFor(modelItem => item.Bezeichnung)
    </td>
    <td>
        <div class="btn-group">
            <a class="btn btn-default btn-xs" href="/FiBuKonto/Edit/@item.ID">
                <span class="glyphicon glyphicon-pencil" aria-hidden="true"></span>
            </a>
            <a class="btn btn-default btn-xs" href="/FiBuKonto/Details/@item.ID">
                <span class="glyphicon glyphicon-info-sign" aria-hidden="true"></span>
            </a>
            <a class="btn btn-default btn-xs" href="/FiBuKonto/Delete/@item.ID">
                <span class="glyphicon glyphicon-trash" aria-hidden="true"></span>
            </a>
        </div>
    </td>
</tr>
</table>

@Html.PagedListPager(Model, page => Url.Action("Index", new
{
    page = page,
    sortOrder = ViewBag.currentSort
}))

Kategorie.Create
@model ch.muster.se.inv.dal.Models.Kategorie

@{
    ViewBag.Title = "Create";
}

<h2>Erstellen</h2>

@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>Kategorie</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        <div class="form-group">
            @Html.LabelFor(model => model.Bezeichnung, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Bezeichnung, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Bezeichnung, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.FiBuKontoID, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("FiBuKontoID", null, "", htmlAttributes: new { @class = "form-control" })
                @Html.ValidationMessageFor(model => model.FiBuKontoID, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Erstellen" class="btn btn-default" />
            </div>
        </div>
    </div>

    <div>
        @if (Session["BackLinkKategorie"] != null)
        {
            <a href="@Session["BackLinkKategorie"]">Zurück</a>
        }
        else
        {
            @Html.ActionLink("Zurück", "Index")
        }
    </div>

    @section Scripts {
        @Scripts.Render("~/bundles/jqueryval")
    }
}

```

Kategorie.Delete

@model ch.muster.se.inv.dal.Models.Kategorie

08.09.2018

Seite 127 von 134

```

@{
    ViewBag.Title = "Delete";
}

<h2>Löschen</h2>

<h3>Sind Sie sicher, dass Sie das löschen wollen?</h3>
<div>
    <h4>Kategorie</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.Bezeichnung)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Bezeichnung)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.FiBuKonto.Nummer)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.FiBuKonto.Nummer)
        </dd>
    </dl>

    @using (Html.BeginForm())
    {
        @Html.AntiForgeryToken()

        <div class="form-actions no-color">
            <input type="submit" value="Löschen" class="btn btn-default" /> |
            @if (Session["BackLinkKategorie"] != null)
            {
                <a href="@Session["BackLinkKategorie"]">Zurück</a>
            }
            else
            {
                @Html.ActionLink("Zurück", "Index")
            }
        </div>
    }
</div>

```

Kategorie.Details

```

@model ch.muster.se.inv.dal.Models.Kategorie

@{
    ViewBag.Title = "Details";
}

<h2>Details</h2>

<div>
    <h4>Kategorie</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.Bezeichnung)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Bezeichnung)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.FiBuKonto.Nummer)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.FiBuKonto.Nummer)
        </dd>
    </dl>
</div>
<p>
    @Html.ActionLink("Bearbeiten", "Edit", new { id = Model.ID }) |
    @if (Session["BackLinkKategorie"] != null)
    {
        <a href="@Session["BackLinkKategorie"]">Zurück</a>
    }
    else

```



```

    {
        @Html.ActionLink("Zurück", "Index")
    }
</p>

Kategorie.Edit
@model ch.muster.se.inv.dal.Models.Kategorie
@{
    ViewBag.Title = "Edit";
}

<h2>Bearbeiten</h2>

@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>Kategorie</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        @Html.HiddenFor(model => model.ID)

        <div class="form-group">
            @Html.LabelFor(model => model.Bezeichnung, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Bezeichnung, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Bezeichnung, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.FiBuKontoID, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("FiBuKontoID", null, "", htmlAttributes: new { @class = "form-control" })
                @Html.ValidationMessageFor(model => model.FiBuKontoID, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Speichern" class="btn btn-default" />
            </div>
        </div>
    </div>

    <div>
        @if (Session["BackLinkKategorie"] != null)
        {
            <a href="@Session["BackLinkKategorie"]">Zurück</a>
        }
        else
        {
            @Html.ActionLink("Zurück", "Index")
        }
    </div>

    @section Scripts {
        @Scripts.Render("~/bundles/jqueryval")
    }

Kategorie.Index
@model PagedList<ch.muster.se.inv.dal.Models.Kategorie>
@using ch.muster.se.inv.web.Utilities
@using PagedList;
@using PagedList.Mvc;

@{
    ViewBag.Title = "Kategorie";

    string currentSort = ViewBag.CurrentSort;
    if (string.IsNullOrEmpty(currentSort))
    {
        currentSort = "date_desc";
    }
}

<h2>Kategorie</h2>

<p>
    @Html.ActionLink("Kategorie erfassen", "Create", null, new { @class = "btn btn-default" })
</p>
<table class="table">
    <tr>

```

```

        <th>
            @Html.ActionLink("Bezeichnung", "Index", Request.QueryString.ToRouteValueDictionary("sortOrder",
(string)ViewBag.BezeichnungSortParam))
            @Html.SortIdentifier(currentSort, "Bezeichnung")
        </th>
        <th>
            @Html.ActionLink("FiBuKonto", "Index", Request.QueryString.ToRouteValueDictionary("sortOrder",
(string)ViewBag.FiBuKontoSortParam))
            @Html.SortIdentifier(currentSort, "FiBuKonto")
        </th>
    </th></th>
</tr>

@foreach (var item in Model)
{
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.Bezeichnung)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.FiBuKonto.Nummer)
        </td>
        <td>
            <div class="btn-group">
                <a class="btn btn-default btn-xs" href="/Kategorie/Edit/@item.ID">
                    <span class="glyphicon glyphicon-pencil" aria-hidden="true"></span>
                </a>
                <a class="btn btn-default btn-xs" href="/Kategorie/Details/@item.ID">
                    <span class="glyphicon glyphicon-info-sign" aria-hidden="true"></span>
                </a>
                <a class="btn btn-default btn-xs" href="/Kategorie/Delete/@item.ID">
                    <span class="glyphicon glyphicon-trash" aria-hidden="true"></span>
                </a>
            </div>
        </td>
    </tr>
}

</table>

@Html.PagedListPager(Model, page => Url.Action("Index", new
{
    page = page,
    sortOrder = ViewBag.currentSort
}))

Raum.Create
@model ch.muster.se.inv.dal.Models.Raum

@{
    ViewBag.Title = "Create";
}

<h2>Erstellen</h2>

@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>Raum</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        <div class="form-group">
            @Html.LabelFor(model => model.Lokalisierung, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownListFor(model => model.Lokalisierung, ViewData["lokalisierungItems"] as
IEnumerable<SelectListItem>, "", htmlAttributes: new { @class = "form-control" })
                @Html.ValidationMessageFor(model => model.Lokalisierung, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Bezeichnung, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Bezeichnung, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Bezeichnung, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Erstellen" class="btn btn-default" />
            </div>
        </div>
    </div>
}

```

```

    </div>
}

<div>
    @if (Session["BackLinkRaum"] != null)
    {
        <a href="@Session["BackLinkRaum"]">Zurück</a>
    }
    else
    {
        @Html.ActionLink("Zurück", "Index")
    }
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}

Raum.Delete
@model ch.muster.se.inv.dal.Models.Raum
@{
    ViewBag.Title = "Delete";
}

<h2>Löschen</h2>

<h3>Sind Sie sicher, dass Sie das löschen wollen?</h3>
<div>
    <h4>Raum</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.Lokalisierung)
        </dt>
        <dd>
            @Html.DisplayFor(model => model.Lokalisierung)
        </dd>
        <dt>
            @Html.DisplayNameFor(model => model.Bezeichnung)
        </dt>
        <dd>
            @Html.DisplayFor(model => model.Bezeichnung)
        </dd>
    </dl>
    @using (Html.BeginForm())
    {
        @Html.AntiForgeryToken()

        <div class="form-actions no-color">
            <input type="submit" value="Löschen" class="btn btn-default" /> |
            @if (Session["BackLinkRaum"] != null)
            {
                <a href="@Session["BackLinkRaum"]">Zurück</a>
            }
            else
            {
                @Html.ActionLink("Zurück", "Index")
            }
        </div>
    }
</div>

Raum.Details
@model ch.muster.se.inv.dal.Models.Raum
@{
    ViewBag.Title = "Details";
}

<h2>Details</h2>

<div>
    <h4>Raum</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.Lokalisierung)
        </dt>
        <dd>
            @Html.DisplayFor(model => model.Lokalisierung)

```

```

        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Bezeichnung)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Bezeichnung)
        </dd>
    </dl>
</div>
<p>
    @Html.ActionLink("Bearbeiten", "Edit", new { id = Model.ID }) |
    @if (Session["BackLinkRaum"] != null)
    {
        <a href="@Session["BackLinkRaum"]">Zurück</a>
    }
    else
    {
        @Html.ActionLink("Zurück", "Index")
    }
</p>

Raum.Edit
@model ch.muster.se.inv.dal.Models.Raum

@{
    ViewBag.Title = "Edit";
}

<h2>Bearbeiten</h2>

@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>Raum</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        @Html.HiddenFor(model => model.ID)

        <div class="form-group">
            @Html.LabelFor(model => model.Lokalisierung, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownListFor(model => model.Lokalisierung, ViewData["lokalisierungItems"] as
IEnumerable<SelectListItem>, htmlAttributes: new { @class = "form-control" })
                @Html.ValidationMessageFor(model => model.Lokalisierung, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Bezeichnung, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Bezeichnung, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Bezeichnung, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Speichern" class="btn btn-default" />
            </div>
        </div>
    </div>
}

<div>
    @if (Session["BackLinkRaum"] != null)
    {
        <a href="@Session["BackLinkRaum"]">Zurück</a>
    }
    else
    {
        @Html.ActionLink("Zurück", "Index")
    }
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}

```

Raum.Index

```

@model PagedList<ch.muster.se.inv.dal.Models.Raum>
@using ch.muster.se.inv.web.Utilities
@using PagedList;
@using PagedList.Mvc;

@{
    ViewBag.Title = "Raum";

    string currentSort = ViewBag.CurrentSort;
    if (string.IsNullOrEmpty(currentSort))
    {
        currentSort = "date_desc";
    }
}

<h2>Raum</h2>

<p>
    @Html.ActionLink("Raum erfassen", "Create", null, new { @class = "btn btn-default" })
</p>
<table class="table">
    <tr>
        <th>
            @Html.ActionLink("Lokalisierung", "Index", Request.QueryString.ToRouteValueDictionary("sortOrder",
                (string)ViewBag.LokalisierungSortParam))
            @Html.SortIdentifier(currentSort, "Lokalisierung")
        </th>
        <th>
            @Html.ActionLink("Bezeichnung", "Index", Request.QueryString.ToRouteValueDictionary("sortOrder",
                (string)ViewBag.BezeichnungSortParam))
            @Html.SortIdentifier(currentSort, "Bezeichnung")
        </th>
    </tr>

    @foreach (var item in Model)
    {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.Lokalisierung)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Bezeichnung)
            </td>
            <td>
                <div class="btn-group">
                    <a class="btn btn-default btn-xs" href="/Raum/Edit/@item.ID">
                        <span class="glyphicon glyphicon-pencil" aria-hidden="true"></span>
                    </a>
                    <a class="btn btn-default btn-xs" href="/Raum/Details/@item.ID">
                        <span class="glyphicon glyphicon-info-sign" aria-hidden="true"></span>
                    </a>
                    <a class="btn btn-default btn-xs" href="/Raum/Delete/@item.ID">
                        <span class="glyphicon glyphicon-trash" aria-hidden="true"></span>
                    </a>
                </div>
            </td>
        </tr>
    }
</table>

@Html.PagedListPager(Model, page => Url.Action("Index", new
{
    page = page,
    sortOrder = ViewBag.currentSort
}))

```

Shared.Layout

```

@using ch.muster.se.inv.web.Localization

```

```

<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>@ViewBag.Title - Inventar - @Html.Resource("Company")</title>
    @Styles.Render("~/Content/css")
    @Scripts.Render("~/bundles/modernizr")
    @Scripts.Render("~/bundles/jquery")
</head>
<body>
    <div class="navbar navbar-inverse navbar-fixed-top">
        <div class="container">

```

```

        <div class="navbar-header">
            <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
            </button>
            <span class="navbar-brand nav-title">Inventar</span>
        </div>
        @Html.Partial("Header")
    </div>
</div>
<div class="container body-content">
    @RenderBody()
    <hr />
    <footer>
        <p>&copy; @DateTime.Now.Year - @Html.Resource("Company") - Inventar </p>
    </footer>
</div>

@Scripts.Render("~/bundles/bootstrap")
@RenderSection("scripts", required: false)
</body>
</html>

Shared.Header
@using ch.muster.se.inv.web.Utilities
@using ch.muster.se.inv.web.Localization

@if (User.Identity.IsAuthenticated)
{
    @* Der Benutzer muss eingeloggt sein, damit die Navigation angezeigt wird. *@
    <div class="navbar-collapse collapse">
        <ul class="nav navbar-nav">
            <li class="@Html.IsActive("Artikel")">@Html.ActionLink("Artikel", "Index", "Artikel")</li>
            <li class="@Html.IsActive("Kategorie")">@Html.ActionLink("Kategorien", "Index", "Kategorie")</li>
            <li class="@Html.IsActive("Raum")">@Html.ActionLink("Räume", "Index", "Raum")</li>
            @if (this.User.IsInRole("Administrator"))
            {
                <li class="@Html.IsActive("Benutzer")">@Html.ActionLink("Benutzer", "Index", "Benutzer")</li>
            }
            <li class="@Html.IsActive("FiBuKonto")">@Html.ActionLink("FiBu Konten", "Index", "FiBuKonto")</li>
        </ul>
        <ul class="nav navbar-nav navbar-right">
            <li><a href="../Content/Anleitung_Inventar_Verwaltung.pdf"
target="_blank">@Html.Resource("Tutorial")</a></li>
            <li><a class="noLink">@User.Identity.Name</a></li>
            <li>@Html.ActionLink("Logout", "Logout", "Account")</li>
        </ul>
    </div>
}
else
{
    <div class="navbar-collapse collapse">
    </div>
}

<style>
    a.noLink:hover {
        color: #999999 !important;
    }
</style>

```