Zero1one

Dokumentation von Gruppe 2

vorgelegt am 28. Januar 2020

Fakultät Wirtschaft

Studiengang Wirtschaftsinformatik

Kurs [Kursbezeichnung]

von

[Vorname Nachname]

|  |  |
| --- | --- |
| Betreuer in der Ausbildungsstätte: | DHBW Stuttgart: |
| [Name des Unternehmens] [Titel, Vorname und Nachname d. Betreuers] [Funktion des Betreuers] [Unterschrift des Betreuers] | [Titel, Vorname und Nachname des wissenschaftl. Betreuers/Prüfers] |

**Inhaltsverzeichnis**

[Abkürzungsverzeichnis (bei Bedarf) III](#_Toc53905181)

[Abbildungsverzeichnis (bei Bedarf) IV](#_Toc53905182)

[1 Einleitung 1](#_Toc53905183)

[1.1 Anforderungsanalyse 2](#_Toc53905184)

[1.2 Pflichtenheft 3](#_Toc53905185)

[2 Modelle 4](#_Toc53905186)

[2.1 Phasenplan 4](#_Toc53905187)

[2.2 Use-Case-Diagramm 4](#_Toc53905188)

[2.3 Klassendiagramm 7](#_Toc53905189)

[2.4 Entity-Relationship-Modelle 9](#_Toc53905190)

[Literaturverzeichnis 12](#_Toc53905191)

# Abkürzungsverzeichnis (bei Bedarf)

CMS = Configuration Management System

CI = Configuration Item

ERM = Entity-Relationship-Modell

ITIL = Information Technology Infrastructure Library

# Abbildungsverzeichnis (bei Bedarf)

[Abb. 1: Phasenplan 4](#_Toc53904840)

[Abb. 2: Use-Case-Diagramm CMS 5](#_Toc53904841)

[Abb. 3: Klassendiagramm CMS 8](#_Toc53904842)

[Abb. 4: Berechtigungsdatenbank-ERM 10](#_Toc53904843)

[Abb. 5: CI-Datenbank-ERM 10](#_Toc53904844)

1. Einleitung

Durch die Information Technology Infrastructure Library (ITIL), welche als eines der meist genutzten Frameworks für das IT Service Management gilt,[[1]](#footnote-1) werden verschiedene Konzepte und Leitlinien zur praktischen Handhabung des IT Service Managements zur Verfügung gestellt.[[2]](#footnote-2) Einer der in ITIL beschriebener Prozesse stellt hierbei das Configuration Management dar.

Das Configuration Management dient der Handhabung von IT-Komponenten, wobei sichergestellt werden soll, dass die Assets ordentlich kontrolliert werden und dass korrekte Informationen über diese verfügbar sind.[[3]](#footnote-3) Das Ziel ist somit die Bereitstellung einer zentralen Informationsquelle über alle IT-Komponenten, Ressourcen und deren Abhängigkeiten untereinander. Hierfür werden Informationen vor allem über die Konfiguration der Assets und die Beziehung zwischen den einzelnen Assets in einem Configuration Management System (CMS) dokumentiert und in einer dazugehörigen Configuration Management Database abgespeichert. Die einzelnen Assets und deren Konfigurationsdetails werden hierbei anhand von Configuration Records festgehalten.[[4]](#footnote-4)

Die einzelnen Assets oder Komponenten werden als Configuration Item (CI) bezeichnet. Die unterschiedlichen CIs können hierbei stark in Komplexität, Größe und Art variieren. So kann ein CI ein komplettes System inklusive aller Hardware, Software und Dokumentationen oder auch nur eine kleine Hardwarekomponente darstellen.[[5]](#footnote-5) Um die vielen unterschiedlichen CIs besser zu strukturieren, werden unterschiedliche Configuration Item Typen definiert, welche üblicherweise alle eingesetzten Hardware und Software Komponenten, wie zum Beispiel Server oder Datenbanken, beinhalten. Jedes CI und der dazugehörige Configuration Record ist somit genau einem Configuration Item Typ zugeordnet.[[6]](#footnote-6) Die CIs werden anhand von Configuration Records, welche alle für das bestimmte CI relevanten Informationen beinhalten, in einem CMS festgehalten. Hierbei beschreibt jeder Configuration Record genau ein bestimmtes CI, wobei jedes CI ein eindeutiges Kennzeichen haben muss.[[7]](#footnote-7)

In der nachfolgenden Dokumentation ist nun detailliert der Entwurf, die Entwicklung und die anschließende Qualitätssicherung und Wartbarkeit eines CMS beschrieben. Dabei werden zunächst in den nachfolgenden Abschnitten die genauen funktionalen und nicht funktionalen Anforderungen von der Seite des Auftraggebers an das CMS betrachtet und anschließend deren Umsetzung durch das Projektteam dargestellt.

* 1. Anforderungsanalyse

Nach der groben Beschreibung des Auftrags sollen nun die konkreten funktionalen und anschließend die nicht funktionalen Anforderungen an das CMS betrachtet werden.

Mithilfe des CMS sollen alle möglichen verschiedenen CI-Typen verwaltet werden können. Bei der Erstellung eines neuen CI-Typs sollen auch für diesen Typ übliche Attribute erstellt werden können, sodass nicht für jeden CI-Record die Attribute einzeln erstellt werden müssen, sondern bereits die vordefinierten Attribute des jeweiligen Typs vorhanden sind. Außerdem sollen die CI-Typen auch gelöscht werden können, wenn diese nicht mehr benötigt werden, um die Übersichtlichkeit des Systems zu erhöhen.

Um die einzelnen CIs im CMS festzuhalten soll für jedes CI ein CI-Record mit einem bestimmten CI-Typ einfach erstellt werden können. Hierbei sollen neben den bereits vordefinierten Attributen noch weitere hinzugefügt werden können, um auch alle besonderen Konfigurationen eines speziellen CIs im zugehörigen CI-Record abzuspeichern. Des Weiteren sollen die CI-Records auch im Nachhinein noch änderbar sein, damit die CI-Records bei Neuerungen schnell aktualisiert werden können und nicht neu erstellt werden müssen. Da viele CIs nach bestimmter Zeit auch ausgetauscht werden, sollen die zugehörigen CI-Records im CMS gelöscht werden können. Außerdem sollen die CI-Records einsehbar und möglichst schnell auffindbar sein, sodass man die gespeicherten Informationen möglichst schnell und einfach finden kann.

Um die Übersichtlichkeit noch weiter zu erhöhen, soll angezeigt werden, wie viele CI-Records zu einem bestimmten CI-Typ vorhanden sind. Wenn möglich sollten auch Diagramme in Form von Kreis oder Balkendiagrammen zur Veranschaulichung der Mengenverteilung erstellt werden.

Eine weitere angeforderte Funktionalität stellt das Login dar. Hierbei soll der Benutzer sich mit einem Benutzernamen und einem Passwort anmelden können und bei den richtigen Eingaben Zugang zur Software erlangen. Je nach der Berechtigung des angemeldeten Benutzers kann dieser dann unterschiedliche Tätigkeiten durchführen. Nach Beendigung der Tätigkeiten soll der Benutzer sich wieder ausloggen können, damit die Sitzung beendet wird und niemand Unberechtigtes Zugang erhält.

Die Benutzer sollen in unterschiedliche Benutzergruppen unterschieden werden, sodass verschiedene Berechtigungen vergeben werden können. Eine Benutzergruppe soll hierbei berechtigt sein, neue Benutzer anzulegen, wobei der Benutzername festgelegt und ein initiales Passwort vergeben wird, sodass der Benutzer sich erstmal mit dem vorher festgelegten Passwort anmelden kann und dieses dann anschließend ändern kann. Bei der Erstellung des Benutzers soll auch die dem Benutzer zugeordnete Berechtigung vergeben werden. Diese Rechte sollen auch später noch bearbeitbar sein und gegebenenfalls der Benutzername abgeändert werden können. Außerdem sollen Benutzer auch gelöscht werden können.

Neben den funktionalen Anforderungen gibt es auch noch die nicht funktionalen Anforderungen an das CMS, welche nun genauer betrachtet werden sollen. Dabei soll für eine möglichst gute Bedienbarkeit gesorgt werden, sodass die Benutzung möglichst intuitiv erfolgen kann und nicht anhand einer komplizierten Anleitung befolgt werden muss. Eine weitere Eigenschaft, wie die Bedienbarkeit verbessert werden kann, ist die Plausibilitätsprüfung bei den Eingaben. Des Weiteren soll die Performance des Systems optimiert werden, indem zum Beispiel die Antwortzeiten möglichst geringgehalten werden. Außerdem soll die Software wartbar und auch änderbar sein, damit diese in Zukunft angepasst werden kann, wenn es Änderungsbedarf gibt. Eine wichtige Anforderung an das System stellt auch die Sicherheit dar, da das CMS viele wichtige Daten beinhaltet, welche nur durch Berechtigte anpassbar sein sollen und nicht durch Unberechtigte manipuliert werden können.

* 1. Pflichtenheft

Nach der Beschreibung der Anforderungen an das CMS soll nun deren Umsetzung durch das Projektteam grob betrachtet und in den nachfolgenden Kapiteln vertieft werden.

Grundsätzlich wird für das Tool eine Webanwendung erstellt, in welcher die Funktionen des CMS verwendet werden können. So können hier die CI-Typen inklusive der dazugehörigen CI-Records angezeigt werden und nach einem bestimmten Namen eines CI-Records oder eines CI-Typen durchsucht werden, damit schnell das Gewünschte gefunden werden kann.

Des Weiteren können in der Webanwendung CI-Records und CI-Typen schnell angelegt werden, wobei beliebig viele Attribute definiert werden können, um alle besonderen Konfigurationsdetails zu definieren. Hier können diese auch geändert, indem neue Attribute hinzugefügt werden, oder gelöscht werden.

Zusätzlich kann ein Benutzer seine Daten ändern. Die Benutzer werden hierbei in zwei Benutzergruppen, Admin und sonstige Benutzer, unterschieden. Die Benutzer, welche als Admin angemeldet sind, können in der Webanwendung auch die Benutzer verwalten.

Die Oberfläche der Webseite wurde hierbei mit HTML erstellt und mit CSS gestylt. Aufgrund von Vorkenntnissen in der Softwareentwicklung mit Java wurde die Datenbankanbindung mithilfe von Java implementiert, wobei das Springframework zur Vereinfachung verwendet wurde, da dieses den Zugriff auf die Datenbank erleichtert und eine integrierte Datenbank anbietet. Zur Unterstützung für die Darstellung der Daten auf der Webseite wurde neben Spring auch Thymeleaf verwendet. Die Verwendung der Frameworks und die Umsetzung der im Lastenheft definierten Anforderungen wird in den nachfolgenden Kapiteln detaillierter dargestellt.

1. Modelle

In dem nachfolgenden Kapitel soll nun der Entwurf für das CMS anhand von verschiedenen Modellen dargestellt werden. Zuerst wird hierbei der allgemeine Phasenplan für die Durchführung der Fallstudie beschrieben. Anschließend werden vertieft die Funktionalitäten anhand eines Use-Case-Diagrammes und deren Implementierung anhand eines Klassendiagrammes betrachtet und der Aufbau der Datenbank anhand eines Entity-Relationship-Modells (ERM) erklärt.

* 1. Phasenplan

Um den Ablauf der Durchführung zu verdeutlichen, werden die Phase in einem Phasenplan dargestellt (siehe Abb. 1).



Abb. 1: Phasenplan

Zu Beginn der Fallstudie fand eine Organisation der Gruppenmitglieder in unterschiedliche Kleingruppen statt, um die verschiedenen Aufgaben besser zu verteilen. Hierbei wurde vor allem aufgeteilt, wer sich um das Frontend und wer sich um das Backend kümmert. Nachdem die Anforderungen analysiert wurden, wurde deren Umsetzung entschieden und ein Entwurf für das Programm erstellt.

Anschließend wurde das System in den zuvor gebildeten Kleingruppen erstellt. Nach der Erstellung von Frontend und Backend, wurden diese zusammengeführt, wobei kleine Anpassungen vorgenommen wurden, damit beides optimal zueinander passt.

Nachdem das System fertiggestellt wurde, wurde dieses noch ausgiebig von allen Gruppenmitgliedern getestet, um sicherzugehen, dass dies auch auf allen Rechnern gut funktioniert. Hierbei wurden noch einzelne Dinge optimiert und das System schließlich zur Abgabe fertiggestellt.

* 1. Use-Case-Diagramm

Um die verschiedenen Funktionalitäten des CMS detaillierter darzustellen, wurde ein Use-Case-Diagramm (siehe Abb. 2) zur Veranschaulichung erstellt.

Ein Use-Case-Diagramm dient zur Darstellung, wie ein Benutzer des Systems, auch gennant Akteur, verschiedene Tätigkeiten in dem System durchführt. Ein Use-Case beschreibt hierbei den Ablauf, welcher ein Akteur bei der Durchführung einer Tätigkeit durchläuft, um ein bestimmtes Ziel zu erreichen.[[8]](#footnote-8)

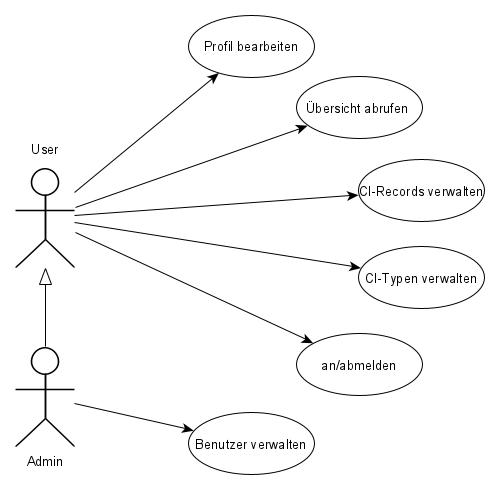


Abb. 2: Use-Case-Diagramm CMS

Als Akteure agieren in dem CMS die Benutzer, wobei die Benutzer in Admins und sonstige Benutzer unterschieden werden. Hierbei sind die Admins Benutzer mit zusätzlichen besonderen Berechtigungen. So können die Admins die gleichen und noch weitere Funktionalitäten wie die anderen Benutzer nutzen. Im Folgenden werden nun die einzelnen Use-Cases detaillierter betrachtet.

Der erste Use Case ist hierbei „Profil bearbeiten“. Jeder Benutzer kann sein Profil bearbeiten, wobei dieser seinen Benutzername und das Passwort ändern kann. Die Voraussetzung für diesen Use Case ist, dass der Benutzer bereits am System angemeldet ist. Diese Funktionalität kann genutzt werden, indem der Benutzer auf sein Profil geht und die entsprechenden Änderungen einträgt. Um das Passwort zu ändern, muss das aktuelle Passwort und das neue zweimal eingegeben werden, damit eine versehentliche falsche Eingabe eines neuen Passworts verhindert wird. Somit wäre ein sekundäres Szenario, dass das aktuelle Passwort falsch oder zwei unterschiedliche neue Passwörter eingegeben werden. In diesen Fällen wird eine Fehlermeldung angezeigt, damit der Benutzer seine Eingaben überprüfen kann. Wenn sich keine Widersprüche ergeben, werden die Daten des Benutzers aktualisiert und in der Datenbank gespeichert. Bei der nächsten Anmeldung im System kann der Benutzer sich nun mit den neuen Daten anmelden.

Ein weiterer Use Case stellt hierbei die „Übersicht abrufen“ dar. Eine Übersicht über die CI-Records und die CI-Typen kann angezeigt werden, indem in der Navigationsleiste das „Dashboard“ aufgerufen wird. Anschließend kann der Benutzer alle vorhandenen CI-Typen und die dazugehörigen CI-Records einsehen. Hierbei ist auch vorausgesetzt, dass der Benutzer am System angemeldet ist, damit niemand Unberechtigtes die gespeicherten Informationen einsehen kann.

Bei dem Use-Case „CI-Records verwalten” werden unterschiedliche Funktionalitäten zusammengefasst. So kann ein Benutzer sowohl CI-Records erstellen, bearbeiten und löschen. Bei der Erstellung muss hierbei der CI-Typ, der Name des CIs und die Werte der dazugehörigen Attribute angegeben werden. Um eine fehlerhafte Eingabe des CI-Typs zu vermeiden, kann dieser hier anhand einer Drop-down-Liste ausgewählt werden. Wie bei den vorherigen Use-Cases ist auch hier vorausgesetzt, dass der Benutzer am System angemeldet ist. Um CI-Records zu ändern oder löschen, muss der Benutzer den entsprechenden CI-Record auswählen und kann diesen dann durch das Betätigen des jeweiligen Buttons bearbeiten oder auch löschen.

Um CI-Records zu erstellen, müssen auch CI-Typen definiert werden. Dies wird durch den Use-Case „CI-Typen verwalten“ beschrieben. Um neue CI-Typen zu erstellen muss lediglich ein Name für den CI-Typ und die Namen der dazugehörigen Attribute definiert werden. Hierbei muss der Name des CI-Typs eindeutig sein, da es nicht mehrere CI-Typen mit gleichen Namen geben kann. Ein sekundäres Szenario wäre somit, dass ein CI-Typ erstellt werden soll, der schon vorhanden ist. Bei der Erstellung eines neuen CI-Typs können hierbei beliebig viele Attribute erstellt werden. Wenn CI-Typen nicht mehr benötigt werden, können diese auch gelöscht oder auch nur geändert werden. Hierfür muss der entsprechende CI-Typ ausgewählt werden und kann dann durch das Betätigen eines Buttons geändert oder gelöscht werden. Analog zu den vorher beschriebenen Use Cases muss auch hier der Benutzer angemeldet sein.

Ein weiterer Use Case, welchen alle Benutzer anstoßen können, ist das „an/abmelden“. Um Zugang zum CMS zu erlangen, müssen die Benutzer sich anmelden. Hierfür müssen diese ihren Benutzernamen und ihr Passwort im Anmeldefenster eingeben. Wenn die Eingaben stimmen, erhält dieser Zugang. Durch das Anmelden wird dafür gesorgt, dass nur Berechtigte Zugriff auf die im CMS gespeicherten Daten erhalten. Um die Sitzung zu beenden, können sich die Benutzer durch das Betätigen des Logout-Buttons abmelden. Eine Voraussetzung für das Anmelden ist hierbei, dass der Benutzer bereits registriert ist und ein initialer Benutzername und Passwort vergeben wurde. Die Voraussetzung für das Abmelden ist, dass der Benutzer in dem Moment angemeldet ist. Ein sekundäres Szenario wäre hierbei, dass der Benutzer den Benutzername oder das Passwort falsch eingibt. In diesem Fall erscheint eine Fehlermeldung und der Benutzer kann seine Eingaben nochmal überprüfen.

Die Benutzer werden hierbei durch die Admins angelegt. Dies wird durch den Use-Case „Benutzer verwalten“ beschrieben. Die Admins haben somit eine andere Ansicht als die anderen Benutzer, da diese im Profil noch die Möglichkeit haben einen neuen Benutzer zu erstellen. Bei der Erstellung eines neuen Benutzers vergeben die Admins den Benutzernamen und ein Passwort, wobei der Name des Benutzers eindeutig sein muss. Wenn also ein Benutzer mit bereits vorhandenem Namen angelegt werden soll, erscheint hierbei eine Fehlermeldung. Um einen neuen Benutzer anzumelden ist vorausgesetzt, dass der Admin im System angemeldet ist und eine entsprechende Berechtigung hat.

* 1. Klassendiagramm

Nach der Funktionenerfassung anhand des Use-Case-Diagrammes (siehe Abschnitt 2.2) wurde auch ein Klassendiagramm (siehe Abb. 3) in der Unified Modeling Language (UML) zum Entwurf der Software erstellt. UML ist eine standardisierte Sprache zur Modellierung von objektorientierten Klassen.[[9]](#footnote-9)

In dem nachfolgenden Klassendiagramm werden zur besseren Übersichtlichkeit der Beziehungen zwischen den Klassen des Diagrammes nur die Klassennamen dargestellt, da die Variablen der Klassen mit den im ERM (siehe 2.4) dargestellten Attributen übereinstimmen. Die Methoden der Klassen sind sogenannte Get- und Set- Methoden, mit denen die Instanzvariablen geändert oder abgefragt werden können, da die Variablen private sind, um die Daten zu schützen.

In UML werden verschiedene Beziehungsarten zwischen den Klassen definiert. Die Assoziation wird hierbei durch eine einfache Linie dargestellt. Eine Assoziation bedeutet, dass die Objekte zweier Klassen etwas miteinander zu tun haben. Die genaue Beziehung wird anhand einer kurzen Beschreibung an der Verbindungslinie und anhand der Multiplizität definiert. Über die Multiplizität kann angegeben werden, wie viele Objekte einer Klasse an der Beziehung beteiligt sind.[[10]](#footnote-10)

Eine Verfeinerung der Assoziation stellt die Aggregation und die Komposition dar. Die Aggregation wird durch eine unausgefüllte und die Komposition anhand einer schwarz ausgefüllten Raute dargestellt. Eine Aggregation ist eine Beziehung, die eine „Ist-Teil von Beziehung“ beschreibt. Eine Komposition ist eine Aggregation mit engerer Beziehung, da dort die Lebensdauer beider Instanzen gleich ist.[[11]](#footnote-11)

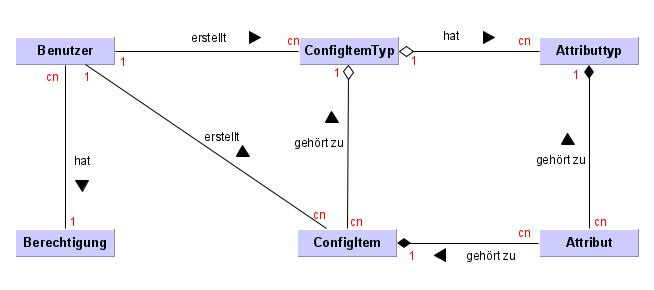


Abb. 3: Klassendiagramm CMS

Analog zum Akteur Benutzer im Use-Case-Diagramm gibt es hier eine Klasse Benutzer, welche die jeweiligen Attribute des Benutzers definiert. Da ein Admin auch ein Benutzer mit ähnlichen Eigenschaften ist, werden dessen Daten auch anhand einer Instanz der Klasse Benutzer gespeichert. Die Berechtigungsgruppen werden hierbei dann anhand der zugeordneten Berechtigungsinstanz definiert, wobei jeder Benutzer nur eine Berechtigung haben kann. Die Benutzer erstellen dann null bis beliebig viele ConfigItems und ConfigItemTypen.

Da ein ConfigItemTyp aus vielen ConfigItems besteht, handelt es sich hierbei um eine Aggregation. Allerdings kann ein ConfigItem auch nicht mehr zu einem bestehenden ConfigItemTyp zuordbar sein, wenn beispielsweise ein ConfigItemTyp gelöscht wurde, ohne dass zuvor alle dazugehörigen ConfigItems gelöscht wurden. Da die ConfigItems nicht automatisch gelöscht werden sollen, damit keine Daten verloren gehen, können diese auch noch ohne den dazugehörigen ConfigItemTyp bestehen. Deshalb handelt es sich hierbei nicht um eine Komposition, sondern um eine Aggregation.

Die vordefinierten Attribute für den ConfigItemTyp werden hierbei in einer anderen Klasse festgehalten, damit eine beliebige Anzahl an Attributen zu einem ConfigItemTyp hinzugefügt werden können. Die Attributarten werden hierbei in der Klasse Attributtyp definiert. In der Klasse Attribut wird dann einem bestimmten ConfigItem ein Wert zu einem gewissen Attributtyp zugewiesen. Zum Beispiel kann für den ConfigItemTyp „Drucker“ eine Instanz der Klasse AttributTyp mit dem Namen „Anschaffungsjahr“ erstellt werden. Wenn dann ein ConfigItem-Record des Typs Drucker erstellt wird, wird in einer Instanz der Klasse Attribut das Anschaffungsjahr des bestimmten Druckers festgehalten. Da ein Attribut genau zu einem Attributtyp und einem ConfigItem gehört und nur existiert, solange der dazugehörige Attributtyp und der CI-Record existiert, handelt es sich hierbei um Kompositionen.

Hierbei werden Attributtypen immer zu einem bestimmten ConfigItemTyp erstellt, falls unterschiedliche ConfigItemTypen einen gleichen Attributtyp besitzen, wird hierfür also ein weiterer Attributtyp mit der gleichen Bezeichnung erstellt. Deshalb handelt es sich hierbei um eine „1:n“-Beziehung, da kein Attributtyp mehreren ConfigItemTypen zugeordnet werden kann.

* 1. Entity-Relationship-Modelle

Nach der Beschreibung des Klassendiagramms, wird in diesem Abschnitt der Entwurf für die Datenbank beschrieben. Hierfür wurden Entity-Relationship-Modelle (ERM) erstellt. Dabei wurde zwischen der Datenbank für die Configuration Items (siehe Abb. 5) und der Berechtigungsdatenbank (siehe Abb. 4) unterschieden, da nicht festgehalten wird, wer welchen CI-Record oder CITyp erstellt hat, sodass die Verbindung zwischen Benutzer und CI-Record beziehungsweise CI-Typ nicht in der Datenbank gespeichert wird.

Das ERM ist ein Modell, um Daten strukturiert darzustellen. Hierbei entspricht eine objektorientierte Klasse einem Entitätstyp des ERM, weshalb das ERM und das Klassendiagramm einander sehr ähnlich sind. Eine Entität ist ein eindeutig identifizierbares Objekt, welches sowohl konkrete Gegenstände wie Gebäude oder Personen oder auch abstrakte Konzepte wie Verträge oder Gesetze darstellen kann. Ähnliche Entitäten werden hierbei zu Entitätstypen zusammengefasst, welche verschiedene Eigenschaften haben. Diese Eigenschaften werden als Attribute bezeichnet. Im ERM werden die Beziehungen zwischen den verschiedenen Entitätstypen mit den dazugehörigen Multiplizitäten dargestellt. Die Multiplizität beschreibt, wie viele Entitäten eines Entitätstypen an der Beziehung beteiligt sind.[[12]](#footnote-12)



Abb. 4: Berechtigungsdatenbank-ERM

In der Usertabelle wird der Name und das Passwort des Benutzers gespeichert. Außerdem wird noch die BerechtigungsID als Fremdschlüssel aus der Berechtigungstabelle in der Benutzertabelle abgespeichert, da jeder Benutzer eine bestimmte Berechtigung hat. In der Berechtigungstabelle wird die Bezeichnung der jeweiligen Berechtigung festgehalten und als eindeutiger Schlüssel eine ID vergeben.



Abb. 5: CI-Datenbank-ERM

In der CI-Datenbank werden alle Informationen bezüglich der CI-Records und CI-Typen festgehalten. Beim ConfigItemTyp wird lediglich der Name festgehalten, welcher zugleich auch als Schlüssel dient. Das bedeutet, dass der Name eindeutig sein muss und nicht öfters vorkommen darf. Die dazugehörigen Attribute werden hier in der AttributTyp Tabelle mit einem Namen und dem ConfigItemTypnamen als Fremdschlüssel des dazugehörigen ConfigItemTyp gespeichert.

Ebenso hat die Tabelle, in der die ConfigItems gespeichert werden, den Namen des übergeordneten ConfigItemTyp als Fremdschlüssel gespeichert, damit zuordbar ist, zu welchem ConfigItemTyp das ConfigItem gehört. Außerdem hat das ConfigItem einen Namen und eine eindeutige ID als Schlüssel.

Der Schlüssel des ConfigItems wird als Fremdschlüssel in der Attribut-Tabelle festgehalten, sodass eindeutig zuordbar ist, zu welchem ConfigItem das Attribut gehört. Außerdem wird zur Zuordnung des Attributs zu einem bestimmten Attributtyp der Fremdschlüssel gespeichert. In der Attribut-Tabelle wird dann der Wert des Attributtyps eines bestimmten CI-Records abgespeichert.

Literaturverzeichnis

**Goll, J. (2011):** Methoden und Architekturen der Softwaretechnik, 1. Aufl., Wiesbaden: Vieweg+Teubner Verlag

**Gross, J. (2008):** Best Practice im wertorientierten IT-Servicemanagement — zwei Frame-works im Vergleich, in: HMD Praxis der Wirtschaftsinformatik, 45. Jg., Nr. 6, S. 38–47

**Leimeister, J. M. (2020):** Service Management und Service Operations, in: Dienstleistungsengineering und -management, (Hrsg.: Leimeister, J. M.), 2nd ed. 2020, Berlin, Heidelberg: Springer Berlin Heidelberg, S. 279–342

**Marrone, M./Kolbe, L. M. (2011):** Einfluss von IT-Service-Management-Frameworks auf die IT-Organisation, in: WIRTSCHAFTSINFORMATIK, 53. Jg., Nr. 1, S. 5–19

**Rance, S. (2011):** ITIL, Service Transition, 2011 ed., 2. impr, London: TSO The Stationery Office

**Xu, L. D./Chaudhry, S. S./Tjoa, A. M. (2008):** Research and Practical Issues of Enterprise Information Systems II, Volume 2 IFIP TC 8 WG 8.9 International Conference on Research and Practical Issues of Enterprise Information Systems (CONFENIS 2007) October 14-16, 2007, Beijing, China, 255. Jg., Boston, MA: International Federation for Information Processing

1. Vgl. Marrone/Kolbe 2011, S. 5 [↑](#footnote-ref-1)
2. Vgl. Gross 2008, S. 38 [↑](#footnote-ref-2)
3. Vgl. Leimeister 2020, S. 299 ff. [↑](#footnote-ref-3)
4. Vgl. Rance 2011, S. 89 f. [↑](#footnote-ref-4)
5. Vgl. Rance 2011, S. 93 [↑](#footnote-ref-5)
6. Vgl. Rance 2011, S. 308 f. [↑](#footnote-ref-6)
7. Vgl. Rance 2011, S. 94 [↑](#footnote-ref-7)
8. Vgl. Xu/Chaudhry/Tjoa 2008, S. 1142 f. [↑](#footnote-ref-8)
9. Vgl. Goll 2011, S. 316 [↑](#footnote-ref-9)
10. Vgl. Goll 2011, S. 336 [↑](#footnote-ref-10)
11. Vgl. Goll 2011, S. 311 [↑](#footnote-ref-11)
12. Vgl. Goll 2011, S. 258 f. [↑](#footnote-ref-12)