E-Healthpraktikum – Lessons learned – OPS-5

Inhaltsverzeichnis

Meilenstein 1

In dem ersten Meilenstein haben wir ein einfaches Datenmodell für ein OPS aufgestellt. Dabei konnten wir die theoretischen Inhalte aus der Vorlesung nochmal vertiefen und auf eine explizite Situation anwenden, also sowohl wie man ein Datenmodell erstellt als auch wie genau ein OPS aufgebaut ist. Vor der Implementierung der Datenbank war eine gründliche Konzipierung des Schemas notwendig. Dabei konnten wir nochmal verinnerlichen, wie genau Tabellen mittels Schlüsseln verknüpft werden und wie die Umsetzung von n:m-Beziehungen mit Hilfe einer weiteren Tabelle genau geschieht. Theoretisches Wissen wird durch die Praxis verdeutlicht und vertieft. Es gibt viele Kleinigkeiten, die uns beim Erlernen der Theorie nicht unbedingt aufgefallen sind und wir erst bei der Umsetzung bemerkt haben, dass sie in der Praxis sehr wichtig sind. Wie zum Beispiel: Die Kennzeichnung von Primär- und Fremdschlüsseln sollte stets gegeben sein. 1:1 Beziehungen sind in der Praxis unüblich, beziehungsweise in Datenbanken eigentlich überflüssig und somit werden einer der Tabellen dann alle Attribute der zweiten Tabelle zugeordnet.

Außerdem konnten wir die Wichtigkeit von Stammtabellen kennenlernen, um die Qualität der Inhalte der Bewegungstabellen auf Basis von eingeschränkten Datentypen zu gewährleisten.
Wir haben insgesamt viele Sachen hinzulernen können, die für ein guten Datenbankentwurf wichtig sind. Dazu zählt zum einen, dass wir zuvor nicht wussten, dass auch die Stammtabellen mit in das ER-Diagramm gehören und als richtige Tabellen aufgefasst werden und nicht nur als Domänen betrachtet werden.
Außerdem haben wir die Wichtigkeit von der Benennung von Attributen und Tabellen unterschätzt. Sinnvolle eindeutige Attribut- und Tabellennamen erleichtern spätere SQL-Abfragen aber auch generell die Lesbarkeit der Datenbank. Dabei sollte man stets einheitlich sein, was Groß- und Kleinschreibung angeht, aber auch dass man, wenn mehrere Wörter in einem Namen vorkommen, immer das gleiche Trennzeichen nutzt. Außerdem sollte man drauf achten, Dinge unterschiedlich zu benennen, um die Eindeutigkeit aufrecht zu halten. Dabei haben wir ein paar Attribute gehabt, die etwas irreführend waren, da diese nicht das widergespiegelt haben, was der Name des Attributes aussagt.
In Bezug auf den medizinischen Aspekt haben wir dazulernen können, dass zusätzliche Klartextdiagnosen unbedingt mit angegeben werden müssen, damit dem Arzt die nötige Flexibilität gewährleistet wird noch zusätzliche Informationen zum ICD-Code hinzuzufügen. Uns war ebenfalls nicht bewusst, wie man am besten dokumentieren könnte, wie man die Ersteller/Bearbeiter und zugehörigen Zeiten am besten umsetzen könnte. Dabei haben wir zu kompliziert gedacht und wollten eine eigene Tabelle pro Bewegungstabelle machen, um jede mögliche Veränderung speichern zu können. Da dies für den Rahmen unseres Praktikums zu umfangreich ist, wurde uns dann geraten nur die neuste Veränderung zu speichern. Somit konnte man die Bearbeitungsdaten einfach als Attribute zu den Bewegungstabellen hinzufügen.
Wir haben also bereits im 1. Meilenstein sehr viel hinzulernen können zur Erstellung eines guten Datenbankentwurfs.

Meilenstein 2

Im zweiten Schritt sollten wir das Datenmodell in eine Datenbank überführen. Dafür haben wir erst die MySQL Workbench genutzt um das Schema aufzubauen. Es wurden alle Tabellen inklusive Attributen und Domänen erstellt und dann in Beziehung zueinander gesetzt. Dies war mittels der Workbench relativ intuitiv, da man die Tabellen grafisch darstellen konnte und diese einfach durch nacheinanderigem Anklicken mit einer Beziehung verknüpfen konnte. Somit wurde vieles einfacher gemacht und hat uns auch Zeit erspart. Darüber hinaus wurden auch automatisch Indizes erstellt, wodurch man dies ebenfalls nicht mehr manuell erledigen musste. Nachdem das Schema stand, sollten wir es in phpMyAdmin übertragen, um es zur MariaDB hinzuzufügen. Dies konnte man innerhalb der Workbench relativ einfach durchführen. Dabei sind uns allerdings immer wieder Fehler unterlaufen, da wir ein paar Inkonsistenzen anhand der Domänen oder Ähnliches hatten. Dies war aber gestützt von der Workbench, sodass in die MariaDB wirklich nur Tabellen/Beziehungen geladen werden können, die keine Fehler mehr aufweisen. Anschließend sollten wir die vorgegebenen Stammtabellen, die uns als txt-Dateien vorlagen importieren. Dies war über php sehr einfach, da man manuell festle-

gen konnte, wie die einzelnen Spalten und Zeilen voneinander getrennt sind (bei den einzelnen TXT-Dateien ein wenig unterschiedlich). Beim Importieren der Stammdaten ist uns noch aufgefallen, dass die Stammdaten nicht ganz zu unserer Datenbank passen und wir somit unser Schema etwas anpassen mussten, also weitere Attribute hinzugefügt haben. Am Schluss haben wir noch ein paar Beispieldatensätze hinzugefügt.
Auch in diesem Meilenstein haben wir durch ein paar Fehler, die wir gemacht haben, viel lernen können. Zum einen haben wir gelernt, dass man mit Datentypen sehr vorsichtig sein muss, da viele Menschen unterschiedliche Auffassungen haben, wie man z.B. IDs festlegt. Somit hatten wir von Anfang an alle IDs auf den Typ Integer gesetzt. Als unsere Datenbank so gut wie fertig war und wir die Stammtabelle für die Mitarbeiter einfügen wollten, ist uns aufgefallen, dass diese IDs führende Nullen haben und somit nicht in den Typ Integer übertragen werden können. Da an der ID viele Fremdschlüssel hängen, konnte man den Datentyp nicht einfach ändern, da dies zu Inkonsistenzen geführt hätte. Somit mussten wir noch einmal die ganze Datenbank zurücksetzen und unsere Änderungen vornehmen. Also sollte man immer vorsichtig und vorausschauend mit Datentypen umgehen. Zusätzlich ist es auch noch sinnvoll, passende Datentypen und Längen zu wählen und Nullwerte zu erlauben, um die Integrität einer Datenbank zu wahren und Daten vollständig und korrekt eingeben zu können.
Außerdem haben wir zum Teil redundante Attribute gespeichert, zum Beispiel in der Operation sowohl Beginn, Datum, Ende, als auch Dauer. Dabei könnte z.B. das Ende oder die Dauer außen vorgelassen werden, da dies aus den restlichen Attributen berechnet werden kann und damit auch weniger Inkonsistenzen innerhalb der Daten auftreten können.
Auch in diesem Meilenstein hatten wir mit der Benennung/Beschreibung von Attributen noch ein paar Probleme, da wir uns nicht einig waren, wie man beispielsweise OP-Säle benennt.
Außerdem ist es, gerade in Hinblick auf den Meilenstein 3 sehr sinnvoll bei IDs mit AutoIncrement zu arbeiten, sodass diese fortlaufend sind und nicht immer manuell festgelegt werden müssen.
Zusammenfassend lässt sich also zeigen, dass wir auch hier nochmal vieles hinzulernen konnten, was eine gute Datenbank ausmacht und generell die Werkzeuge kennengelernt haben, mit denen man dies umsetzen kann.

Meilenstein 3

Der Meilenstein 3 teilt sich in zwei Teile auf. Im ersten Teil passen wir ein vorgegebenes GUI-Template an, mit dem man Patienten/Fälle/Operationen hinzufügen kann und generell anzeigen/bearbeiten/… lassen kann. Außerdem müssen wir eine Kommunikationsschnittstelle zwischen einer KIS- und einer OPS-Gruppe umsetzen. Im zweiten Teil sollen wir zusätzliche fallbezogene Zusatzinformationen aus internetbasierten Webdiensten abrufen und visualisieren können.

GUI-Template und Kommunikationsschnittstelle

GUI-Template und Verknüpfung mit Datenbank

Um möglichst einfach SQL-Statements in Java einzubetten und dabei eine gewisse Typsicherheit beibehalten zu können, haben wir uns entschieden jOOQ zu verwenden. Dies ermöglicht

objektorientiert Anfragen zu stellen, ohne dabei SQL-Statements im String-Format verwenden zu müssen. Mit dieser Bibliothek haben wir uns erst einmal auseinandergesetzt und haben in dieser Hinsicht ein, unserer Meinung nach, sehr gutes Tool kennengelernt. Nachdem wir dies alles eingerichtet hatten, haben wir die einzelnen Unteraufgaben des dritten Meilensteins, also M3.2.2 bis M3.2.5 unter uns vier Gruppenmitgliedern aufgeteilt, um möglichst effizient die komplexe Aufgabe in mehrere kleine aufzuteilen, um es zeitlich bis zum gegebenen Termin zu schaffen. Dafür haben wir mit unterschiedlichen Branches in Git gearbeitet und sie am Ende in den Master gemergt, damit es während des Arbeitens nicht ständig zu Mergekonflikten kommt.
Insgesamt lässt sich sagen, dass sowohl unsere Programmierfähigkeiten, der Umgang mit jOOQ aber auch der Umgang mit FXML und dem Scene Builder deutlich vorangeschritten sind und die Aufgabenaufteilung in unserer Gruppe echt gut funktioniert hat. Wir haben uns gegenseitig bei Fragen unsere Erkenntnisse präsentiert, sodass der Programmierstil (in Bezug auf jOOQ) möglichst einheitlich ist. Dabei haben wir als Kommunikation zwischen Programmcode und Datenbank die DAOs benutzt, was unserer Meinung nach eine sehr schöne und elegante Weise für den Zugriff ist.
Unser Ziel war es etwa 1-2 Wochen vor Abgabe mit den Punkten fertig zu werden, um uns dann in der letzten Woche an M3.2.6 zu setzen, also der Kommunikationsschnittstelle. Dies haben wir auch bis auf wenige Bugs geschafft.

Kommunikationsschnittstelle

Für die Kommunikation zwischen dem KIS und dem OPS nutzen wir die HAPI Library. Diese stellt nicht nur einfache Parsingmöglichkeiten für HL7-Nachrichten zur Verfügung, sondern auch Client und Server zum Empfangen und Senden von Nachrichten.
Zum Empfangen von Nachrichten initialisieren wir zu Beginn (wenn das Programm gestartet wird) einen Hapiserver. Dieser hört auf ADTA01 Nachrichten (und auf BARP05 zum Testen, wenn wir uns selbst etwas zuschicken). Der Server läuft die gesamte Zeit in der die Applikation läuft. Zum Senden wird kurzzeitig ein Client von HAPI erstellt, der die Nachricht sendet und auf das ACK wartet. Um beim Senden einer Nachricht an das KIS mögliche kurzweilige Störungen zu umgehen, kann die Nachricht mehrfach gesendet werden. Dafür wird das Timeout der Verbindung auf 7 Sekunden gesetzt. Wenn nach diesem Timeout keine Antwort vom empfangenden System erhalten wurde, wird die Nachricht erneut gesendet. Dies wird wiederholt, bis die Nachricht fünfmal gesendet wurde. Wenn dann immer noch keine Antwort vom KIS vorliegt, wird dem Nutzer ein Alert mit der Meldung „Die Nachricht kann nicht gesendet werden.“ angezeigt. Der Client wird auf einen zusätzlichen Thread geladen, damit die GUI bedienbar bleibt. Wenn das Senden fertig ist, wird der Thread auch wieder unterbrochen.

Zum Parsen der ADT und BAR Nachrichten, haben wir uns zuerst den Aufbau der HL7-Nachrichten angeschaut und geschaut, welche Segmente wir mit welchen Informationen von unseren Instanzen des Patienten/Fall/Operation/… belegen müssen. Diese haben wir wie folgt zugeordnet:

MSH-Segment:

MSH.3	Sending Application	OPS5
MSH.5	Receiving Application	KIS2
MSH.7	Date/Time of Message	LocalDate.now() (aktueller Zeitpunkt)
MSH.9	Message Type	BAR^P05^BAR_P05
MSH.13	Sequence Number	123

EVN-Segment:

EVN.1	Event Type Code	P05
EVN.2	Recorded Date/Time	LocalDate.now() (aktueller Zeitpunkt)

PID-Segment:

PID.2	Patient ID	Patient.ID
PID.5.1	Family Name	Patient.Name
PID.5.2	Given Name	Patient.Vorname
PID.7	Date of Birth	Patient.Geburtsdatum
PID.8	Sex	Patient.Geschlecht (transformiert in ISS-Standard)
PID.11.1	Street address	Patient.Straße
PID.11.5	Zip or postal code	Patient.Postleitzahl
PID.13	Phone Number - Home	Patient.Telefonnummer
PID.23	Birth Place	Patient.Geburtsort

PV1-Segment:

PV1.2	Patient Class	Fall.Falltyp
PV1.3	Assigned Patient Location	Fall.Station
PV1.4	Admission Type	Patient.Notfall (Emergency oder Routine)
PV1.17.1	Admitting Doctor – ID number	Ersteller.ID
PV1.17.2	Admitting Doctor – Family Name	Ersteller.Name
PV1.17.3	Admitting Doctor – Given Name	Ersteller.Vorname
PV1.17.6	Admitting Doctor – Prefix	Ersteller.Anrede
PV1.17.7	Admitting Doctor – Degree	Ersteller.Titel
PV1.19	Visit Number	Fall.ID
PV1.44	Admit Date/Time	Fall.Aufnahmedatum
PV1.45	Discharge Date/Time	Fall.Entlassungsdatum

DG1-Segment:

DG1.1	SET ID - Diagnosis	Diagnose.DiagnoseID
DG1.3	Diagnosis Code	Diagnose.ICD10Code
DG1.4	Diagnosis Description	Diagnose.Klartextdiagnose
DG1.5	Diagnosis Date/Time	Diagnose.Datum
DG1.16.1	Diagnosing Clinician – ID number	Diagnose.Ersteller

DG1.16.2	Diagnosing Clinician – Family Name	Ersteller.Name
DG1.16.3	Diagnosing Clinician – Given Name	Ersteller.Vorname
DG1.16.6	Diagnosing Clinician – Prefix	Ersteller.Anrede
DG1.16.7	Diagnosing Clinician – Degree	Ersteller.Titel

PR1-Segment:

PR1.1	SET ID - Procedure	Prozedur.ProzID
PR1.3	Procedure Code	Prozedur.OPSCode
PR1.4	Procedure Description	Prozedur.Anmerkung
PR1.5	Procedure Date/Time	Prozedur.Erstellzeit
PR1.11.1	Surgeon – ID number	Prozedur.Ersteller
PR1.11.2	Surgeon – Family Name	Ersteller.Name
PR1.11.3	Surgeon – Given Name	Ersteller.Vorname
PR1.11.6	Surgeon – Prefix	Ersteller.Anrede
PR1.11.7	Surgeon – Degree	Ersteller.Titel

ROL-Segment:

ROL.3.1	Role - Identifier	Role.RolleSt
ROL.3.2	Role - Text	RolleSt.Bezeichnung
ROL.4.1	RolePerson - ID	MedPersonal.PersID
ROL.4.2	RolePerson – Family Name	MedPersonal.Name
ROL.4.3	RolePerson – Given Name	MedPersonal.Vorname
ROL.4.6	RolePerson – Prefix	MedPersonal.Anrede
ROL.4.7	RolePerson – Degree	MedPersonal.Titel

Wenn das KIS uns eine Nachricht sendet, dann ist der Ersteller/Bearbeiter immer das KIS. Dafür haben wir eine eigene Instanz bei den Mitarbeitern erstellt.

Um eine Nachricht zu erstellen, wird erst einmal eine BARP05 Nachricht mit den benötigten Segmenten erstellt. Die einzelnen Felder der Segmente können nun mittels Getter- und Settermethoden geholt und bearbeitet werden. Somit lassen sich sehr einfach die Nachrichten bauen (wenn wir etwas verschicken wollen) und vom KIS empfangene Nachrichten in einen Patienten, … parsen. Da die Felder, die wir belegen nicht ganz mit denen übereinstimmen, die das KIS belegt, ist das Format der Nachrichten sehr gut, da so nur die Felder ausgelesen werden müssen, die das jeweilige System benötigt. Um diese Nachrichten zu erstellen haben wir die Klasse MessageParser erstellt, die die jeweilige Operation nimmt und dann in die BARP05 Nachricht umwandelt oder die ADT01 Nachricht nimmt und daraus den jeweiligen Patienten/Fall/Diagnose erstellt, der/die dann in die Datenbank eingefügt werden kann.

Beim Empfangen wird zunächst überprüft, ob es sich um einen neuen/validen Patient handelt. Wenn er neu ist, wird dieser in die DB eingefügt, andernfalls wird dieser Patient mit den gesendeten Parametern geupdatet.
Danach wird das gleiche für den Fall getan, also geprüft, ob dies ein valider Fall und ein neuer/vorhandener Fall ist.
Danach wird geschaut, ob die ADT Nachricht Diagnosen enthält und wenn ja müssen diese ebenfalls eingefügt/bearbeitet werden, je nachdem ob diese schon existieren oder nicht. Bei einer neuen Operation wird geprüft, ob es schon eine Operation zu den Fall gibt, wenn nicht wird eine neue Operation erstellt, an diese die Diagnose hinzugefügt wird. Andernfalls wird die zuletzt hinzugefügte Operation (mit der höchsten ID) gewählt.
Die Prozeduren müssen laut Betreuern nicht beachtet werden, da diese (ausschließlich) von dem OPS erstellt werden, somit machen wir hier dies nicht für die Prozeduren.

Synchronisierung mit der Partnergruppe

Um die fehlerfreie Kommunikation mit der Partnergruppe zu gewährleisten, haben wir uns darauf geeinigt die Daten initial zu synchronisieren und dass wir den Fall ausschließen (Empfehlung der Tutoren), dass beide Gruppen gleichzeitig z.B. einen neuen Patienten einfügen und die IDs beim anschließenden Verschicken dann nicht mehr übereinstimmen. Wir gehen also davon aus, dass die Datenbank bezüglich der Primärschlüssel immer synchron ist. Wenn einem die ID des empfangenen Datensatzes noch nicht bekannt ist, wird von dem Patienten/Fall ein neuer erstellt und eingefügt. In dem Falle, dass beispielsweise die DB bis zum Patient 10 synchron ist und das OPS dann 3 Patienten mit 11, 12 und 13 als Notfall aufnimmt, aber nur die 13 sendet, wird der Patient im KIS ebenfalls mit der ID 13 eingefügt und mittels Autoincrement bekommen alle folgenden Einträge höhere IDs. So bleiben die beiden Systeme immer synchron. Dieser Datensatz kann jederzeit wieder raufgespielt werden über php. Die entsprechenden SQL-Statements sind in einer Datei in dem Git-Repository hinterlegt.

Abruf von externem Wissen mittels PubMed

Die Aufgabe 3.3 erfordert die Abfrage von der externen Webseite „PubMed“. Als Erstes sollten die ICD10 Codes in MeSH-Terme übersetzt werden und dann sollen diese MeSH-Terme als Suchbegriff in die PubMed-Anfrage eingefügt werden. Dies haben wir mit JSON ermöglicht. Unser Problem war, dass einige ICD10 Codes nicht übersetzt werden konnten und kein Ergebnis in der Suche lieferten. Um dieses Problem zu lösen, haben wir der Suche zwei weitere Varianten hinzugefügt. Wenn der originale ICD10 Code kein Ergebnis liefert, wird mit dem „short code“ (den ersten drei Codestellen) gesucht. Die Ergebnisse dieser Suche liefern allgemeinere Informationen zu dem eigentlich gesuchten Code. Wenn es mit dem short code auch nicht funktioniert, startet die Suche nach der vierten Stelle (die Stelle nach „.“) der Codes um „0“ bis „9“ ergänzt. Die Codes mit „!“ und dem „*“ am Ende sind aus dem Datensatz gelöscht, somit kann die Suche nach diesen vernachlässigt werden.

Nachdem wir den Link zum PubMed vervollständigt hatten, sollten wir nach einer Möglichkeit suchen dem Benutzer die erhaltenen Informationen anzuzeigen. Wir haben uns entschieden

ein zusätzliches Fenster zu öffnen, in dem der Inhalt des Links angezeigt wird. Dafür haben wir dann die WebView.fxml Datei erstellt, welche eine WebView von JavaFx enthält. Der dazugehörige Controller enthält eine Methode, welche mithilfe der angegebenen URL die PubMed Seite in die WebView lädt.
Diese Seite wird dem Benutzer angezeigt, sobald er den Info Button neben den ICD-10 Codes in dem Diagnose Fenster gedrückt hat. In dem geöffneten Fenster kann der Benutzer dann Artikel zu der gesuchten Krankheit finden. Über die zwei Pfeile oben gelangt der Benutzer auch wieder aus den Artikeln zurück auf die eigentliche PubMed Seite.

Andere Designentscheidungen

Overview

In dem Übersichts-Tab gibt es drei Tabellen. In der ersten sind die Patienten mit ihren Daten aufgelistet. Klickt man auf einen dieser Patienten, füllt sich die zweite Tabelle mit den Fällen, die zu diesem Patienten gehören. Die dritte Tabelle mit den Operationen füllt sich wiederum erst nach Anklicken eines Falles mit den Operationen, die diesem Fall angehören.
Wenn man nun noch eine Operation auswählt, erscheint ein Button mit der Aufschrift ‚Rolle'. Dieser zeigt eine Übersicht der Rollen der Operation und erlaubt das Bearbeiten sowie das Erstellen neuer Rollen.

Wenn man eine Operation auswählt und auf diese einen Doppelklick macht, wird ein Fenster zum Bearbeiten dieser geöffnet. Dabei werden die Felder automatisch mit den Werten gefüllt, die die Operation bis jetzt besessen hat.

Die beiden Buttons ‚Diagnosen' und ‚Prozeduren' öffnen ebenfalls je ein neues Fenster mit der Übersicht über die Diagnosen beziehungsweise Prozeduren. Dabei unterscheiden wir, ob zu dem Zeitpunkt eine Operation ausgewählt ist oder nicht. Falls keine Operation gewählt ist, werden alle Diagnosen / Prozeduren aus der Datenbank angezeigt, sonst nur die zu der entsprechenden Operation gehörenden. Dies wird dem Nutzer auch über einen Alert mitgeteilt.
In der Diagnosen-Übersicht (analog dazu die Prozeduren und Rollen) kann nun eine neue Diagnose eingefügt werden oder eine Diagnose in der Tabelle wird zum Bearbeiten ausgewählt. In diesem Fall werden die zugehörigen Werte in die Felder zum Bearbeiten eingetragen. Über den ‚Bearbeiten'- bzw. ‚Speichern'-Button wird die Diagnose dann geupdated.

Es ist ebenfalls möglich sich alle Fälle einer Station anzeigen zu lassen. Dabei kann man über die Comboxbox die Station auswählen. Wenn eine Station ausgewählt wurde, werden alle Fälle von allen Patienten angezeigt, die dort auf der Station liegen. Sobald man dann einen anderen Patienten auswählt, wird die Filterung nach Stationen aufgehoben und wieder alle Fälle angezeigt (in dem Fall dann nur die Fälle dieses Patienten).

Generell muss zum Bearbeiten aus der Liste eine angeklickt werden, Diese ausgewählte Diagnose/Prozedur/Rolle/Operation (also auf Basis der ID) wird dann mit den neuen Werten überschrieben.

Stornieren

Zum Stornieren von Operationen ist in der Übersicht ein Button ‚Stornieren' vorgesehen. Ist eine Operation ausgewählt und der Button wird betätigt, wird diese Operation mitsamt ihren Diagnosen, Prozeduren und Rollen storniert. Um sich die stornierten Operationen anzeigen zu lassen betätigt man die CheckBox die sich über der Tabelle befindet. Wählt man eine stornierte Operation und drückt erneut den ‚Stornieren'-Button, wird die Stornierung (inklusive Diagnosen, Prozeduren und Rollen) wieder rückgängig gemacht.
Bei den Diagnosen und Prozeduren gibt es diese CheckBox nicht, hier ist in den Tabellen vermerkt, welche davon storniert sind und welche nicht (letzte Spalte) und bleiben somit in der allgemeinen Übersicht über alle Diagnosen/Prozeduren sichtbar.
Das Stornieren von Patienten und Fällen liegt in der Hand des KIS und muss somit nicht von uns implementiert werden.

Logging

Um auch in der Konsole nachvollziehen zu können, welche Schritte das Programm gerade vornimmt, haben wir uns entschieden Logging zu verwenden. Dies ist übersichtlicher und gibt mehr Informationen als ein einfaches Schreiben in die Konsole mit dem print-Befehl. Verwendet werden dazu Klassen aus dem java-package java.util.logging. Somit kann man besser verfolgen, von welcher Klasse diese Nachricht kommt. Dabei werden Änderungen und Einfügungen sowie Stornierungen angezeigt, aber auch wenn man eine falsche Eingabe getätigt hat. Somit bekommt man alle Informationen zum Einen in der Komandozeile, aber auch in der GUI als Alert.

Attribute zur Synchronisierung

Um kenntlich zu machen, dass der Patient vom OPS aufgenommen wurde, und es sich somit um eine Notfallaufnahme handelt, haben wir bei dem Patienten eine weitere Spalte (notfall) hinzugefügt. Wenn der Patient von unserem System aus gesetzt wird, setzen wir den Wert auf true. Wenn das KIS und einen neuen Patienten schickt, setzen wir das Feld auf false. In dem PV1 Segment machen wir dies beim Senden auch kenntlich. Wenn es sich um einen Notfallpatienten handelt, setzen wir den Aufnahmetyp (PV1.4) auf Emergency (E), ansonsten (also wenn es ein Patient vom KIS ist), setzen wir den Wert auf R (Routine).
Des weiteren haben wir bei der Operation ein Feld eingefügt, welches angibt, ob es sich um eine normale Operation handelt, die vom OPS erstellt wird, oder ob es eine Dummy-Operation ist, da das KIS uns Diagnosen zu einem Fall mitgeschickt hat. Da bei uns die Diagnosen am Fall hängen, müssen wir solch eine Dummy-Operation erstellen, damit wie die Diagnosen ebenfalls einfügen können. Dieses Attribut heißt geplant, und wenn dies true ist, bedeutet es, dass es sich um eine solche Dummy-Operation handelt. Diesen Wert setzen wir, wenn das KIS uns eine Diagnose mitschickt, bei dem der angegebene Fall noch keine Operation hat. In den anderen Fällen, also wenn wir eine Operation erstellen, wird der Wert auf false gesetzt.

Login von Mitarbeiter

Damit gewährleistet ist, dass immer jemand eingeloggt ist, haben wir zunächst (wenn das Programm gestartet wird) ein Login Fenster geöffnet, in welchen sich ein Mitarbeiter mit seinem Passwort verifizieren muss. Das Passwort lautet für jeden Mitarbeiter „OPS5“. Innerhalb der GUI kann dann direkt der Mitarbeiter gewechselt werden, oder wieder zurück zum Login Fenster gewechselt werden, bis sich jemand anderes einloggen möchte.

Versenden

Beim Versenden von Operationen kann man diese mithilfe der gegebenen Checkbox auswählen. Die darunterliegende Übersicht zeigt sowohl die versendeten, aber auch die empfangenen Nachrichten an. Dabei wird zum einen die HL7-Nachricht dargestellt, genauso wie der Zeitpunkt, an dem die Nachricht versendet/empfangen wurde. In der letzten Spalte gültig, wird angezeigt, wenn die Nachricht gültig war. Dabei werden die Nachrichten zunächst auf nicht gültig gesetzt. Wenn es eine gesendete Nachricht ist, wird es auf ja gesetzt, wenn das ACK ein „AA“ war. Wenn es sich um eine empfangene Nachricht handelt, wird gültig auf ja gesetzt, wenn Patient/Fall/(Diagnosen) erfolgreich hinzugefügt/geupdatet wurden. Es können nur Operationen versendet werden, die nicht storniert sind (die stornierten werden rausgefiltert).

Ungültige Abfragen

Wenn eine ungültige Eingabe gemacht wurde, werden Alerts geworfen, sodass der Nutzer weiß, was genau er ändern muss. Dabei fangen wir sowohl ab, wenn es Werte sind, die nicht null sein dürfen aber auch semantische Fälle. Dabei fangen wir folgende Fälle ab:

- Patient anlegen: Vor- und Nachname müssen existieren und das Geburtsdatum darf nicht in der Zukunft liegen
- Fall anlegen: Entlassungsdatum darf nicht vor dem Aufnahmedatum sein, wenn nichts für das Aufnahmedatum gewählt wurde, dann wird das heutige Datum gesetzt
- Op anlegen: für die 4 Zeitpunkt muss folgender Zusammenhang gelten: OP-Beginn vor Schnittzeit vor Nahtzeit vor OP-Ende, wenn die Bauchtücher ungleich sind, wird darauf aufmerksam gemacht
- Es dürfen bei den ICD-10-Codes nur endständige Codes eingefügt werden.

Generell prüfen wir auch bei allen Textfeldern, ob HL7-reservierte Symbole verwendet wurden, also ^,~,& und \. Diese dürfen generell nicht verwendet werden. Außerdem überprüfen wir bei den Textfeldern, ob diese zu lang für das jeweilige Feld in der DB ist. Somit kann es nicht zu Insert-Problemen kommen, wenn zu lange Felder versucht werden einzufügen.

Rückblick

Rückblickend haben wir durch das Praktikum vieles dazulernen können. Wir haben erstmals Datenbanken in der Praxis kennenlernen dürfen und somit bei der Modellierung als auch bei der Umsetzung und der Programmierung viele neue wichtige Punkte dazugelernt. Ebenfalls

konnten wir viele Punkte der E-Healthvorlesung (wie unter anderen die HL7-Nachrichten) vertiefen und verinnerlichen. Über diese Punkte hinaus, konnten wir ebenfalls unsere Softskills verbessern, da wir die Arbeit in der Gruppe und die Aufteilung der Arbeit managen mussten.