**NHPlus**

**- User-Stories, Akzeptanzkriterien, Testfälle -**

1. Plane innerhalb deiner Gruppe die noch ausstehenden Module, indem du
   1. User-Stories formulierst
   2. über die Formulierung von Akzeptanzkriterien definierst, wann eine User-Story fertig implementiert ist
   3. aus der User-Story und ihren Akzeptanzkriterien Tasks ableitest. **Die Tasks sollen genau festlegen, welche Klassen neu zu implementieren und welche Klassen an welcher Stelle um welches Element anzupassen sind.**
   4. abschließend aus den Akzeptanzkriterien für die Testphase Testfälle ableitest.

Benutze dafür die unten abgebildete Tabellenform. Für weitere User-Stories ist die Tabellenstruktur zu kopieren.

|  |
| --- |
| **User-Story**  Als **Datenschutzbeauftragter** möchte ich **nicht mehr die Daten des Vermögensstandes erfassen**, um der DSGVO zu entsprechen. |
| **Akzeptanzkriterien**  A\_1: Alle bisherigen Daten zum Vermögensstandes der Patienten sind gelöscht  A\_2: Es sind keine Daten bezüglich des Vermögensstandes erfassbar  A\_3: Vermögensstand wurde aus der Tabellarischen Übersicht entfernt |
| **Testfälle**  **TF1\_: Entfernung aus der GUI**  - **Vorbedingung**: Der Nutzer hat im Hauptfenster der Anwendung die Option Patienten/innen ausgewählt  - **auszuführende Testschritte**: -  - **erwartetes Ergebnis**: Es werden alle Patienten, die in der Datenbank gespeichert sind, mit ihrer ID, Nachnamen, Vornamen, Geburtsdatum, Pflegestufe und Raumnummer in einer Tableview angezeigt  ohne die Spalte Vermögensstand und ohne das Textfeld für das Hinzufügen eines Patienten mit Vermögensstand abzuzeigen.  **TF2\_: Entfernung aus der Datenbank**  **- Vorbedingung:** Der Administrator ist auf der Datenbank eingeloggt.  **- auszuführende Testschritte:** Select Befehl auf Patient-Tabelle für Spalte ASSETS ausführen  **- erwartetes Ergebnis:** Fehler beim Ausführen des Select-Befehls, dass die Spalte nicht gefunden wurde  **TF3\_: Entfernung aus Modell**  - **Vorbedingung**: Der Nutzer hat im Hauptfenster der Anwendung die Option Patienten/innen ausgewählt. TF2\_ wurde erfolgreich ausgeführt.  - **auszuführende Testschritte**: Einen neuen Patienten hinzufügen. Auf den Tab Behandlungen wechseln.  Wieder in den Tab Patienten/innen wechseln  **- erwartetes Ergebnis:** Der soeben angelegte Patient ist mit den zuvor bestimmten Daten in der Tabelle vorhanden |
|  |
| **Tasks**  T\_1: Aus der Modellklasse Patient muss das Attribut „asset“ entfernt werden F  T\_2: Aus der Klasse PatientDAO müssen aus allen SQL-Strings der„assets“-Teil entfernt werden F T\_3: Aus der Datenbank muss aus der Tabelle Patient die Spalte „assets“ entfernt werden  T\_4: Aus dem Tabelview in AllPatientView muss die Spalte „colAssets“ entfernt werden T\_5: Aus dem AllPatientController müssen die Methoden zum Anzeigen der Patienten, Ändern, Löschen und Hinzufügen nicht mehr den Vermögensstand behandeln T\_6: Aus AllPatientView muss das Textfeld „txtAssets“ entfernt werden |
| **User-Story**  Als Haupt-Arzt möchte ich die Löschung der persönlichen Daten der Patienten erst 10 Jahre nach der letzten Behandlung erlauben um die DSGVO einzuhalten. |
| **Akzeptanzkriterien**  A\_1: Löschung der Patientendaten ist erst nach 10 der Letzten Behandlung möglich.  A\_2: Wenn der Patient Löschung beantragt während der 10 Jahre, sollen die Daten gesperrt werden.  A\_3: Entsperrung der Daten soll auch möglich sein  A\_4: Wenn der Patient gesperrt ist dürfen nur limitiert Daten gezeigt werden. |
| **Tasks**  T\_1: Extra Feld zur Klasse „Patient“ hinzufügen um zu signalisieren das die Patientendaten gesperrt sind z.B: „Locked“ / „IsLocked“.  T\_2: Überprüfung zu denn Methoden im „AllPatientController“: „handleOnEditRoomnumber“, „handleOnEditCareLevel“,“handleOnEditDateOfBirth“, „handleOnEditSurname“,“handleOnEditFirstname“,“handleDeleteRow“ hinzufügen die auf das neue Feld und die 10 Jahres Periode checken damit die Daten nicht verändert werden können wenn sie gesperrt oder die 10 Jahre noch nicht vergangen sind.  T\_3: Neuen Button: „Daten sperren“ zur „AllPatientView.fxml“ hinzufügen um die Daten des Ausgewählten Patienten zu sperren.  T\_4: Neuen Button: „Daten entsperren“ zur „PatientView.fxml“ hinzufügen um die Daten des Ausgewählten Patienten zu entsperren.  T\_5: Die nötige Interaktions Logik für den Button: „Daten sperren“ im „AllPatientController“ hinzufügen.  T\_6: Die nötige Interaktions Logik für den Button: „Daten entsperren“ im „AllPatientController“ hinzufügen. |
| **Testfälle**  **TF1\_:**  - **Vorbedingung**:  Programm starten und zur Patienten Ansicht navigieren.  Patient anlegen danach zur Behandlungs-Ansicht navigieren und eine Behandlung für den Patienten anlegen mit einem Aktuellen Datum.  - **auszuführende Testschritte**:  in der Patienten Ansicht denn vorher angelegten Patienten versuchen zu bearbeiten  und die Änderungen zu speichern.  - **erwartetes Ergebnis**:  Da die Daten vom Patienten nicht gesperrt sind sollte die Bearbeitung erfolgreich sein.  **TF2\_:**  - **Vorbedingung**:  Programm starten und zur Patienten Ansicht navigieren.  Patient anlegen danach zur Behandlungs-Ansicht navigieren und eine Behandlung für den Patienten anlegen mit einem Aktuellen Datum und dann auf den Button „Daten sperren“ klicken.  - **auszuführende Testschritte**:  in der Patienten Ansicht denn vorher angelegten Patienten versuchen zu bearbeiten  und die Änderungen zu speichern.  - **erwartetes Ergebnis**:  Da die Daten vom Patienten gesperrt sind sollte die Bearbeitung nicht erfolgreich sein und es sollte ein Popup kommen mit einer Fehlernachricht.  **TF3\_:**  - **Vorbedingung**:  Programm starten und zur Patienten Ansicht navigieren.  Patient anlegen danach zur Behandlungs-Ansicht navigieren und eine Behandlung für den Patienten anlegen mit dem Datum „28.02.2010“ und dann auf den Button „Daten sperren“ klicken.  - **auszuführende Testschritte**:  in der Patienten Ansicht denn vorher angelegten Patienten versuchen den Patienten zu Löschen.  - **erwartetes Ergebnis**:  Da die Letzte Behandlung vom Patienten außerhalb der 10 Jahre ist sollte die Löschung erfolgreich durchgehen.  **TF4\_:**  - **Vorbedingung**:  Programm starten und zur Patienten Ansicht navigieren.  Patient anlegen danach zur Behandlungs-Ansicht navigieren und eine Behandlung für den Patienten anlegen mit einem Aktuellen Datum und dann auf den Button „Daten sperren“ klicken.  - **auszuführende Testschritte**:  in der Patienten Ansicht denn vorher angelegten Patienten versuchen den Patienten zu Löschen.  - **erwartetes Ergebnis**: Da die Letzte Behandlung vom Patienten innerhalb der 10 Jahre ist sollte die Löschung fehlschlagen und es sollte ein Popup mit Fehlernachricht gezeigt werden. |
|  |
| **User-Story**  Der behandele Arzt hat sich eingeloggt, um sich die Patientendaten angucken zu können. Danach will er das Behandlungszimmer verlassen. Da der Patient nicht in der Lage sein soll, alle Daten ein zu sehen, sperrt der Arzt das Programm. Dadurch sind die Daten nicht mehr einsehbar. |
| **Akzeptanzkriterien**  A\_1: Wir erstellen die Einträge in der Datenbank per Hand, es wird nicht möglich sein, in der Software selbst Benutzer zu erstellen.  A\_2: Die Software startet nicht im Hauptfenster, sondern im Login Dialog.  A\_3: Es soll möglich sein, sich mit einem Benutzernamen und Passwort an zu melden.  A\_4: Es soll nicht möglich sein, sich mit einem Benutzer an zu melden, der kein Passwort hat.  A\_5: Man kann sich ausloggen.  A\_6: Man kann die Applikation sperren, wodurch man wieder das Passwort angeben muss, um Zugriff zu bekommen.  A\_7: Es sollte nicht möglich sein, sich mit falschen Daten anmelden zu können. |
| **Tasks**  T\_1: Wir brauchen eine neue Datenbanktabelle: „LOGIN\_USER“, in der die Benutzernamen und Passwörter gespeichert werden (Verschlüsselt).  T\_2: Erstellen von neuen DAO Objekten für Benutzer (BenutzerDAO, Benutzer).  T\_3: Wir müssen zwei neue Button in dem Hauptfenster "MainWindow" einfügen. Durch diese soll es möglich sein, das Fenster zu sperren oder sich aus zu loggen.  T\_4: Erstellen eines Login Dialoges, in dem man sich über einen Benutzernamen und Passwort einloggen kann (dlgLogin).  T\_5: Es sollte beim Start nicht mehr das Fenster "MainWindow" geöffnet werden, sondern in dem neuen dlgLogin Dialog.  T\_6: In dem Dialog brauchen wir zwei Felder für den Benutzernamen und Passwort. Dazu brauchen wir auch noch zwei Button für Abbrechen (schließen) und OK, um sich ein zu loggen. |
| **Testfälle**  **TF1\_:** Start mit Dialog  - Vorbedingung: Der Benutzer startet das Programm.  - auszuführende Testschritte:  - erwartetes Ergebnis: Das Programm startet mit dem Dialog.  **TF2\_:** Einloggen richtig  - Vorbedingung: Der Benutzer hat die Software gestartet und befindet sich im Login Dialog.  - auszuführende Testschritte: Der Benutzer gib sein Benutzernamen und Passwort an und drückt auf OK.  - erwartetes Ergebnis: Der Benutzer sollte jetzt eingelogt werden.  **TF3\_:** Einloggen falsch  - Vorbedingung: Der Benutzer hat die Software gestartet und befindet sich im Login Dialog.  - auszuführende Testschritte: Der Benutzer gibt falsche Daten bei dem Login an. (Ich werde hier nicht jeden einzelnen Fall beschreiben)  - erwartetes Ergebnis: Der Benutzer wird nicht eingelogt.  **TF4\_:** Program sperren  - Vorbedingung: Der Benutzer befindet sich in dem Hauptfenster und ist eingelogt  - auszuführende Testschritte: Der Benutzer klicket auf den Sperren Button.  - erwartetes Ergebnis: Das Haupt Fenster sollte jetzt gespeert werden und der Inhalt sollte nicht mehr lesbar sein. |
|  |
| **User-Story**  Als Heimleiter möchte ich einen eignen Bereich haben, wo meine Angestellten Pfleger angezeigt werden, damit ich sie einfach verwalten kann. |
| **Akzeptanzkriterien**  A\_1: Auf der Navigationsleiste sollte ein Button(Pfleger/in) geben.  A\_2: beim Drücken auf dem Button landet man auf der PflegerDaten Seite.Dort stehen die Daten vom Pfleger(PflegerDaten) in tabellarische Reihenfolge.  A\_3: unten auf der PflegerDaten Seite sollte auch Felder geben, wo ein Pfleger seine Daten eintragen kann. Das soll genau so aussehen wie beim Hinzufügen einer neuen Behandlung.  A\_3: auf der Seite PflegerDaten stehen die folgende Daten vom Pfleger(( ID, Nach- und Vornamen sowie Telefonnummer))  A\_4: Beim Anlegen einer neuen Behandlung soll der entsprechende Pfleger mit Hilfe einer Combobox ausgewählt werden können.  A\_5: Beim Klicken auf die Behandlung von einem Patient, sollten auch die Daten von dem Pfleger gezeigt werden, der die Behandlung durchgeführt hat. |
| **Tasks**  T\_1: Nurse Klasse erstellen: abgeleitet von Person, Konstruktoren mit und ohne NurseID  T\_2: NurseDAO erstellen mit Methode zum selektieren über NID.  T\_3: Datenbanktabelle NURSE anlgegen, die den Primärschlüssel NID, Nach- und Vornamen, sowie die Telefonnummern enthält des jeweiligen Pflegers enthält.  T\_4: Datenbanktabelle TREATMENTS um Fremdschlüssel NID erweitern  T\_5: TreatmentDAO SQL-befehle um NID erweitern.  T\_6: AllNursesView für Pfleger/in erstellen mit TableView der die Daten der Pfleger aus A\_4 enthält, sowie Textfelder zu allen diesen und einen Knopf Hinzufügen und einem Knopf Löschen.  T\_7: AllNursesController zu AllNursesView erstellen.  T\_6: in MainWindowView sollte noch ein Button sein mit dem Title Pfleger/in geben.  T\_7: Beim Drücken auf dem Button Pfleger landet man auf der Seite Pfleger/in.  T\_8: In der Datei NewTreatmentView sollte eine neue Combobox für den Pfleger hinzugefügt werden.  T\_9: NewTreatmentController erweitern, um das hinzufügen des Pflegers über die Combobox.  T\_10: AllTreatmentsView erweitern um die Daten des zuständigen Pflegers mit der Behandlung zusammen anzuzeigen im Format Nachname, Vorname in einem Label, sowie Telefon in einem eigenenm Label.  T\_11: AllTreatmentsController erweitern um Funktionalität um Pflegerdaten aus NURSE-Tabelle über Verbindung NID zu lesen und in das View zu laden |
| **Testfälle**  **TF1\_:** Alle Pfleger anzeigen  - **Vorbedingung**: Pfleger sein(angemeldet)  - **auszuführende Testschritte**: ein Pfleger anlegen  1- Das Button Pfleger drücken  2- auf der Seite PflegerDaten landen  3- unten in den Feldern die Daten eingeben.  - **erwartetes Ergebnis**: Die Daten der Pfleger sollten gezeigt werden.  Die Daten Pfleger sollte im DB gespeichert werden.  **TF2\_:** Details einer Behandlung  - **Vorbedingung**: Pfleger  - **auszuführende Testschritte**: Dem entsprechenden Pfleger in Combobox auswählen  - **erwartetes Ergebnis**: Pfleger sollte auch bei den Details der Behandlung gezeigt werden |
|  |
| User-Story  Der Administrator der Klinik will die Funktion der Datenbank testen. Hierbei gibt er sich die Patientendaten per Sql Befehl aus. Hierbei sind die Daten, die er durch sein Programm erhält, verschlüsselt und er hat damit keine Einsicht. |
| Akzeptanzkriterien  Die Daten, die sich in der Datenbank befinden, sind verschlüsselt und können nur durch die Software eingelesen werden. |
| Tasks  T\_1: Logik einfügen, um String verschlüsseln zu können. Mit festem Schlüssel.  T\_2: Füge zu jedem zu verschlüsselnden Parameter der Models eine eigene Verschlüsselung Repräsentation an. Dies sind immer Strings und enthalten immer nur den verschlüsselten Wert.  T\_3: Verschlüsselungslogik in die Models einfügen (aus T1).  T\_4: Beim Abfragen der Daten werden diese immer erst entschlüsselt und dann zurückgegeben.  T\_5: Neue Logik in allen DAO Objekten einfügen. Damit die Sql Abfragen den verschlüsselten Wert verwenden und nicht den Klartext. |
| Testfälle  TF1\_: Sql Abfrage von der Datenbank  - Vorbedingung: Öffnen vom Programm (nicht unseres), um Sql Abfragen zu machen.  - auszuführende Testschritte: Ausführen von Sql abfrage von den Patientendaten.  - erwartetes Ergebnis: Die Daten sollten jetzt verschlüsselt zurückgegeben werden. |
|  |
| **User-Story**  Der behandelnde Arzt des Patienten möchte die Patientendaten als PDF (weil sein Patient die haben will, weil er diese nicht per Hand für seinen Patienten rausarbeiten will. |
| **Akzeptanzkriterien**  A\_1: Der Arzt kann aus der Patientenübersicht eine PDF anfordern, die die Patientenakte des Ausgewählten Patienten enthält.  A\_2: Die Patientenakte besteht aus allen Behandlungen die an dem Patienten durchgeführt wurden sind,  wer die Behandlungen ausgeführt hat, seinem Nachnamen, Vornamen, Geburtstag, Pflegegrad und Raum |
| **Tasks**  T\_1: Knopf „Patientenakte“ in View AllPatientView hinzufügen T\_2: Klasse PatientPDFCreator in utils erstellen, die Funktionalität zum erstellen der Patientenakte als PDF bereitstellt und in dieser PDF die Daten aus den DB-Tabellen der Patienten, Behandlungen und Pfelgekräfte, die die Behandlungenen durchgeführt haben, über eine übergebene PatientenID zusammenfässt. Die Patientendaten werden über DAOimp.read() gelesen, die Behandlungen über treatmentDAO.readTreatmentsByPid() und aus diesen Behandlungen muss noch der jeweilige Pfleger bestimmt werden.  T\_3: PatientenController: Methoden hinzufügen für das Abspeichern der Patientenakte über den neuen Knopf mit Hilfe des PatientPDFCreators |
| **Testfälle**  **TF1\_: Patientenakte erstellen**  - **Vorbedingung**: Patient wurde mit einigen Behandlungen erstellt.  Tab Patient/-innen ist geöffnet  - **auszuführende Testschritte**: Auf den Knopf Patientenakte klicken und einen Speicherort über den FileDialog auswählen.  Die gespeicherte Datei öffnen.  - **erwartetes Ergebnis**: In der Patientenakte stehen die in der Vorbereitung erstellten Daten des ausgewählten Patienten |
|  |
| User-Story:  Als Haupt-Arzt möchte ich bei der Löschung von Patienten kaskadierend auch alle Behandlungen vom Patient löschen. |
| **Akzeptanzkriterien:**  A\_1: Wenn ein Patient gelöscht wird sollen auch alle zugehörige Behandlung gelöscht werden. |
| **Tasks:**  T\_1: Die schon vorhandene Logik beheben, sodass das kaskadierende Löschen funktioniert. |
| **Testfälle:**  TF1\_:  - Vorbedingung: ein muss ein Patient mit Behandlungen vorhanden sein.  - auszuführende Testschritte:  den Patient löschen.  - erwartetes Ergebnis: Alle Behandlungen vom Patient werden mit gelöscht. |