Learnables

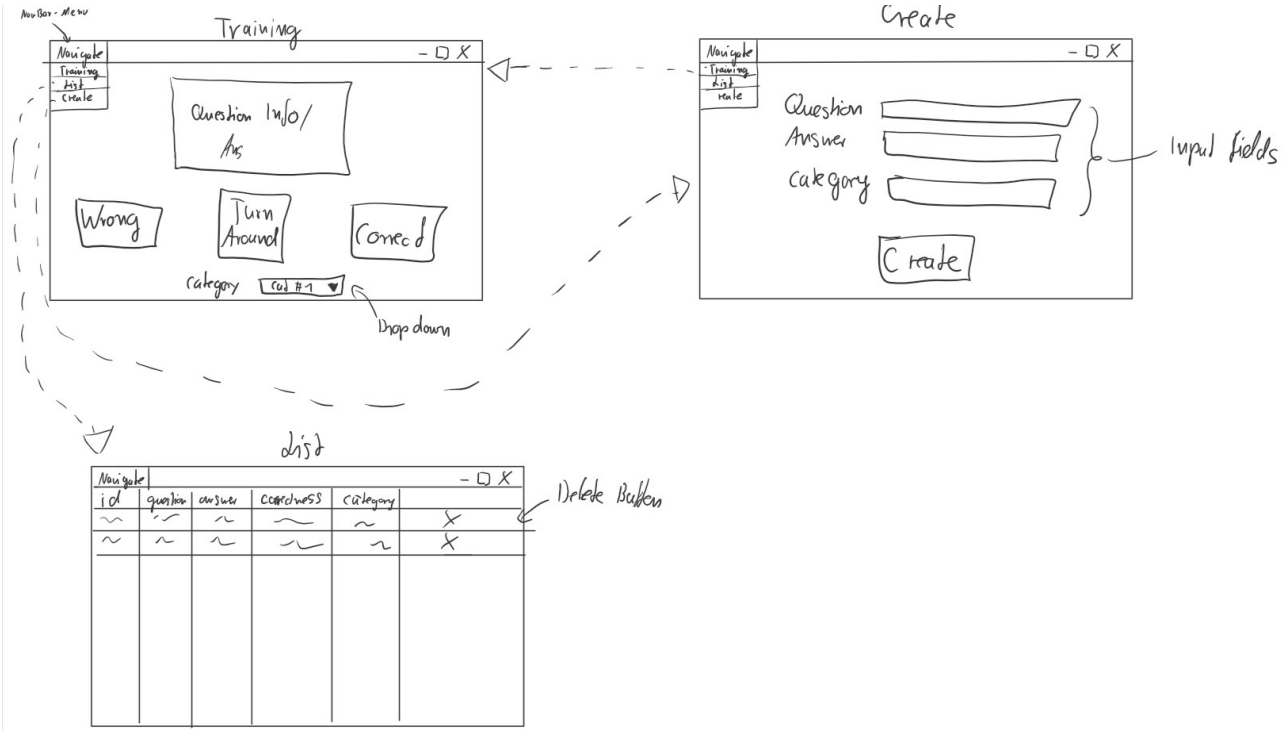
Programmier-Praktikum

von: Jesse Strathmann
Abgabe: 25.03.2021

Inhaltsverzeichnis

1. Projektplanung

1.1 Wireframe



Die Applikation soll aus drei Views bestehen: Training, List und Create.

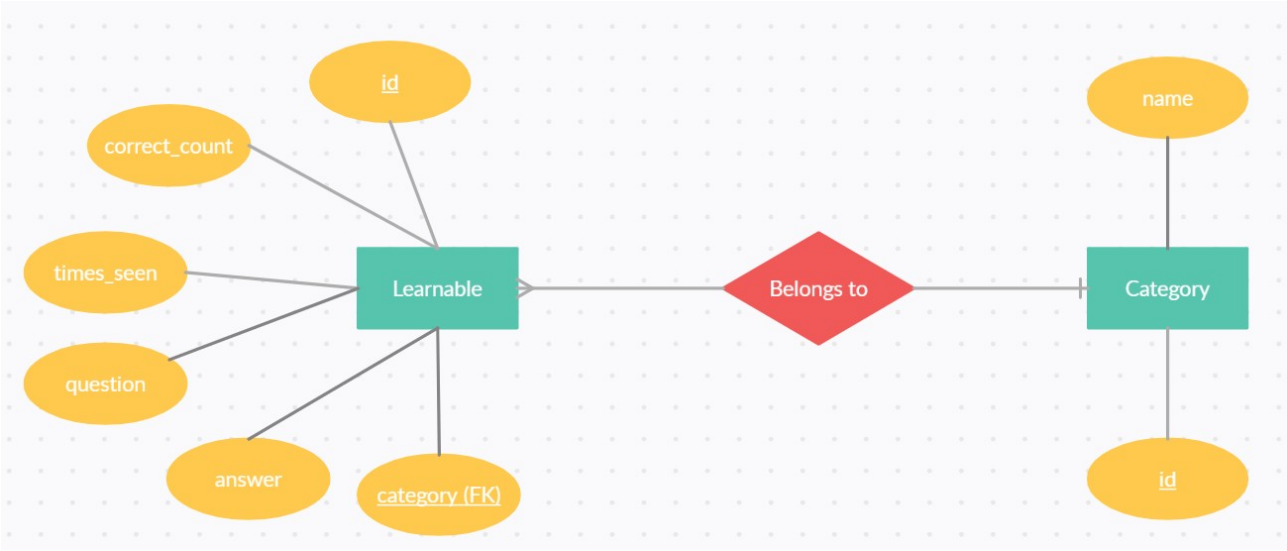
Im Training-View werden einzelne Fragen aus der jeweiligen Kategorie zufällig ausgewählt und im „Question Info / Answer“-Feld angezeigt. Wenn es sich anbietet, sollen dort auch Informationen wie die ID der Frage oder die Anzahl von „Richtig“ und „Falsch“ beim Lernen. Die Knöpfe „Wrong“ und „Correct“ dienen zur Selbstüberprüfung und sollen jeweils gedrückt werden, wenn man die Antwort wusste oder nicht. Um die Antwort auf die Frage sehen zu können, muss der „Turn Around“ Knopf gedrückt werden, welcher die Antwort im „Question Info / Answer“-Feld anzeigt. Die Kategorie aus der die Fragen ausgewählt werden sollen kann über das Dropdown-Menü bestimmt werden. Dort sollen alle Kategorien angezeigt werden, denen Fragen zugeordnet sind. Über den „Navigate“-Menüpunkt in der Menüleiste kann zu den jeweils anderen Views gewechselt

werden.

Im Create-View sollen neue Learnables angelegt werden können. Hierzu gibt es drei Eingabefelder, eins für die Frage, eins für die Antwort und eins für die Kategorie. Per Druck auf den „Create“-Knopf soll ein neuer Datensatz in der Datenbank erzeugt werden und zum Lernen verfügbar stehen.

Im List-View sollen alle Learnables aufgelistet werden. Hierzu soll eine Tabelle dienen, wo in jeder Zeile die jeweiligen Attribute des Learnables aufgezeigt werden und in der letzten Spalte ein Knopf zur Löschung des Learnables ist. Werden alle Learnables einer bestimmten Kategorie gelöscht, soll diese Kategorie automatisch gelöscht werden.

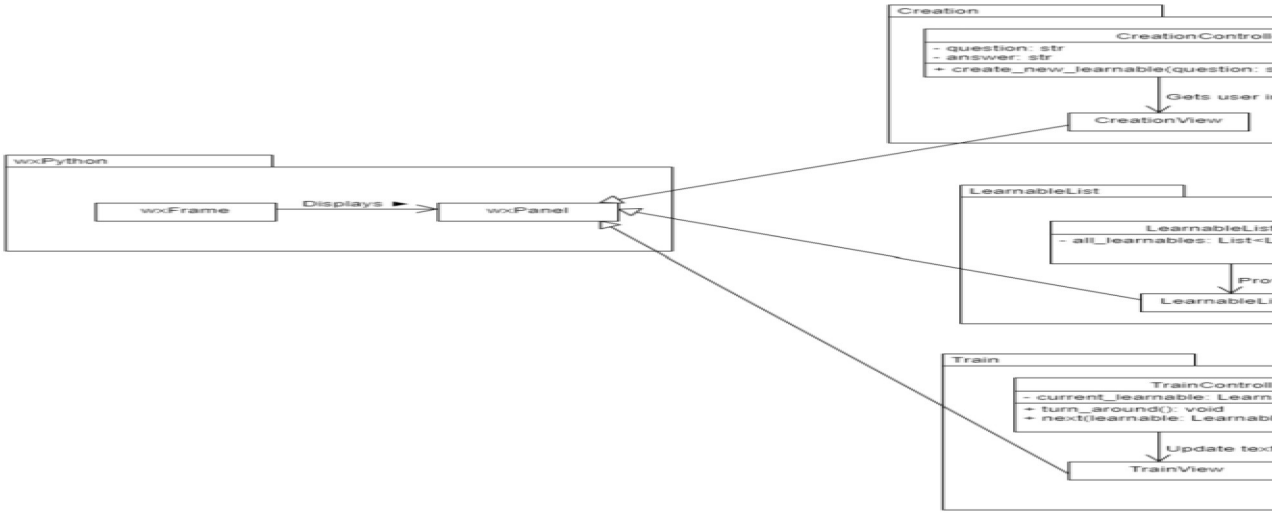
1.2 Entity-Relationship-Modell



Das Entity-Relationship-Modell besteht aus zwei Entitäten und einer Beziehung zwischen diesen Entitäten. Die Entität „Learnable“ enthält dabei Frage, Antwort, die Anzahl der gesehenen Male, die Anzahl der richtigen Antworten, eine ID und einen Fremdschlüssel zu einer Kategorie, in welche das Learnable gehört. Die Entität „Category“ besteht aus dem Namen der Kategorie und einer ID.

Die Beziehung zwischen diesen beiden Entitäten ist eine 1:n-Beziehung, da eine Kategorie zu mehreren Learnables gehören kann, aber ein Learnable in diesem Fall nur zu jeweils einer Kategorie. Die Beschreibung dieser Beziehung „Belongs to“, sagt aus, dass ein Learnable einer Kategorie „angehört“.

1.3 Klassendiagramm



(Dieses Projekt wird in Python mit Hilfe der wxPython-Bibliothek realisiert)

Das Programm folgt dem MVC-Pattern und ist in vier Module aufgeteilt: Creation, LearnableList, Train und Database.

Creation, LearnableList und Train sind hierbei die Module, welche die Views aus dem Wireframe abbilden. Innerhalb dieser Module ist jeweils eine Controller-Klasse und eine View-Klasse. Die View-Klasse erbt von der „wxPanel“-Klasse aus der wxPython-Bibliothek, welche es ermöglicht, dass die View-Klasse GUI-Elemente anzeigen kann und auch selbst innerhalb des Fensters (wxFrame) angezeigt werden kann. Der Controller dient als Schnittstelle zwischen View und Model und erbt von der abstrakten Superklasse „ViewController“.

Der „ViewController“ hat zwei „protected“ Attribute namens „learnbale_manager“ und „category_manager“, welche eine Instanz der jeweiligen Manager aus dem Datenbank Modul speichern und den Controllern aus den View-Modulen zur Verfügung stellen.

Die beiden Manager-Klassen aus dem Datenbankmodul verwenden die generalisierte „DBManager“-Klasse, welche die Datenbank initialisiert, die notwendigen Tabellen erzeugt und den Manager-Klassen das Ausführen von SQL-Anweisungen per Wrapper-Funktionen erleichtert. Außerdem sind die Manager-Klassen für die jeweilige Entität zuständig und bieten durch Modell-Klassen und Wrapper-Funktionen eine Schnittstelle zwischen Programm und Datenbank.

1.4 Testfalltabelle

Testfallbeschreibung	Testwert	Hypothese	Ergebnis
DBManager - init_database	Datenbankdatei nicht vorhanden	Datenbankdatei wird erzeugt und Datenbank mit notwendigen Tabellen befüllt	Hypothese erfüllt
DBManager - init_database	Datenbankdatei vorhanden	Datenbankdatei wird nicht neu erzeugt und die Tabellen werden nicht erneut versucht zu erzeugen und befüllt	Hypothese erfüllt
LearnableManager - Wrapper-Funktionen	DBManager vorhanden	Alle Wrapper-Funktionen operieren erfolgreich	Hypothese erfüllt
CategoryManager - Wrapper-Funktionen	DBManager vorhanden	Alle Wrapper-Funktionen operieren erfolgreich	Hypothese erfüllt
Alle Controller-Klassen haben die selben Instanzen der Manager-Klassen	Vererbte Attribute des ViewControllers	Hashwerte der Instanzen sind in jedem Controller identisch	Hypothese erfüllt
LearnableListView zeigt alle vorhandenen Datensätze an	Keine Datensätze vorhanden	Es werden nur die Tabellenköpfe angezeigt	Hypothese erfüllt
LearnableListView zeigt alle vorhandenen Datensätze an	Datensätze vorhanden	Alle vorhandenen Datensätze werden in der Tabelle angezeigt	Hypothese erfüllt
LearnableListView ermöglicht das Löschen eines Datensatzes	Drücken des Löschknopfes	Datensatz verschwindet aus der Tabelle und aus der Datenbank	Hypothese erfüllt
LearnableListView löscht Kategorie automatisch, wenn die letzte zugehörige Frage gelöscht wurde	Drücken des Löschknopfes	Datensatz verschwindet aus der Tabelle und aus der Datenbank + Kategorie wird aus der Datenbank gelöscht	Hypothese erfüllt
CreationView ermöglicht das Erstellen neuer Datensätze	Kein Eingabefeld leer	Erzeugt den Datensatz und die zugehörige Kategorie falls nicht vorhanden	Hypothese erfüllt
CreationView ermöglicht das	Ein Eingabefeld leer	Erzeugung des Datensatzes kann nicht	Hypothese erfüllt

Erstellen neuer Datensätze		erfolgen	
CreationView ermöglicht das Erstellen neuer Datensätze	Mehr als ein Eingabefeld leer	Erzeugung des Datensatzes kann nicht erfolgen	Hypothese erfüllt
TrainView zeigt nur Fragen der jeweiligen Kategorie an	Zwei Kategorien mit verschiedenen Fragen vorhanden	Nur Fragen der jeweilig ausgewählten Kategorie werden angezeigt	Hypothese erfüllt
(TrainView) Daten der Modell-Instanzen werden korrekt aktualisiert	Druck des „Correct“-Knopfes im TrainView	Anzahl der gesehenen Male und Anzahl der richtigen Beantwortungen werden um 1 erhöht	Hypothese erfüllt
(TrainView) Daten der Modell-Instanzen werden korrekt aktualisiert	Druck des „Wrong“-Knopfes im TrainView	Anzahl der gesehenen Male um 1 erhöht	Hypothese erfüllt
(TrainView) Daten der Modell-Instanzen werden korrekt aktualisiert	Druck des „Turn Around“- Knopfes im TrainView	Keine Änderungen werden vorgenommen	Hypothese erfüllt
(TrainView) Es kann zwischen Frage und Antwort gewechselt werden	Die Frage wird angezeigt und der des „Turn Around“-Knopf gedrückt	Es wird die Antwort angezeigt	Hypothese erfüllt
(TrainView) Es kann zwischen Frage und Antwort gewechselt werden	Die Antwort wird angezeigt und der des „Turn Around“-Knopf gedrückt	Es wird die Frage angezeigt	Hypothese erfüllt
(TrainView) Leere Datenbank	Keine Learnables in der Datenbank	Statt Frage bzw. Antwort wird ein Hinweis angezeigt	Hypothese erfüllt
(TrainView) Leere Datenbank und Druck der Knöpfe	Keine Learnables in der Datenbank	Es wird keine Aktion durchgeführt	Hypothese erfüllt
(TrainView) „Any“-Kategorie	Mehr als ein Learnable in der Datenbank	Learnables werden unabhängig von ihrer Kategorie zufällig ausgewählt	Hypothese erfüllt
Wechsel in den	Letzter Datensatz	Statt Frage bzw.	Es wird weiterhin das

TrainView nach Löschung des letzten Datensatzes	wurde im ListView gelöscht	Antwort wird ein Hinweis angezeigt	letztes Learnable angezeigt bis zum Druck auf entweder „Correct“ oder „Wrong“. Das im Controller gespeicherte Learnable wird bei der Löschung des letzten Datensatzes nicht automatisch aktualisert. Dies passiert erst bei Druck auf „Correct“ oder „Wrong“.

2. Abweichungen und Mängel

2.1 Abweichungen: Wireframe

Das Wireframe aus der Planung wurde weitgehend eingehalten. Die Navigationsleiste ist wie in der Planung über eine Menüleiste realisiert worden, es gibt die geplanten drei Views und diese beinhalten die geplante Funktionalität. Die Layouts der jeweiligen Views entsprechen jedoch nicht exakt dem der Wireframes. So ist beispielsweise der Löschknopf im ListView für einzelne Datensätze über einen Rechtsklick im Kontextmenü zu finden. Das „Question Info / Answer“-Feld im Wireframe wurde in der Planung absichtlich nicht ausführlich dargestellt, um während des Entwicklungsprozesses mehr Spielraum für den dort angezeigten Text zu haben.

Bis auf wenige Kleinigkeiten wurde die Planung des Wireframes also eingehalten.

2.2 Abweichungen: Entity-Relationship-Modell

Vom Entity-Relationship-Modell wurde nicht abgewichen.

2.3 Abweichungen: Klassendiagramm

Grundlegend wurde das Klassendiagramm aus der Planung eingehalten. Die verschiedenen Module wurden entsprechend umgesetzt und die geplante Funktionalität und das geplante Zusammenspiel einzelner Klassen erfolgt wie geplant. Es wurden allerdings einige Erweiterungen und neue Attribute ergänzt. So wurden den Controllern beispielsweise Attribute für die jeweiligen GUI-Elemente des Views hinzugefügt, was in der Planung so noch nicht präzise möglich war. Hinzu kommen weitere interne Methoden wie Listener, die ebenfalls in der Planung nicht präzise abzusehen waren. Auch wurden im DBManager Contract-Klassen hinzugefügt um die

Spaltennamen der Tabellen besser verwalten zu können und Abfragen besser strukturieren zu können. Die „ViewController“-Klasse wurde außerdem in ein separates Modul ausgelagert, damit es in anderen Modulen verwendet werden konnte.

2.4 Mängel

Das Programm entspricht den Anforderungen aus der Aufgabe und ist benutzbar. Ein subjektiver Mangel kann das Layout bieten. Der CreateView beispielsweise ist funktional, aber optisch ausbaufähig. Auch der ListView ist funktional, aber die Aufteilung der Spalten in responsive Größen fehlt. Ebenfalls könnte das Kategorie-Dropdown im TrainingView besser in das Layout integriert werden durch eine andere Positionierung oder Größe in Relation zu den drei Bedienknöpfen.

Ein objektiver Mangel hingegen ist jedoch, dass beim Löschen des letzten Datensatzes und dem folgenden Wechsel zurück in den TrainingView noch immer das vorherige Learnable angezeigt wird und erst bei Druck auf „Correct“ oder „Wrong“ verschwindet. Wie in der Testfalltabelle beschrieben, sollte direkt bei dem Wechsel zurück in den TrainingView der Informationstext anstelle des letzten Learnables angezeigt werden.