VibroStudies

Dokumentation des Entwurfs

Anne-Kathrin Hermann
Tizian Bitschi
Adrian Furrer
Anh Kha Nguyen
Enes Sayin

Sommersemester 2020

Versionslog

Datum	Abschnitt: Beitrag	Bemerkung
29.06.2020	Version 1.0 veröffentlicht	

Inhaltsverzeichnis

1 Codestruktur

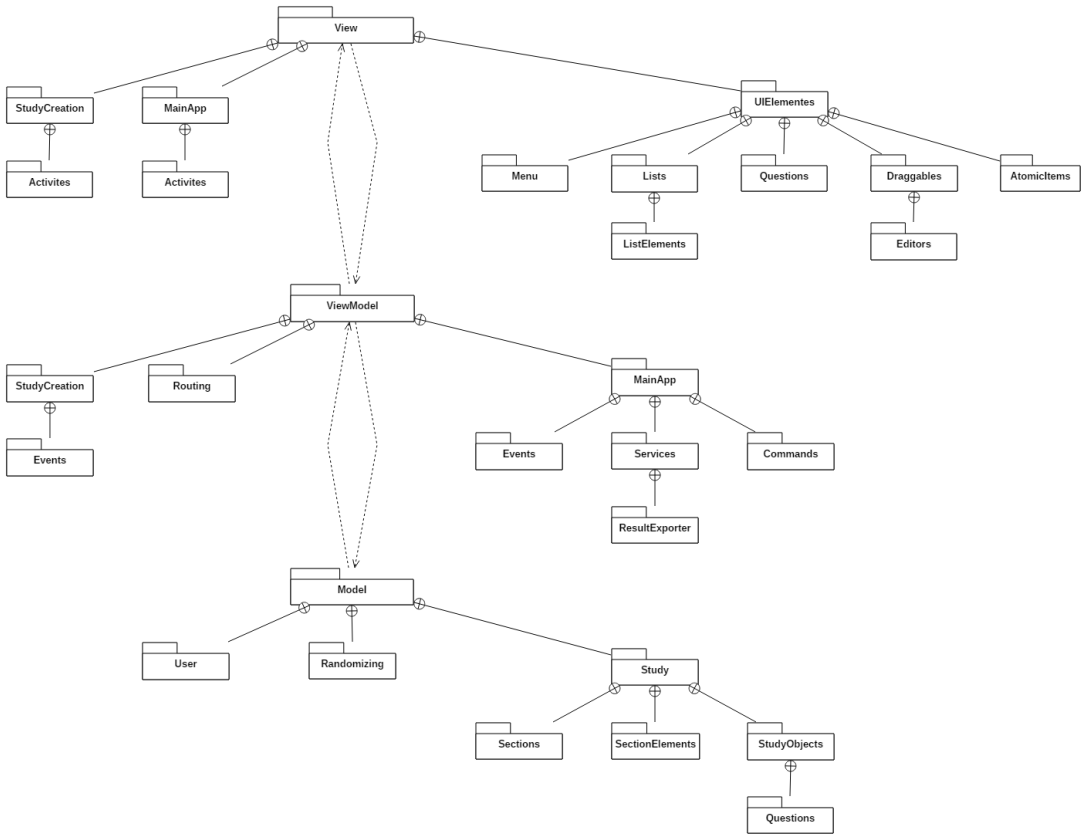

Abbildung 1.0.1: Struktur der ganzen App als Pakete erfasst

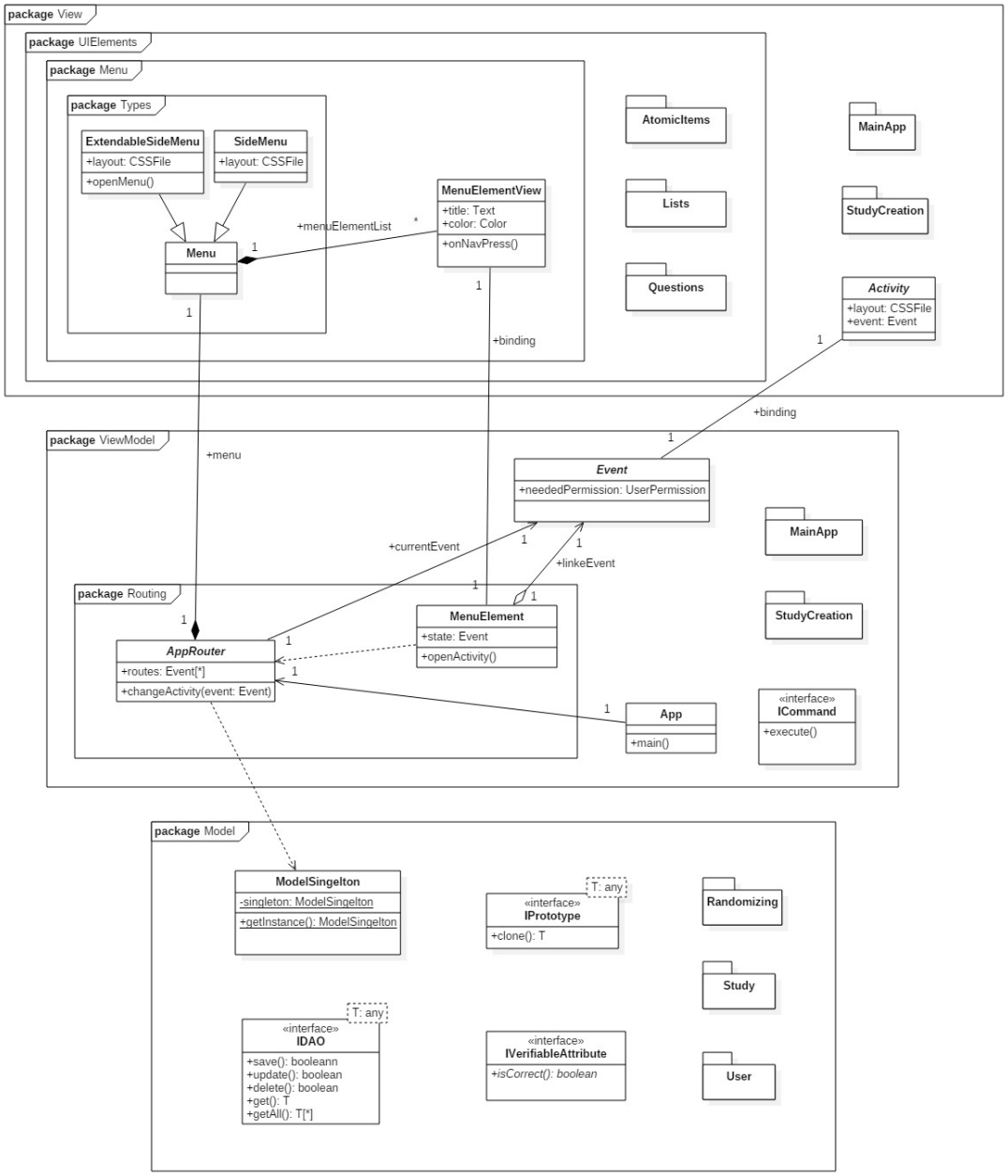

Abbildung 1.0.2: Main

1.1 View

Die View ist für die Benutzeroberfläche der Software zuständig.

View.Activity
Die public abstract class **Activity** gibt vor, wie die einzelnen Activites auszusehen haben.

public **layout** : cssFile. Hier wird das Layout der Ansicht festgelegt.

public **binding** : Event. Dadurch wird die Funktionalität der Ansicht sichergestellt.

1.1.1 Paket View.StudyCreation

Das Paket enhält alle Ansichten, die zur Erstellung der Studie gebraucht werden.

Paket View.StudyCreation.Activities
Dieses Package enthält Klassen, welche Aktivitäten darstellen, die die Erstellung und Bearbeitung einer Studie ermöglichen. Die Klassen erstellen hierzu eine Benutzeroberfläche/Ansicht für die jeweiligen Aktivitäten.

View.StudyCreation.Activities.GeneralEditingActivity extends Activity
Die Klasse PUBLIC CLASS **GeneralEditingActivity** ermöglicht die Bearbeitung der allgemeinen Einstellungen einer Studie im Reiter „Allgemein“.

public **inputList** : any[*] enthält die Elemente der Activity.

public void **onStateChange()** gibt weiter, wenn ein Element angeklickt wird.

View.StudyCreation.Activities.IntroductionEditingActivity extends Activity
Die Klasse PUBLIC CLASS **IntroductionEditingActivity** ermöglicht die Bearbeitung der Einführung einer Studie im Reiter „Einführung“.

public **inputList** : any[*] enthält die Elemente der Activity.

public void **onStateChange()** gibt weiter, wenn ein Element angeklickt wird.

public void **onDraggableInsert()** Diese Methode wird aufgerufen, wenn ein Umfrage-Objekt aus der Toolbox heraus- und in eine Target-Area hineingezogen wird. Hierbei erstellt das Model dasselbe Umfrage-Objekt speziell für das Target-Area, welches in der View auch angezeigt wird.

View.StudyCreation.Activities.SectionEditingActivity extends Activity, implements Editing-Activity

Die Klasse PUBLIC CLASS **SectionEditingActivity** ermöglicht die Bearbeitung der Studiengliederung einer Studie im Reiter „Studiengliederung".

public **inputList** : any[*] enthält die Elemente der Activity.

public void **onStateChange()** gibt weiter, wenn ein Element angeklickt wird.

public **onSectionSelect()** zeigt die entsprechenden Inhalte an.

public **onDraggableInsert()**: void Diese Methode wird aufgerufen, wenn ein Umfrage-Objekt aus der Toolbox heraus- und in eine Target-Area hineingezogen wird. Hierbei erstellt das Model dasselbe Umfrage-Objekt speziell für das Target-Area, welches in der View auch angezeigt wird.

View.StudyCreation.Activities.IEditingActivity

Die Klasse PUBLIC INTERFACE **IEditingActivity** ist eine Schnittstelle, die 3 Methoden zur Verfügung stellt. Sie wird von Klassen implementiert, die Umfrage-Objekte erstellen.

void public **onCreatePress()**: void
Erstellt ein Umfrage-Objekt der jeweiligen Klasse.

void public **onDeletePress()**: void
Entfernt ein Umfrage-Objekt der jeweiligen Klasse.

void public **onStateChange()**: void
Speichert und zeigt die aktualisierte Version des Umfrage-Objekts der jeweiligen Klasse im View an.

View.StudyCreation.Activities.TextBlockEditingActivity extends Activity implements IEditingActivity

Die Klasse PUBLIC CLASS **TextBlockEditingActivity** ermöglicht die Erstellung und Bearbeitung von Textblock-Objekten.

public **inputList** : any[*] enthält die Elemente der Activity.

public void **onStateChange()** gibt weiter, wenn ein Element angeklickt wird.

View.StudyCreation.Activities.QuestionEditingActivity extends Activity implements IEditingActivity

Die Klasse PUBLIC CLASS **QuestionEditingActivity** ermöglicht die Erstellung und Bearbeitung von Frage-Objekten mit unterschiedlichen Fragetypen.

public **inputList** : any[*] enthält die Elemente der Activity.

public void **onStateChange()** gibt weiter, wenn ein Element angeklickt wird.

View.StudyCreation.Activities.VibrationEditingCreationActivity extends Activity implements IEditingActivity

Die Klasse PUBLIC CLASS **VibrationEditingCreationActivity** ermöglicht die Erstellung und Bearbeitung von Vibrationsmuster-Objekten.

public **inputList** : any[*] enthält die Elemente der Activity.

public void **onStateChange()** gibt weiter, wenn ein Element angeklickt wird.

public **onDraggableInsert()**: void
Diese Methode wird aufgerufen, wenn ein Umfrage-Objekt aus der Toolbox heraus- und in eine Target-Area hineingezogen wird. Hierbei erstellt das Model dasselbe Umfrage-Objekt speziell für das Target-Area, welches in der View auch angezeigt wird.

View.StudyCreation.Activities.PlaygroundEditingActivity extends Activity implements IEditingActivity

Die Klasse PUBLIC CLASS **PlaygroundEditingActivity** ermöglicht die Erstellung von Playground-Objekten.

public **inputList** : any[*] enthält die Elemente der Activity.

public void **onStateChange()** gibt weiter, wenn ein Element angeklickt wird.

public **onDraggableInsert()**: void

Diese Methode wird aufgerufen, wenn ein Umfrage-Objekt aus der Toolbox heraus- und in eine Target-Area hineingezogen wird. Hierbei erstellt das Model dasselbe Umfrage-Objekt speziell für das Target-Area, welches in der View auch angezeigt wird.

View.StudyCreation.Activities.TestEditingActivity extends Activity implements IEditingActivity

Die Klasse PUBLIC CLASS **TestEditingActivity** ermöglicht die Erstellung von Test-Objekten.

public **inputList** : any[*] enthält die Elemente der Activity.

public void **onStateChange()** gibt weiter, wenn ein Element angeklickt wird.

public **onDraggableInsert()**: void
Diese Methode wird aufgerufen, wenn ein Umfrage-Objekt aus der Toolbox heraus- und in eine Target-Area hineingezogen wird. Hierbei erstellt das Model dasselbe Umfrage-Objekt speziell für das Target-Area, welches in der View auch angezeigt wird.

1.1.2 Paket View.MainApp

Das Paket MainApp enthält alle restlichen Ansichten, die in der Software exisitieren.

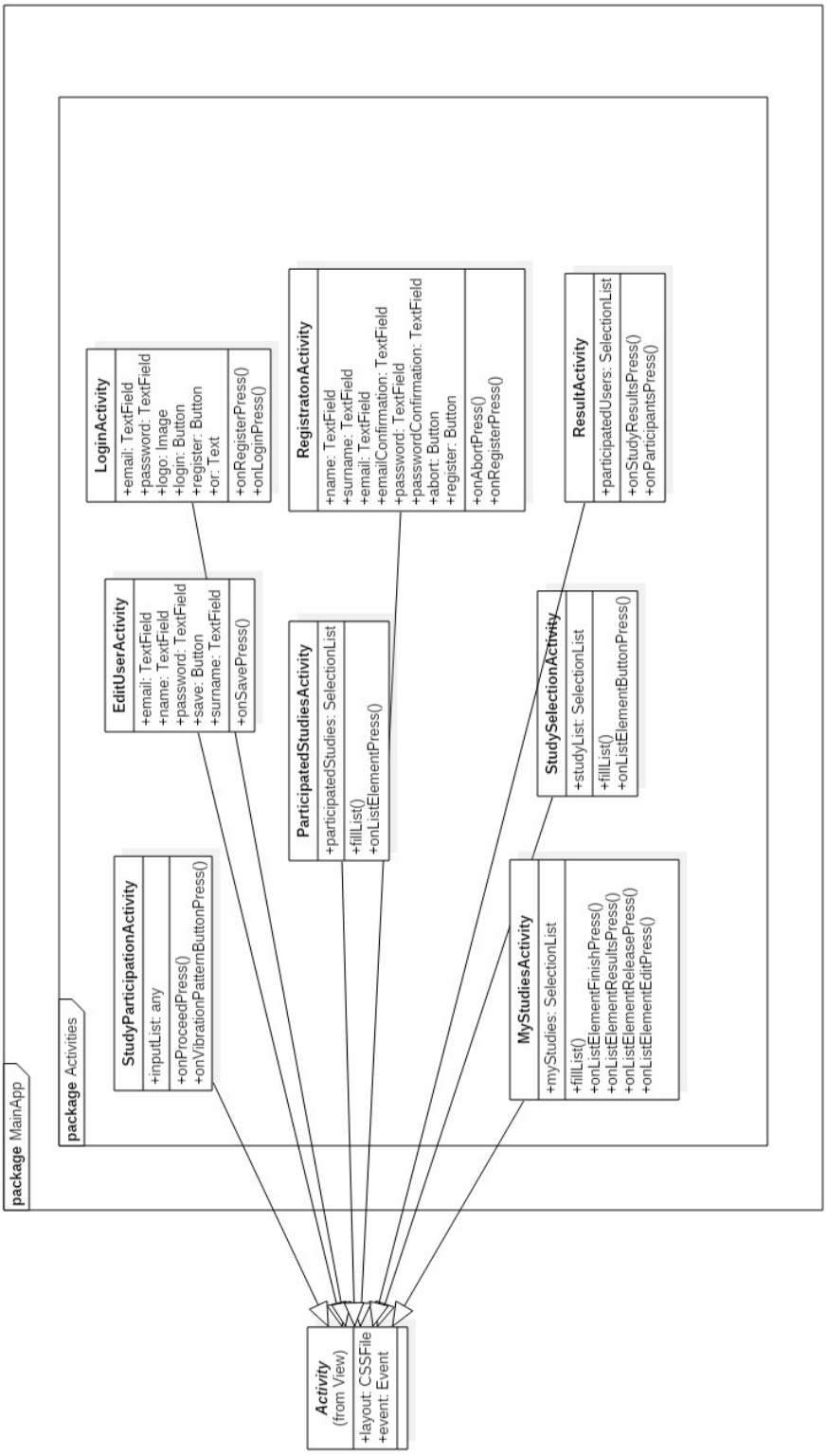

Abbildung 1.1.1: View.MainApp

View.MainApp.Activities.EditUserActivity extends Activity
Die Klasse public class **EditUserActivity** ist die Ansicht für den Benutzer, um seinen Account zu bearbeiten.

public **name** : TextField. Hier kann der Benutzer einen neuen Vornamen eingeben.

public **surname** : TextField. Hier kann der Benutzer einen neuen Nachnamen eingeben.

public **email** : TextField. Hier kann der Benutzer eine neue E-Mail Adresse eingeben.

public **password** : TextField. Hier kann der Benutzer ein neues Passwort eingeben.

public **save** : Button. Durch Drücken des Knopfs werden die Änderungen des Benutzers gespeichert.

public void **onSavePress()** gibt die eingegebenen Daten weiter.

View.MainApp.Activities.StudySelectionActivity extends Activity
Die Klasse public class StudySelectionActivity ist die Ansicht für den Benutzer, in der er eine Studie auswählen kann.

public **studyList** : SelectionList. Diese Liste zeigt die Studien an.

public void **fillList** füllt die Liste mit den Studien, die zur Auswahl stehen.

public void **onListElementButtonPress** reagiert auf das Auswählen einer Studie.

View.MainApp.Activities.LoginActivity extends Activity
Die Klasse public class **LoginAcitivity** enthält die Ansicht für den Loginscreen.

public **email** : TextField. Hier gibt der Benutzer seine E-Mail Adresse ein.

public **password** : TextField. Hier gibt der Benutzer sein Passwort ein.

public **logo** : Image. Hier wird das Logo der VibroStudies Software angezeigt.

public **login** : Button. Hier kann man versuchen, sich einzuloggen.

public **register** : Button. Hier kommt man zur Registrieren-Seite.

public **or** : Text. Dies trennt den Anmelden- und Registrierenbereich.

View.MainApp.Activities.ResultActivity extends Activity

Diese public class **ResultActivity** dient zur Anzeige des Ergebnis einer Studie.

public **participatedUsers** : SelectionList. Stellt die teilgenommenen Benutzer dar.

public void **onStudyResultsPress()** gibt weiter, dass der Benutzer eine Ergebnisliste wünscht.

public **onParticipantsPress()** gibt weiter, dass der Benutzer eine Teilnehmerliste wünscht.

View.MainApp.Activities.StudyParticipationActivity extends Activity

Die public class StudyParticipationActivity zeigt die Teilnahme an der Studie an.

public **inputList** : any[*]. Sammelt alle Ansichten aus der erstellten Studie.

public void **onProceedPress()** gibt weiter, dass der User die nächste Seite bearbeiten möchte.

public void **onVibrationPatternButtonPress()** spielt ein Vibrationsmuster ab.

View.MainApp.Activities.RegistrationActivity extends Activity

Die public class **RegistrationActivity** zeigt die Ansicht für die Registrierung.

public **name** : TextField. Hier gibt der Benutzer seinen Vornamen ein.

public **surname** : TextField. Hier gibt der Benutzer seinen Nachnamen ein.

public **email** : TextField. Hier gibt der Benutzer seine E-Mail-Adresse ein.

public **emailConfirmation** : TextField. Hier gibt der Benutzer seine E-Mail-Adresse erneut ein, um sie zu bestätigen.

public **password** : TextField. Hier gibt der Benutzer sein gewünschtes Passwort ein.

public **passwordConfirmation** : TextField. Hier gibt der Benutzer sein Passwort erneut ein, um es zu bestätigen.

public **abort** : Button. Dient zum Abbrechen der Registrierung.

public **register** : Button. Dient zum Bestätigen der Registrierung.

public void **onAbortPress()** verarbeitet das Abbrechen der Registrierung.

public void **onRegisterPress()** verarbeitet die Registrierung.

View.MainApp.Activities.ParticipatedStudiesActivity extends Activity

Diese public class **ParticipatedStudiesActivity** zeigt die Studien an, an denen der Benutzer bereits teilgenommen hat.

public **participatedStudies** : SelectionList. Sammelt die Studien, an denen der Benutzer bereits teilgenommen hat. public void **fillList()** füllt die Liste der Studien, an denen teilgenommen wurde.

public void **onListElementPress()** gibt weiter, dass ein Element aus der Liste gelöscht wurde.

View.MainApp.Activities.MyStudiesActivity extends Activity

Diese public class **MyStudiesActivity** zeigt die erstellten Studien eines Studienleiters.

public **myStudies** : SelectionList. Sammelt die erstellten Studien.

public void **fillList()** füllt die Liste der Studien, die ein Studienleiter erstellt hat.

public void **onListElementFinishPress()** gibt weiter, dass der Knopf zum Beenden gedrückt wurde.

public void **onListElementResultsPress()** gibt weiter, dass der Benutzer die Ergebnisse einsehen und/oder speichern möchte.

ah ja public void **onListElementReleasePress()** gibt weiter, dass der Studienleiter die Studie veröffentlichen will.

public void **onListElementEditPress()** gibt weiter, dass der Studienleiter seine Studie bearbeiten will.

1.1.3 Paket View.UIElements

Das Paket enthält alle Elemente, aus denen die Ansichten zusammengesetzt werden.

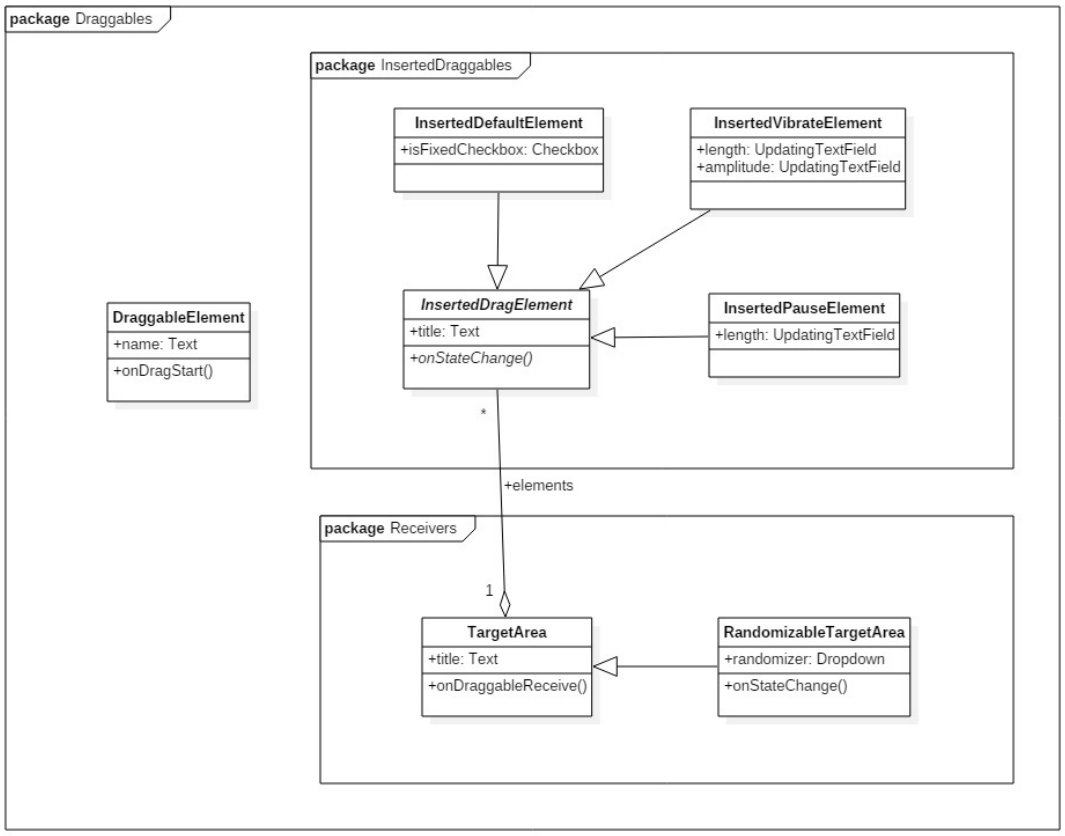

Abbildung 1.1.2: View.UIElements.Draggables

Paket View.UIElements.Draggables

In diesem Paket liegen alle Klassen, die das Drag and Drop implementieren. Sowohl das Element, das man verschieben kann, als auch das Ergebnis (Paket InsertedDraggables) und der Bereich, in den das Element gezogen wird (Paket Receivers).

View.UIElements.Draggables.DraggableElement

Die Klasse public class **DraggableElement** modelliert verschiebbare Elemente (z.B. ein Vibrationsmuster, das in einen Playground gezogen wird).

public **name** : Text ist der Name des Elements, der erscheint.

public void **onDragStart()** ermöglicht das Verschieben.

View.UIElements.Draggables.InsertedDraggables.InsertedDragElement

Die Klasse public abstract class **InsertedDragElement** ist eine Kindklasse von InsertedDragElement und legt fest, wie ein DraggableElement aussieht, nachdem es vom Benutzer in einer TargetArea abgelegt wurde.

public **title** : Text ist der Name des Elements.

public abstract void **onStateChange** verändert das Aussehen zum gewünschten, wenn das Element abgelegt wurde.

View.UIElements.Draggables.InsertedDraggables.InsertedDefaultElement extends InsertedDragElement

Die Klasse public class **InsertDefaultElement** ist eine Kindklasse von InsertedDragElement und wird bei Vibrationsmustern, Fragen, Playgrounds und Tests benutzt.

public **isFixedCheckbox** : Checkbox ist das Kästchen, das angehakt wird, wenn der Benutzer das Element fixieren möchte.

View.UIElements.Draggables.InsertedDraggables.InsertedPauseElement extends InsertedDragElement

Die Klasse public class **InsertPauseElement** ist eine Kindklasse von InsertedDragElement und erzeugt eine Instanz, die bei einer Pause in einem Vibrationsmuster verwendet wird.

public **length** : UpdatingTextField ist das Textfeld, in dem die Länge der Pause eingetragen wird.

View.UIElements.Draggables.InsertedDraggables.InsertedVibrateElement extends InsertedDragElement

Die Klasse public class **InsertVibrateElement** ist eine Kindklasse von InsertedDragElement und erzeugt eine Instanz, die bei einer Vibration in einem Vibrationsmuster verwendet wird.

public **length**: UpdatingTextField ist das Textfeld, in dem der Benutzer die Länge der Vibration eingibt.

public **amplitude**: UpdatingTextField. Hier wird die gewünschte Amplite eingetragen.

View.UIElements.Draggables.Receivers.TargetArea

Die Klasse public class **TargetArea** stellt den Bereich dar, in den DraggableElements gezogen werden können. Sie stellen also z.B. ein Vibrationsmuster da.

public **title** : Text. Hier steht der vom Studienleiter vergebene Name des Elements.

public **elements** : InsertedDragElements sind die Elemente, die in der TargetArea enthalten sind.
public void **onDraggableReceive()**
Diese Methode verarbeitet ein DraggableElement, sobald es in der TargetArea abgelegt wird.

View.UIElements.Draggables.Receivers.RandomizableTargetArea

Diese Klasse erweitert die TargetArea, um die Entscheidung mit welcher Methode randomisiert wird.

public **randomizer** : Dropdown. Hier findet sich die Auswahl der Randomisierungsmethoden

public void **onDropdownChange()**. Diese Methode verarbeitet die Auswahl des Benutzers.

Paket View.UIElements.Menu

Hier liegen alle Klassen, die die Menüs darstellen. Die verschiedenen Typen von Menüs werden in dem Paket Types zusammengefasst.

View.UIElements.Menu.MenuElementView

Die Klasse public class **MenuElementView** ist die Ansicht eines MenuElements.

public **title** : Text ist der Name des Menüelements.

public **color** : Color ist die Farbe des Menüelements.

public void **onNavPress()** gibt weiter, wenn ein Element angeklickt wird.

View.UIElements.Menu.Types.ExtendableSideMenu

Die Klasse public class **ExtendableSideMenu** ist eine Kindklasse von Menu. Diese Klasse modelliert das ausklappbare Menü in der Software.

public void **openMenu()**
Diese Methode klappt das Menü aus.

View.UIElements.Menu.Types.SideMenu

Die Klasse public class **SideMenu** ist eine Kindklasse von Menu. Diese Klasse zeigt ein seitliches Menü an.

public **layout** : cssFile beschreibt das Layout.

View.UIElements.Menu.Types.Menu

Die Klasse public class **Menu** ist dafür verantwortlich ein Menü aufzubauen.

public **menuElementList** : MenuElementView[*] enthält die Elemente des Menüs.

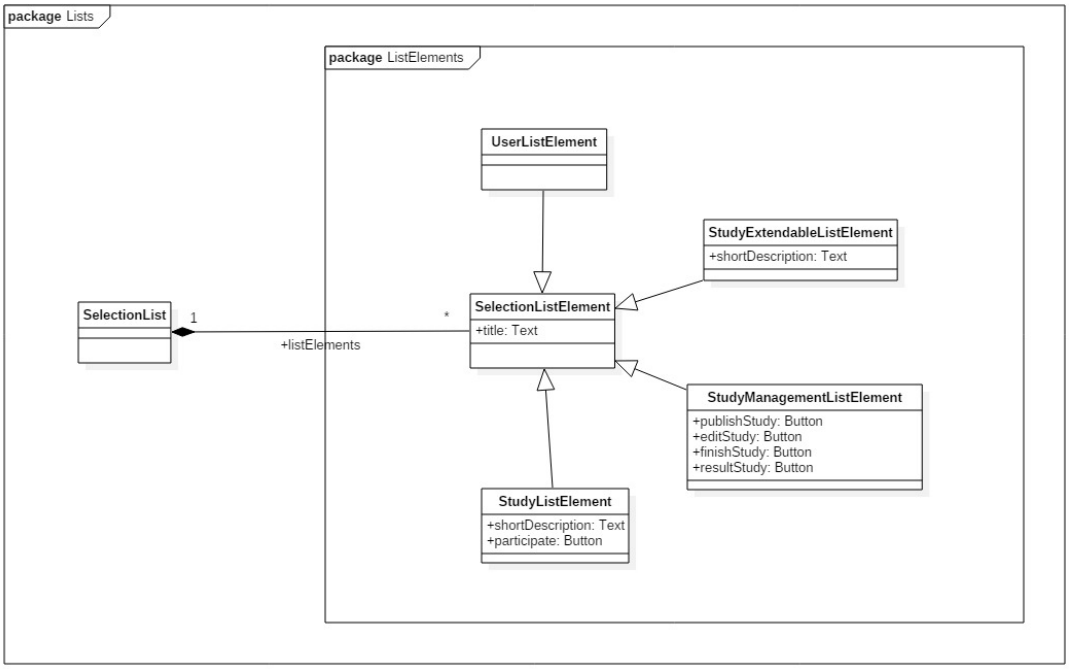

Abbildung 1.1.3: View.UIElements.Lists

View.UIElememts.Lists

Dieses Paket fasst alle Klassen zusammen, die mit Listen zusammenhängen. Im Paket ListElements liegen alle Elemente, die eine Liste enthalten kann.

View.UIElements.Lists.SelectionList

Die Klasse **SelectionList** ist eine Liste, die Elemente besitzt, mit denen interagiert werden kann.

public **listElements** : SelectionListElement[*]
Ist die Liste der anzuzeigenden Elemente.

View.UIElements.Lists.ListElements.SelectionListElement

Die Klasse **SelectionListElement** ist ein einfaches Element in einer SelectionList.

public **title** : Text
Ist der Titel des Elements.

View.UIElements.Lists.ListElements.StudyExtendableListElement extends SelectionListElement

Die Klasse **StudyExtendableListElement** repräsentiert eine Studie, an der man bereits teilgenommen hat. Der Initialzustand zeigt lediglich den Titel an und die Möglichkeit, die Kurzbeschreibung auszuklappen.

public **shortDescription** : Text
Ist die Kurzbeschreibung.

View.UIElements.Lists.ListElements.StudyListElement extends SelectionListElement

Die Klasse **StudyListElement** repräsentiert eine Studie, an der man teilnehmen kann. Der Initialzustand zeigt lediglich den Titel an und die Möglichkeit, die Kurzbeschreibung auszuklappen. Dann kann an der Studie teilgenommen werden.

public **shortDescription** : Text
Ist die Kurzbeschreibung.

public **participate** : Button
Über diesen Button kann man an der Studie teilnehmen.

View.UIElements.Lists.ListElements.StudyManagementListElement extends SelectionListElement

Die Klasse **StudyManagementListElement** repräsentiert eine selbst erstellte Studie. Hierbei gibt es Buttons, mit denen man in die Verwaltung und die Editierung der Studie kommt.

public **publishStudy** : Button
Dieser Button soll für die Veröffentlichung der Studie sorgen.

public **editStudy** : Button
Hier soll die Studie bearbeitet werden können.

public **finishStudy** : Button
Hiermit soll die Studie beendet werden können, sodass kein weiterer Benutzer mehr daran teilnehmen kann.l

public **resultStudy** : Button
Über diesen Button soll man sich die Studienergebnisse anschauen können.

View.UIElements.Lists.ListElements.UserListElement extends SelectionListElement

Die Klasse **UserListElement** repräsentiert einen User.

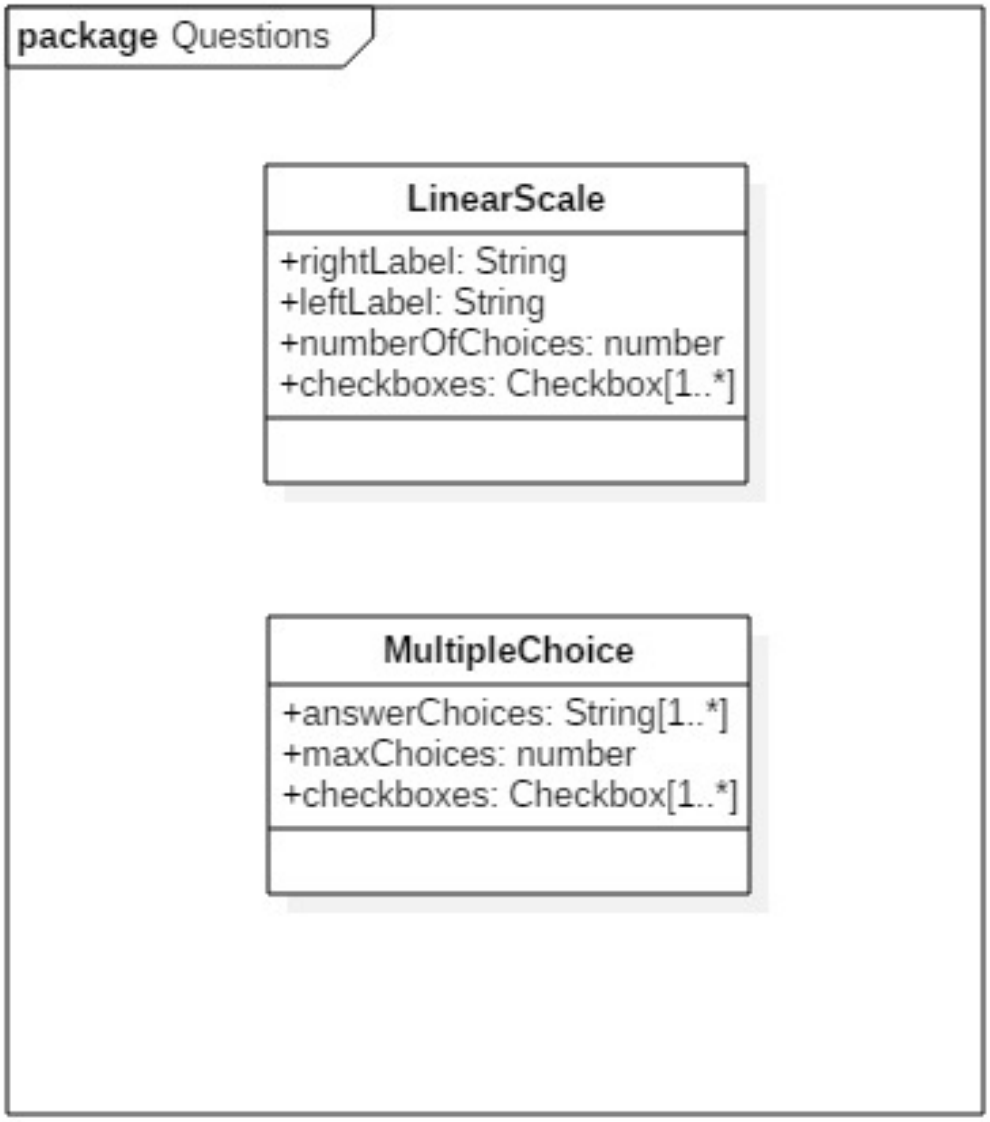

Abbildung 1.1.4: View.UIElements.Questions

View.UIElements.Questions
Hier liegen die Klassen, die zur Ansicht der Fragen dienen.

View.UIElements.Questions.LinearScale

Die Klasse public class **LinearScale** zeigt eine lineare Skala an.

public **rightLabel** : string. Zeigt die Bewertung auf der rechten Seite an, die möglich ist.

public **leftLabel** : string. Zeigt die Bewertung auf der linken Seite an, die möglich ist.

public **numberOfChoices** : number. Ist die Anzahl der Kästchen, die angekreuzt werden können.

public **checkboxes** : checkbox. Sind die Kästchen zum Auswählen der Antwort.

View.UIElements.Questions.MultipleChoice

Die Klasse public class **MultipleChoice** modelliert die Ansicht einer Multiple Choice Frage.

public **answerChoices** : String [..] ist die Bezeichnungen der möglichen Antworten.

public **maxChoices** : number ist die Anzahl der Antwortmöglichkeiten.

public **checkboxes** : checkbox. Sind die Kästchen zum Auswählen der Antwort.

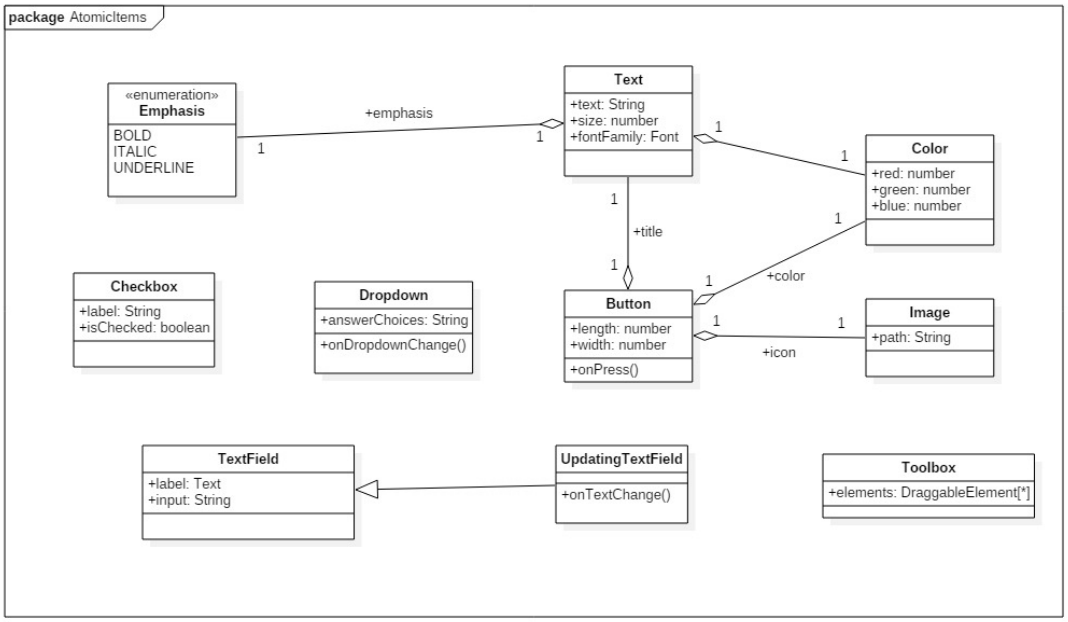

Abbildung 1.1.5: View.UIElements.AtomicItems

View.UIElements.AtomicItems

Hier liegen alle grundlegenden Elemente, die die Ansicht braucht.

View.UIElements.AtomicItems.Button

Die Klasse public class **Button** visualisiert einen Button.

public **length** : Number ist die Länge des Buttons

public **width** : Number ist die Breite des Buttons

public **color** : Color ist die Farbe des Buttons

public **title** : Text kann ein Text auf dem Button.

public **icon** : Image kann ein Bild auf dem Button sein.

View.UIElements.AtomicItems.Color

Die Klasse public class **Color** stellt eine Farbe dar.

public **red** : Number stellt den Rot-Teil einer RGB Farbe dar.

public **green** : Number stellt den Grün-Teil einer RGB Farbe dar.

public **blue** : Number stellt den Blau-Teil einer RGB Farbe dar.

View.UIElements.AtomicItems.Dropdown

Die Klasse public class **Dropdown** visualisiert ein Dropdown Menu.

public **answerChoices** : string [1..n] sind die Auswahlmöglichkeiten des Menü.

public void **onDropdownChange()** verarbeitet die Auswahl des Nutzers.

View.UIElements.AtomicItems.Image

Die Klasse public class **Image** gibt die Möglichkeit ein Bild anzeigen zu lassen.

public **Path** : String ist der Pfad, an dem das Bild gespeichert ist.

View.UIElements.AtomicItems.Text

Die Klasse public class **Text** zeigt einen Text an.

public **text** : String ist der eigentliche Text.

public **size** : Number ist die Schriftgröße des Textes.

public **fontFamily** : Font ist die Schriftart des Textes.

View.UIElements.AtomicItems.TextField

Die Klasse public class **TextField** visualisiert ein Textfeld.

public **label** : Text erscheint vor dem freien Feld zur Texteingabe.

public **input** : String ist der eingegebene Text des Users.

View.UIElements.AtomicItems.UpdatingTextField extends TextField

Die Klasse public class **UpdatingTextField** visualisiert ein Texfeld, das bei der Eingabe aktualisiert wird.

public void **onTextChange()** gibt weiter, wenn der Text sich verändert hat.

View.UIElements.AtomicItems.Emphasis

Das Enum public enum **Emphasis** dient zur Auswahl der Typografie

BOLD fett geschrieben

ITALIC kursiv geschrieben

UNDERLINE unterstrichen

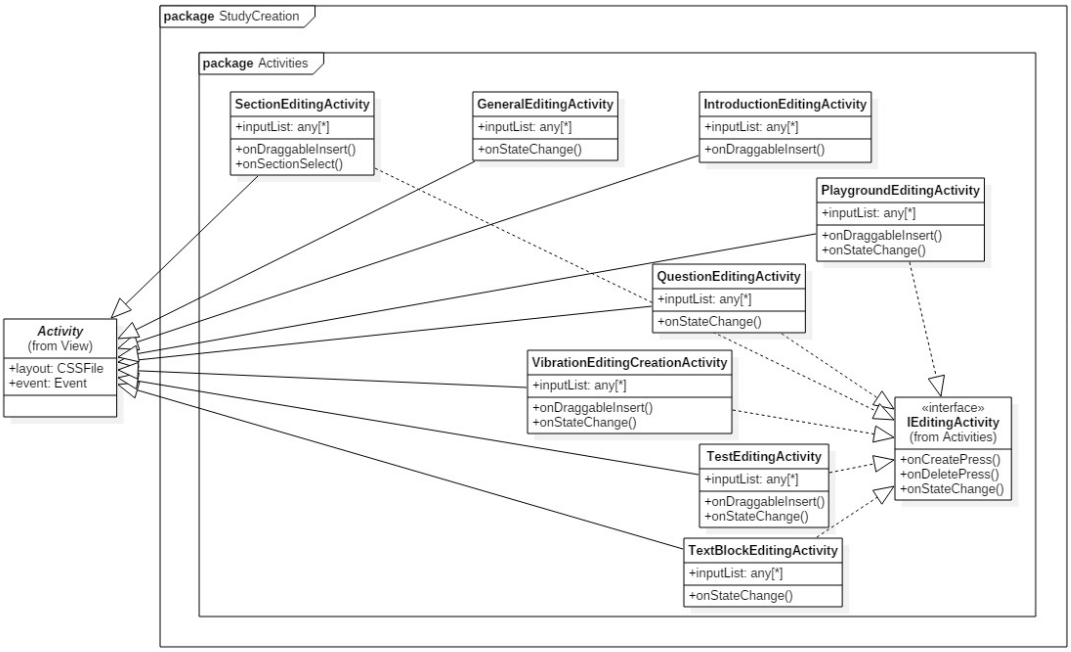

Abbildung 1.1.6: View.StudyCreation

View.StudyCreation.UIElements.AtomicItems.Toolbox

Die Klasse public class **Toolbox** modelliert den Werkzeugkasten. Aus ihr kann man zuvor erstellte Umfrage-Objekte als Draggable heraus- und in vorhandene Target-Areas in der Studienbearbeitung hineinziehen. Ein Objekt dieser Klasse kann nicht ohne eine Instanz der Klasse IntroductionEditingEvent, SectionEditingEvent, PlaygroundEditingEvent oder TestEditingEvent existieren.

public **elements** : DraggableElement[*] sind die Elemente, die in der Toolbox liegen.

1.2 ViewModel

Das ViewModel dient als Bindeglied zwischen View und Model. Es tauscht Informationen aus und enthält die Logik des Programms.

ViewModel.App

Die Klasse: public class **App** ist der Einstiegspunkt des Programms. Sie enthält die main()-Methode um VibroStudies zu starten.

public static **main()** : void
Ist die main()-Methode um die App aufzurufen.

ViewModel.Event

Die Klasse: public abstract class **Event** ist die Elternklasse aller anderen Event-Klassen. Diese wird vom AppRouter genutzt, um das Routing in andere Events zu ermöglichen. Dabei entspricht ein Event, einem Ereignis in der VibroStudies-App. Jedes Event hat demnach ihre eigene Activity (zu sehen im VibroStudies Pflichtenheft Abschnitt 10: Benutzerschnittstelle).

public **binding** : Activity enthält die passende Activity im View und liefert und erhält Daten.

public **neededPermission** : UserPermission
Ist das Attribut für die Zugriffsberechtigung eines Nutzers auf ein Event.

ViewModel.ICommand

Das Interface: public interface **IResultExporter** bietet eine Schnittstelle für das Ausführen bestimmte Befehle an. Es ist ein Bestandteil des Befehlsmusters. Alle Klassen, die ICommand implementieren, realisieren unterschiedliche Befehle.

public **execute()**: void
Ist der Methodenkopf der von allen anderen Command-Klassen implementiert wird.

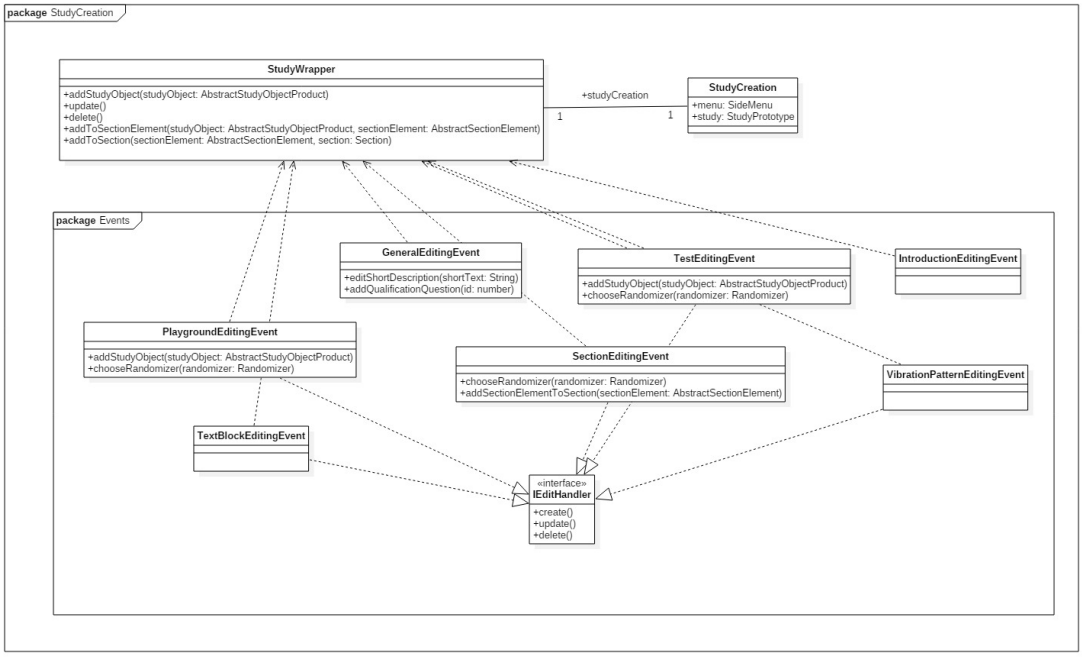

Abbildung 1.2.1: ViewModel.StudyCreation

1.2.1 Paket ViewModel.StudyCreation

Hier liegen alle Klassen, die bei der Studienerstellung relevant sind. Darunter die Verarbeitung der Eingaben des Benutzers (Paket Events) und das Zusammenbauen zu einer Studie.

ViewModel.StudyCreation.StudyCreation extends AppRouter

Die Klasse PUBLIC CLASS **StudyCreation** ist für das Routing bei der Erstellung einer Studie zuständig.

public **menu** : SideMenu ist das feste Menü.

public **study** : StudyPrototype ist die zu bearbeitende Studie.

ViewModel.StudyCreation.StudyWrapper

Die Klasse PUBLIC CLASS **StudyWrapper** baut aus den einzelnen Elementen die fertige Studie.

public **studyCreation** : StudyCreation ermöglicht Zugriff auf den Router bei Erstellung der Studie.

public **addStudyObject(studyObject: AbstractStudyObjectProduct)**: void fügt ein unsortiertes Objekt in eine Studie ein.

public **update(studyObject: AbstractStudyObjectProduct)**: void public **update(sectionElement: AbstractSectionElement)**: void public **update(section: Section)**: void

Die Update-Funktionen für die verschiedenen StudyObjects.

public **delete(studyObject: AbstractStudyObjectProduct)**: void public **delete(sectionElement: AbstractSectionElement)**: void public **delete(section: Section)**: void

Die Delete-Funktionen für die verschiedenen StudyObjects.

public **addToSectionElement(studyObject: AbstractStudyObjectProduct, sectionElement: AbstractSectionElement)**: void

Diese Methode fügt ein Objekt in ein SectionElement ein.

public **addToSection(sectionElement: AbstractSectionElement, section: Section)**: void

Diese Methode fügt ein SectionElement einer Section hinzu.

ViewModel.Events.IEditHandler

Das public interface **IEditHandler** erzwingt die Implementierung der Methoden, die zur Erstellung von Elementen, die in einer Studie vorkommen, benötigt werden.

public void **create()** erstellt ein neues Objekt.

public void **update()** verändert das Objekt.

public void **delete()** löscht das Objekt.

ViewModel.StudyCreation.Events.IntroductionEditingEvent

Die public class **IntroductionEditingEvent** ist für die Erstellung, Aktualisierung und Entfernung der Einführung verantwortlich.

public void **chooseRandomizer(randomizer: Randomizer)** gibt weiter, welche Randomisierungsmethode verwendet werden soll.

public void **addSectionElementToSection(sectionElement: AbstractSectionElement)** fügt ein SectionElement zu einer Section hinzu.

ViewModel.StudyCreation.Events.GeneralEditingEvent

public **editShortDescription(shortText: String)**
Erhält von der View einen String und dditiert die ShortDescription über den StudyWrapper im Model.

public **addQualificationQuestion(id: number)**
Erhält eine ID aus der homelessStudyobjects, übergibt diese an den StudyWrapper welcher die ID im Modell aus der homlessStudyobjects Liste aufruft und die QualificationQuestion hinzufügt.

ViewModel.StudyCreation.Events.SectionEditingEvent implements EditHandler

Die public class **SectionEditingEvent** ist für die Erstellung, und Entfernung von Sections verantwortlich. Außerdem legt die Klasse für die Randomisierungsmethoden fest.

public void **chooseRandomizer(randomizer: Randomizer)** gibt weiter, welche Randomisierungsmethode verwendet werden soll.

public void **addSectionElementToSection(sectionElement: AbstractSectionElement)** fügt ein SectionElement zu einer Section hinzu.

ViewModel.StudyCreation.Events.TextBlockEditingEvent implements IEditHandler

Die public class **TextBlockEditingEvent** ist für die Erstellung, Aktualisierung und Entfernung des TextBlock-Objekts verwantwortlich.

ViewModel.StudyCreation.Events.VibrationPatternEditingEvent implements IEditHandler

Die public class **VibrationPatternEditingEvent** ist für die Erstellung, Aktualisierung und Entfernung des Vibrationsmustern-Objekts verwantwortlich.

ViewModel.StudyCreation.Events.QuestionEditingEvent implements IEditHandler

Die public class **QuestionEditingEvent** ist für die Erstellung, Aktualisierung und Entfernung des Questions-Objekts verwantwortlich.

ViewModel.StudyCreation.Events.PlaygroundEditingEvent implements IEditHandler

Die public class **PlaygroundEditingEvent** ist für die Erstellung, Aktualisierung und Entfernung des Playground-Objekts verwantwortlich.

public boolean **addStudyObject(studyObject: AbstractStudyObjectProduct)** fügt dem Playground ein Studien-Objekt hinzu.

public void **chooseRandomizer(randomizer: Randomizer)** gibt weiter, welche Randomiseriungsmethode verwendet werden soll.

ViewModel.StudyCreation.Events.TestEditingEvent implements IEditHandler

Die public class **TestEditingEvent** ist für die Erstellung, Aktualisierung und Entfernung des Test-Objekts verwantwortlich.

public boolean **addStudyObject(studyObject: AbstractStudyObjectProduct)** fügt dem Test ein Studien-Objekt hinzu.

public void **chooseRandomizer(randomizer: Randomizer)** gibt weiter, welche Randomiseriungsmethode verwendet werden soll.

1.2.2 Paket ViewModel.Routing

Dieses Paket steuert den Ablauf der Software.

ViewModel.Routing.AppRouter

Der public abstract class **AppRouter** beinhaltet die Routen zu allen Events und deren Activities. Außerdem ist er für die Anzeige und den Wechsel von Events zuständig.

public **currentEvent** : Event ist das aktuelle Event.

public void **changeActivity(event: Event)** wechselt auf das übergebene Event.

ViewModel.Routing.MenuElement

Die public class **MenuElement** beinhaltet das passende Event und kann es öffnen.
public **state** : Event ist das zugehörige Event zum MenuElement.

public void **openActivity()** lädt im AppRouter das eigene Event als currentEvent.

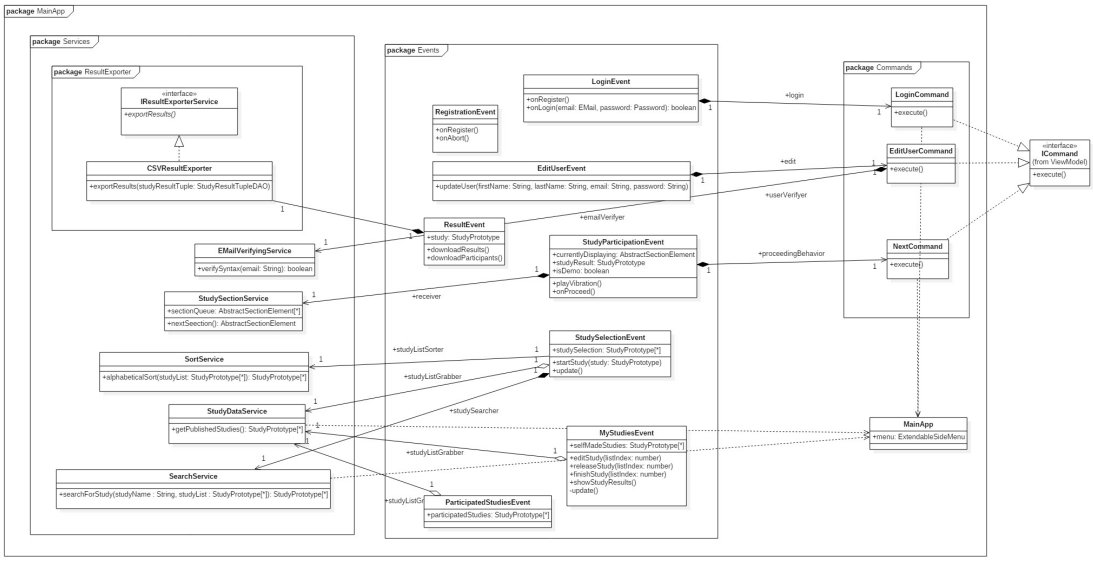

Abbildung 1.2.2: ViewModel.MainApp

1.2.3 Paket ViewModel.MainApp

In diesem Paket liegen alle Klassen, die die restlichen Eingaben des Benutzers verarbeiten und sie an das Model weiter geben (Pakete Events und Commands). Im Paket Services liegen alle wichtigen Dienstleistungen, wie z.B. das Exportieren der Ergebnisse in eine Datei.

ViewModel.MainApp.MainApp

Die Klasse public class **MainApp** beeinhaltet das Routing der Hauptfunktionen.

public **menu** : ExtendableSideMenu ist das Menu der Hauptapp

ViewModel.MainApp.Events.RegistrationEvent extends Event

Die Klasse: public class **RegistrationEvent** ist für die Datenübertragung bei einer Nutzerregistrierung zuständig.

public **onRegister()**: void
Die eingegebeenen Parameter (Vorname, Nachname, E-Mail, Passwort) werden als neue Accountdaten aufgefasst und aus der UserDAO die save()-Methode aufgerufen, um einen neuen Account anzulegen. Standardmäßig wird UserPermission auf DEFAULT gesetzt.

public **onAbort()**: void
Es werden alle bereits eingegebenen Parameter gelöscht und der AppRouter wechselt zum LoginEvent.

ViewModel.MainApp.Events.LoginEvent extends Event

Die Klasse: public class **RegistrationEvent** ist für das Verifizieren der Anmeldedaten eines Nutzers zuständig.

public **login** : LoginCommand
LoginCommand Instanz zur Ausführung eines Logins.

public **onRegister()**: void
Es werden alle bereits eingegebenen Parameter gelöscht und der AppRouter wechselt zum RegistrationEvent.

public **onLogin(eMail : String, password : String)**: boolean

- **eMail**: Es ist der vom Nutzer eingegebene String für die E-Mail.
- **password**: Es ist der vom Nutzer eingegebene String für das Passwort.

Ruft die execute()-Methode aus login mit eMail und password als dessen Parameter auf. Wenn ein Nutzer mit solchen Eingabedaten existiert, wird TRUE zurückgegeben und zum StudySelectionEvent gewechselt. Wenn nicht, wird FALSE zurückgegeben.

ViewModel.MainApp.Events.EditUserEvent extends Event
Die Klasse: public class **RegistrationEvent** ist für das Verifizieren der Anmeldedaten eines Nutzers zuständig.

public **edit** : EditUserCommand
EditUserCommand-Instanz für das Ausführen der Nutzer-Editierung.

public **updateUser(firstName : String, lastName : String, eMail : String, password : String)**: void

- **firstName**: Es ist der vom Nutzer eingegebene String für den Vornamen.
- **lastName**: Es ist der vom Nutzer eingegebene String für den Nachnamen.
- **eMail**: Es ist der vom Nutzer eingegebene String für die E-Mail.
- **password**: Es ist der vom Nutzer eingegebene String für das Passwort.

Die Accountdaten des Nutzers werden aktualisiert, indem die ID des eingeloggten Users zusammen mit den Parametern als neue Attribute in die update()-Methode des UserDAO eingegeben werden.

ViewModel.MainApp.Events.StudySelectionEvent extends Event
Die Klasse: public class **StudySelectionEvent** ist für das Weitergeben der Liste aller veröffentlichenten Studien, sowie für die Übertragung zuständig, welche Studie gesucht oder ausgewählt wurde.

public **studySelection** : StudyProtottype[*]
Ist eine Liste der bisher veröffentlichten StudiePrototype-Objekten, an welcher der Nutzer noch nicht teilgenommen hat.

public **studyListGrabber** : StudyDataService
StudyDataService Instanz für das Laden der Liste von veröffentlichten Studien.

public **studyListSorter** : SortService
SortService Instanz für das Sortieren der Liste von veröffentlichten Studien.

public **studySearcher** : SearchService
SearchService Instanz für das Suchen von beliebig vielen Studien in der Liste von veröffentlichten Studien.

public **startStudy(study : StudyPrototype)**: void

- study: Die vom Nutzer ausgewählte Studie wird hier in die Methode als StudyPrototype-Objekt übergeben.

Der Parameter study wird dem Konstruktor von StudyParticipationEvent für das Attribut StudyParticipationEvent.studyResult übergegeben und es wird zu diesem Event gewechselt.

private **update()**: void
Vom StudyListGarabber wird nochmals die getPublishedStudies()-Methode aufgerufen. Diese kriegt zusätzlich den User als Parameter, um zu unterscheiden, an welcher Studie dieser User teilgenommen hat. studySelection wird mit dem neuen Rückgabewert gesetzt.

ViewModel.MainApp.Events.MyStudiesEvent extends Event
Sie ist für das Weiterleiten von Operationen zuständig, welche auf eine Studie anwendbar sind. Diese sind das Editieren, Update, Veröffentlichen und Beenden einer Studie, sowie das Anzeigen der Studienteilnehmer und -ergebnisse.

public **selfMadeStudies** : StudyPrototype[*]
Ist eine Liste der selsbt erstellten StudyPrototype-Objekten.

public **studyListGrabber** : StudyDataService
StudyDataService Instanz für das Laden der Liste aller selbsterstellten StudyPrototypes.

public **editStudy(listIndex : number)**: void

- **listIndex**: Ist der Index des StudyPrototypes in selfMadeStudies

Wenn der Nutzer mit einem Smartphone eingeloggt ist, wird ihm eine Fehlermeldung angezeigt, der ihn dazu auffordert zur Desktop-Version zu wechseln.
Wenn der Nutzer sich auf dem Desktop eingeloggt hat, wird durch den jeweiligen listIndex von selfMadeStudies der dazugehörige StudyPrototype bestimmt zum GeneralEditingEvent übertragen und dorthin gewechselt.

public **releaseStudy(listIndex : number)**: void

- **listIndex**: Ist der Index des StudyPrototypes in selfMadeStudies

An Hand des Index wird der ausgewählte StudyPrototype bestimmt. Mit der StudyResultTupleDAO wird mit der ID des StudyPrototypes die update()-Methode aufgerufen und isPublished auf TRUE gesetzt. Nach der Veröffentlichung kann die Studie nicht mehr

bearbeitet werden.

public **finishStudy(listIndex : number)**: void

- **listIndex**: Ist der Index des StudyPrototypes in selfMadeStudies

An Hand des Index wird der ausgewählte StudyPrototype bestimmt. Mit der StudyResultTupleDAO wird mit der ID des StudyPrototypes die update()-Methode aufgerufen und isFinished auf TRUE gesetzt. Nach der Veröffentlichung kann die Studie nicht mehr bearbeitet werden.

public **showStudyResults()**: void
Es wird zum ResultEvent gewechselt.

private **update()**: void
Vom StudyListGarabber wird nochmals die getPublishedStudies()-Methode aufgerufen. Diese kriegt zusätzlich den User als Parameter, um zu unterscheiden, welche Studie vom jeweiligen User erstellt wurde. selfMadeStudies wird mit dem neuen Rückgabewert gesetzt.

ViewModel.MainApp.Events.ParticipatedStudiesEvent extends Event

Die Klasse: public class **ParticipatedStudiesEvent** ist für das Weiterleiten der Liste von bereits teilgenommenen Studien zuständig.

public **participatedStudies** : StudyPrototype[*]
Ist eine Liste von StudyPrototypes an welcher der Nutzer teilgenommen hat.

public **studyListGrabber** : StudyDataService
StudyDataService Instanz für das Laden der Liste von veröffentlichten Studien.

ViewModel.MainApp.Events.StudyParticipationEvent extends Event

Die Klasse: public class **StudyParticipationEvent** ist für die Studienteilnahme, -durchführung, sowie -speicherung zuständig. Dazu wird die Studie in Segmente eingeteilt und die Datenübertragung bei Nutzereingaben in den Segmenten verwaltet.

public **currentlyDisplaying** : AbstractSectionElement

Attribut für die aktuelle Seite der Studie.

public **studyResult** : StudyPrototype
Attribut für die aktuelle Studie an welcher der Nutzer aktuell teilnimmt und in welcher die Ergebnisse gespeichert werden.

public **isDemo** : boolean
Attribut um festzulegen, ob die durchzuführende Studie eine Demo ist, um zu wissen ob die Ergebnisse gespeichert werden oder nicht.

public **receiver** : StudySegmentService
StudySegmentSerive Instanz für das Laden eines weiteren Studiensegments.

public **proceedingBehaviour** : NextCommand
NextCommand Instanz, welches für das Abspeichern aller Ergebnisse eines Segments zuständig ist.

constructor (studyResult : StudyPrototype)

- studyResult: Sie ist die aktuell ausgewählte Studie.

Der Konstruktor setzt die aktuell teilzunehmende Studie auf studyResult.

public **playVibration()**: void
Ruft die Vibrationsmuster aus dem aktuellen Segment auf.

public **onProceed()**: void
Ruft die nextSegment()-Methode vom receiver auf.

ViewModel.MainApp.Events.ResultEvent extends Event
Die Klasse: public class **ResultEvent** wird für den Download der Studienergebnisse und -teilnehmer verwendet.

public **study** : StudyPrototype
Sie ist die aktuell ausgewählte Studie als StudyPrototype.

public **downloadResults()**: void
Auf dem eingeloggten Endgerät werden die Studienergebnisse als CSV-Datei lokal abgespeichert.

public **downloadParticipants()**: void
Auf dem eingeloggten Endgerät werden die Studienteilnehmer als CSV-Datei lokal abgespeichert.

ViewModel.MainApp.Commands.LoginCommand implements ICommand

Die Klasse: public class **LoginCommand** ist für die Überprüfung der Accountdaten des Nutzers zuständig, sodass er sich mit seinem Account einloggen kann.

public **execute()** : boolean
Ruft mit den Eingabedaten des Nutzers die get()-Methode von PUBLIC CLASS USERDAO auf, falls sich ein User-Objekt mit den eingegebenen Parametern finden lässt. Lässt sich einer finden wird der Nutzer als loggedInUser im PUBLIC ABSTRAT CLASS APPROUTER gesetzt und als Rückgabewert wird TRUE zurückgegeben. Falls kein Nutzer gefunden wird, wird auch loggedInUser nicht gesetzt und FALSE zurückgegeben.

ViewModel.MainApp.Commands.EditUserCommand implements ICommand

Die Klasse: public class **EditUserCommand** ist für die Editierung der Accountdaten des Nutzers zuständig, sodass sie in der Datenbank auf dem aktuellen Stand gehalten werden.

public **eMailVerifyer** : EMailVerifyingService
UserVerifyingSerive Instanz für das Verifizieren der neuen E-Mail.

public **execute()**: boolean
Bevor die neuen Eingabedaten in die Datenbank abgespeichert werden, wird der eMail-Verifyer aufgerufen, welcher lokal ein E-Mail-Objekt mit dem angegebenen E-Mail String erzeugt um ihre syntaktische Korrektheit zu überprüfen. Ist es nicht korrekt, wird das Objekt gelöscht und der Rückgabewert ist FALSE. Stimmt die E-Mail, wird das E-Mail-Objekt mit den anderen Eingabedaten (Vor-, Nachname, Passwort), sowie mit der ID des editierten Accounts in die update()-Methode von PUBLIC CLASS USERDAO übergeben und TRUE zurückgegeben.

ViewModel.MainApp.Commands.NextCommand implements ICommand

Die Klasse: public class **NextCommand** enthält die Methode für die Speicherung der

Ergebnisse.

public **execute()**: boolean
Falls currentDisplaying nicht leer ist, kann der Nutzer diese auch bearbeiten und das Event läuft weiter. Hat er dies getan, wird die AbstractSectionElement in studyResult, welche die gleiche ID wie currentDisplaying hat, von currentDisplaying ersetzt. Die Methode gibt daraufhin TRUE als Rückgabewert wieder.
Ist currentDisplaying jedoch leer, wird studyResult in die Datenbank mittels der addResult()-Methode vom StudyPrototypeResultList geschrieben und auch in diesem Objekt aufbewahrt. Die Methode gibt daraufhin FALSE als Rückgabewert wieder, um mitzuteilen, dass kein weiters StudySectionElement vorhanden ist und die Studie abgeschlossen wurde.

ViewModel.MainApp.Services.EMailVerifyingService

Die Klasse: public class **EMailVerifyingService** ist für die Validationstechnik werden jeweils Methoden angeboten.

public **verifySyntax(String eMail)**: boolean

- **eMail**: Ist ein String der vom Nutzer in das E-Mail Eingabefeld eingegeben wurde.

Es wird mit eMail ein neues E-Mail Objekt erzeugt, welches die verifyCorrectness()-Methode auf sich selbst aufruft. Der Rückgabewert von verifyCorrectness() wird als Rückgabewert dieser Methode genutzt. Es wird FALSE zurückgegeben, wenn die Syntax der E-Mail falsch ist und TRUE zurückgegeben wenn die Syntax der E-Mail richtig ist.

ViewModel.MainApp.Commands.StudyDataService

Die Klasse: public class **StudyDataService** ist für das Rausfiltern der bereits veröffentlichten Studien zuständig. Das Filtern und das Setzen der Studienliste übernimmt ihre Methode.

public **getPublishedStudies()** : StudyPrototype[*]
Es wird eine leere StudyPrototype-Liste erzeugt in der die veröffentlichten Studien eingefügt werden. Es wird die getAll()-Methode aus der StudyResultTupleDAO aufgerufen um die Liste der StudyResultTuple zu bekommen und für jedes dieser Tupel wird getStudyPrototype() aufgerufen und in die StudyPrototype-Liste eingefügt. Wenn über die Liste der StudyResultTuple fertig iteriert wurde, werden die noch nicht veröffentlich-

ten StudyPrototypes aus der Liste entfernt. Dafür wird für jedes StudyPrototype die getIsPublished()-Methode aufgerufen. Nach dem Entfernen wird die StudyPrototype-Liste zurückgegeben.

ViewModel.MainApp.Commands.SearchService

Die Klasse: public class **public class SearchService** ist für die Suche der StudyPrototypes innerhalb einer Liste zuständig, welche übereinstimmende Studiennamen haben.

public **searchForStudy(studyName : String, studyList : StudyPrototype[*])**: StudyPrototype[*]

- **studyName**: Ist ein String der vom Nutzer in das Sucheingabefeld eingegeben wurde. Dieser steht für den gewünschten Studiennamen.
- **studyList**: Ist eine Liste von StudyPrototypes, die gefiltert werden muss.

Es wird eine leere StudyPrototype-Liste erzeugt in der die StudyPrototypes eingefügt werden, die den gesuchten Studiennamen entsprechen. Es wird über studyList iteriert und für jeden StudyPrototype die getName()-Methode aufgerufen und mit der studyName verglichen. Jedes StudyPrototype mit übereinstimmenden Studiennamen wird in die Liste eingefügt. Wenn über studyList fertig iteriert wurde, wird die Liste zurückgegeben.

ViewModel.MainApp.Services.SortService

Die Klasse: public class **SortService** ist für die Sortierung von StudyPrototype-Listen zuständig. Dafür bietet sie Methoden für die Sortierung an.

public **alphabeticalSort(studyList : StudyPrototype[*])**: StudyPrototype[*]

- **studyList**: Ist eine Liste von StudyPrototypes, die sortiert werden sollen.

studyList wird alphabetisch sortiert und zurückgegeben.

ViewModel.MainApp.Services.StudySectionService

Die Klasse: public class **StudySectionService** ist für die Segmentierung eines StudyPrototypes zuständig. Sie bietet eine Methode an, um die nächste Studienseite in die

Sicht des Nutzers zu laden. Eine Studienseite entspricht einem AbstractSectionElement der Studie.

public **sectionQueue** : AbstractSectionElement[*]
Ist eine FIFO Queue für AbstractSectionElement-Objekte.

constructor (study : StudyPrototype)

- study: Sie ist die aktuell ausgewählte Studie als StudyPrototype.

Der Konstruktor ruft die private convertToSectionElementQueue()-Methode für study auf und sectionQueue wird auf den Rückgabewert gesetzt.

public **nextSection()** : AbstractionSectionElement
Ruft die remove()-Methode (herkömmliche Definition bei einer Queue) für die sectionQueue auf und gibt das AbstractionSectionElement zurück.

pirvate **convertToSectionElementQueue(study : StudyPrototype)**: AbstractSectionElement[*]

- **study**: Sie ist die aktuell ausgewählte Studie als StudyPrototype.

Sie konvertiert die studyx in eine FIFO Queue von StudyPrototypes.

ViewModel.MainApp.Services.ResultExplorer.IResultExporter

Das Interface: public interface **IResultExporter** bietet eine Schnittstelle für das Exportieren der Studienergebnisse an. Alle Klassen, die IResultExplorter implementieren, werden verschiedene Datentypen zum Export anbieten.

public abstract **exportResults()**: void Ist der Methodenkopf der von allen anderen Exportklassen implementiert werden soll, um die Studienergebnisse in den jeweiligen Datentyp umzuwandeln und zu exportieren.

ViewModel.MainApp.Services.ResultExporter.CSVResultExporter implements IResultExporter

Die Klasse: public class CSVResultExplorer besitzt eine Methode, um ein StudyResulTuple-Objekt zu einer CSV-Datei umzuwandeln und zu exportieren.

public **exportResults(tuple : StudyResultTuple)**: void

- **tuple**: Ist das StudyResultTuple, welches exportiert werden muss.

Es werden die nötigen Daten aus study durch die jeweiligen get()-Methoden ermittelt (bspw. Name des Studienleiters, etc.) und in eine CSV-Datei geschrieben. Die fertige Datei wird anschließend lokal auf dem Endgerät des Benutzers gespeichert.

1.3 Model

Das Paket Model enthält jene Klassen und Interfaces, welche die Studie und die daran beteiligten User modellieren und strukturiert auf Datenebene darstellen.

Model.ModelSingelton

Das Einzelstück public class **ModelSingelton** stellt die Instanz des Models dar.

private static **singelton** : ModelSingelton implementiert das Entwurfsmuster Einzelstück.

public **studyAccess** : studyResultTupleDAO beinhaltet alle erstellten Studien mit ihren Ergebnissen.

public **userAccess** : UserDAO beinhaltet alle registrierten User mit ihren Ergebnissen.

public ModelSingelton **getInstance()** gibt die eine Instanz zurück.

Model.IDAO<T>

Das Interface: public interface **IDAO<T>** stellt eine Schnittstelle zur Verfügung um Daten aus dem Model abzurufen und im Model zu speichern.

public **save()**: boolean
Diese Methode speichert einen Eintrag.

public **update()**: boolean
Die Methode updated einen Eintrag.

public **delete()**: boolean

Die Methode löscht einen Eintrag.

public **get()**: T
Die Methode gibt ein bestimmtes Object zurück.

public **getAll()**: T[*]
Die Methode gibt eine Liste aller Obejcts zurück.

Model.IPrototype<T>

Das Interface public interface **IPrototype<T>** bietet eine Methode an, um neue Instanzen zu erzeugen.

public **clone()** : T Die Methode erzeugt eine neue Instanz einer Klasse.

Model.IVerifiableAttribute

Das Interface: public interface **IVerifiableAttribute** ist eine Schnittstelle, die jene Klassen implementieren, die Attribute besitzen, die auf syntaktische oder semantische Korrektheit überprüft werden müssen.

public **isCorrect()**: boolean
Allgemeine Methode um zu überprüfen, ob die Attribute einer Klasse eine valide Belegung haben.

1.3.1 Paket Model.Study

Das Model.Study Paket enthält jene Klassen, die die Struktur und Inhalte einer Nutzerstudie modellieren.

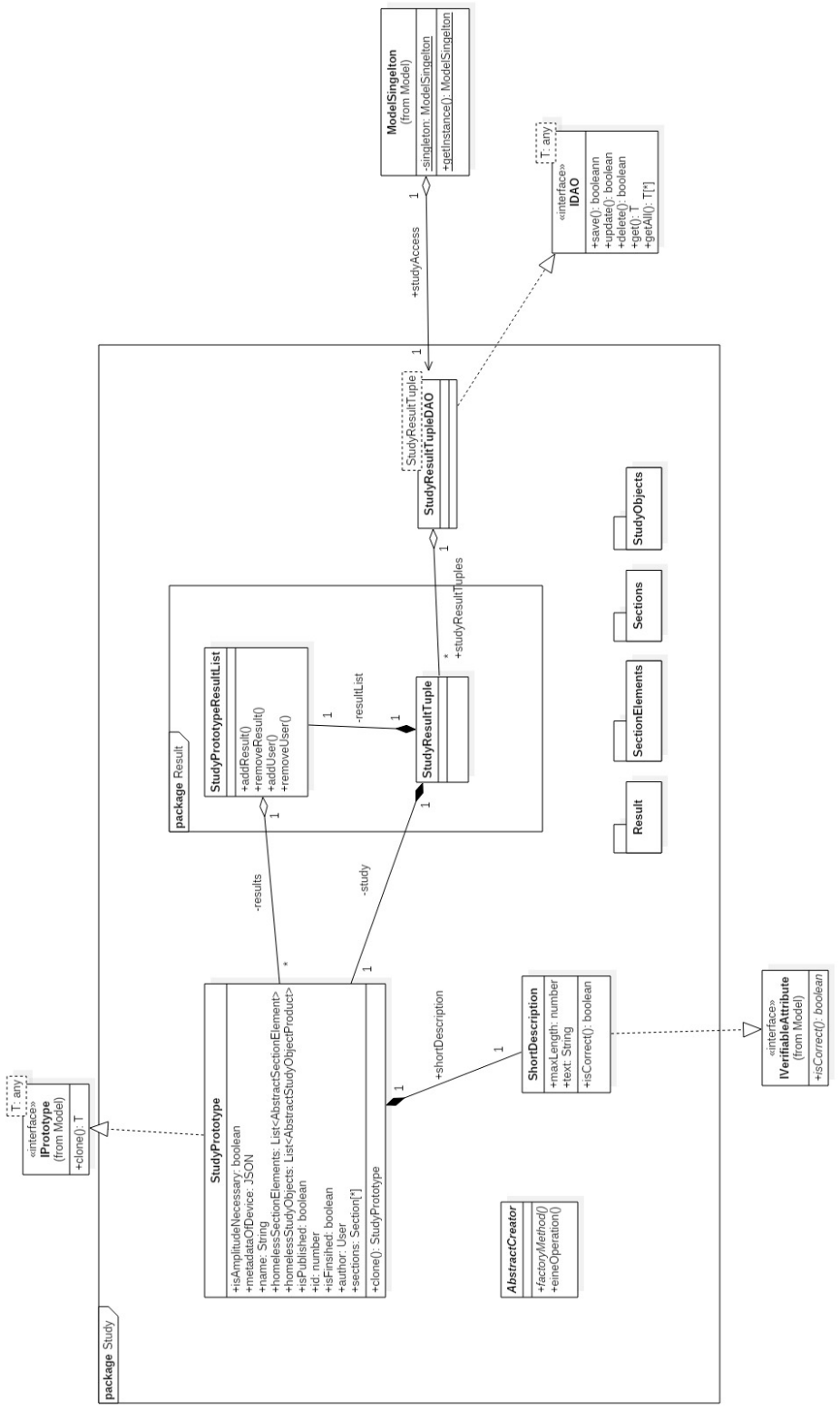

Abbildung 1.3.1: Study

Model.Study.StudyPrototype implements IPrototype

Die Klasse public class **StudyPrototype** ist ein Template für die Instanz einer Studie, in der gleichzeitig das Ergebnis nach Ausführung der Studie durch einen User gespeichert wird.

public **isAmplitudeNecessary**: boolean
Dieses Attribut gibt an, ob das Gerät auf dem die Studie ausgeführt wird, die Amplitude unterstützen muss.

public **metadataOfDevice**: JSON
In diesem Attribut werden die Metadaten des Endgerätes als JSON gespeichert.

public **name**: String
Dieses Attribut speichert den Namen der Studie.

public **homlessSectionElements**: List<AbstractSectionElement>
In dieser Liste werden Section Elemente gespeichert, die nicht zu einer konkreten Section gehören. Beim Einfügen in eine Section werden die Elemente aus dieser Liste gelöscht.

public **homlessStudyObjects**: List<AbstractStudyObjectProduct>
In dieser Liste werden bereits erstellte Study Objects gepeichert, die nicht zu einem kokreten Section Element gehören. Die Elemente werden aus der Liste gelöscht sobald sie einem Section Element hinzugefügt werden.

public **isPublished**: boolean
Dieses Attribut gibt an, ob eine Studie bereits veröffentlicht wurde.

public **id**: number
Dieses Attribut gibt einer (nicht ausgeführten) Studie eine eindeutige ID um sie von anderen Studien zu unterscheiden.

public **author**: User
Dieses Attribut gibt es Autoren einer Studie an.

public **sections** : Section
enthält alle Sections der Studie

public **shortDescription** : ShortDescription enthält die Kurzbeschreibung.

public void **clone()**: StudyPrototype Diese Methode klont einen StudyPrototype, damit dieses von einem User bearbeitet werden kann.

Model.Study.AbstractCreator

public abstract **factoryMethod**
public **eineOperation**:

Model.Study.ShortDescription implements IVerifiableAttribute

private final **maxLength**: number
Dieses Attribut gibt an, wie lang die ShortDescription sein darf.

public **text**: String
Dieses Attribut speichert den Text der ShortDescription, der in der Studienübersicht angezeigt wird.

public **isCorrect**: boolean
Diese Methode verifiziert, ob die ShortDescription den Anforderungen entspricht.

Model.Study.StudyResultTuple

Die Klasse public class **StudyResultTuple** ist eine Klasse, um die Ergebnisse einer Studie der User Liste zuzuordnen.

public **studyResultList** : StudyPrototypResultList ist die Instanz, die alle Ergebnisse zur entsprechenden Studie beinhaltet.

public **study** : studyPrototype ist die Studie, die ausgefüllt werden kann.

Model.Study.StudyResultTupleDAO implements IDAO

Die Klasse public class **StudyResultTupleDAO** ist eine Klasse um einen geregelten Zugriff über das DAO Muster auf die StudyResultTuple zu ermöglichen.

public **studyResultTuples** : studyResultTuple[*] enthält die Liste der StudyResultTuples.

public **save()**: boolean
Speichert ein StudyResultTuple Element.

public **update()**: boolean

Updated ein bestehendes StudyResultTuple Element.

public **delete()**: boolean
Löscht ein bestehendes StudyResultTuple Element.

public **get()**: StudyResultTuple
Gibt ein spezifisches StudyResuötTuple Element zurück.

public **getAll()**: List<StudyResultTuple>
Gibt alle existierenden StudyResultTuple Elemente in einer Liste zurück.

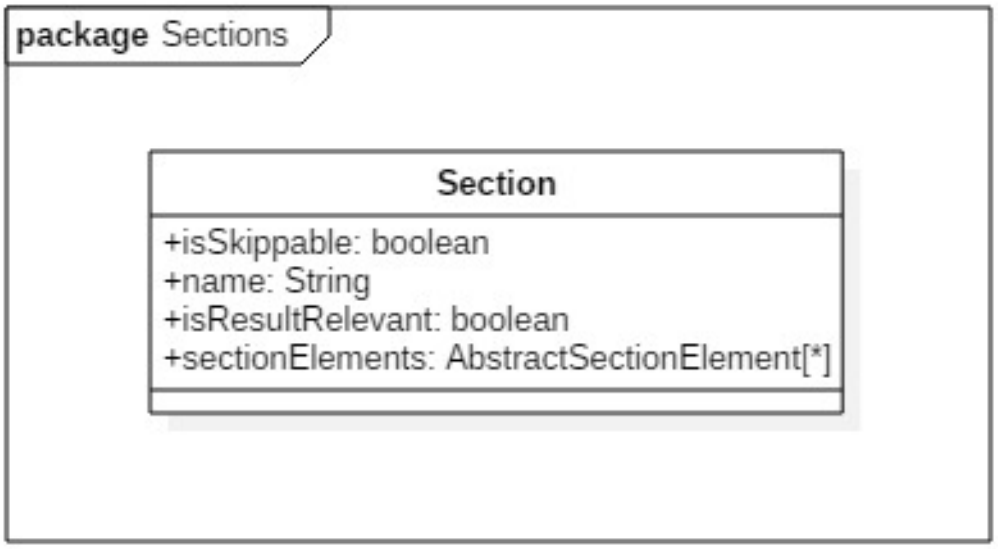

Abbildung 1.3.2: Sections

Model.Study.Sections.Section

Die Klasse: PUBLIC CLASS **SECTION** enthält die grundlegenden Informationen, die eine abstrakte Section benötigt um weiter konkretisiert und in eine Studie eingegliedert zu werden.

public **sectionElements**: List<AbstractSectionElement> enthält die Elemente der Container für die atomaren Bestandteile der Studie sind.

public **isSkippable**: boolean

Dieses Attribut gibt an, ob eine Section übersprungen werden darf.

public**isResultRelevant**: boolean
Dieses Attribut legt fest, ob die Auswertung einer Section in die Ergebnisse eingetragen werden soll und somit relevant für die Statistische Auswertung wird.

public **name**: String
Dieses Attribut enthält den Namen der Section.

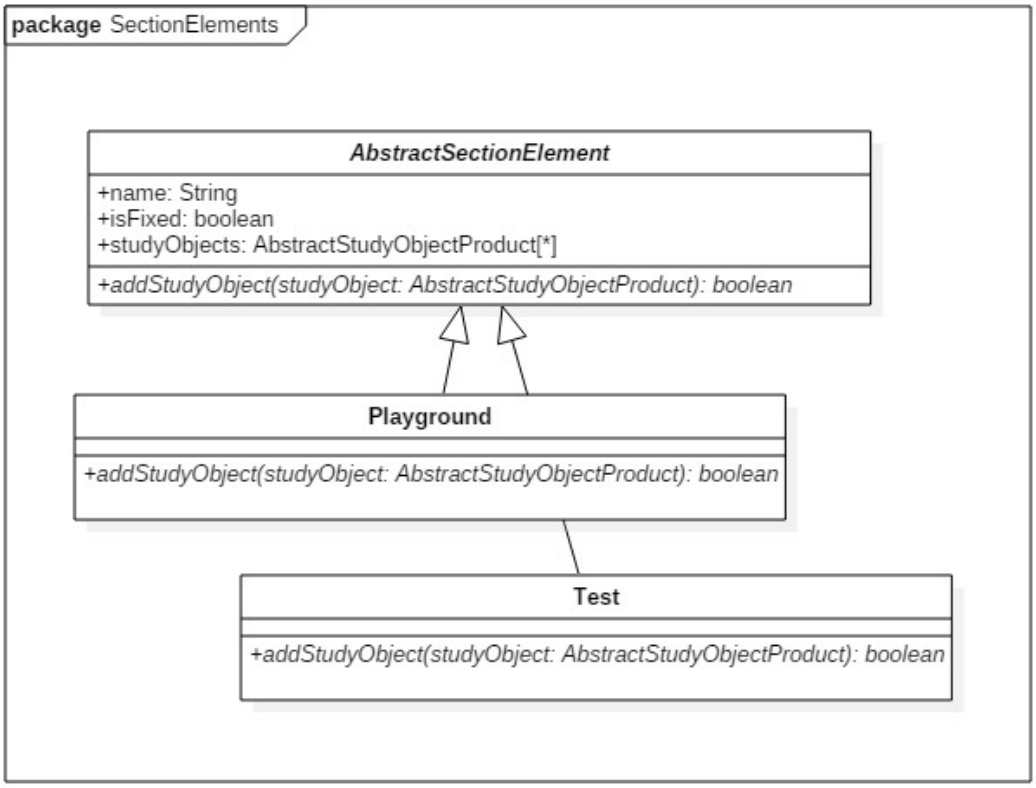

Abbildung 1.3.3: Section Elements

Model.Study.SectionElements.AbstractSectionElement

Die Klasse: public abstract class **AbstractSectionElement** ist ein Container für StudyObjects um diese zu partitionieren.

public **studyObjects**: List<AbstractStudyObjectProduct>

Eine Liste an elementaren Objekten mit denen der User einer Studie interagiert.

public **name**: String
Name des Elements.

public **isFixed**: boolean
Das Attribut gibt an, ob das AbstractSectionElement innerhalb der Section an eine feste Position gesetzt wird.

public **addStudyObject(studyObject:AbstractStudyObjectProduct** : boolean fügt ein neues Studienobjekt ein und überprüft, ob passende Studienobjekte eingefügt werden.

Model.Study.SectionElements.Playground extends AbstractSectionElement

Die Klasse: public class **Playground** ist ein logischer Container für Vibrationpatterns und Textblocks.

public **addStudyObject(studyObject:AbstractStudyObjectProduct** : boolean fügt ein neues Studienobjekt ein und überprüft, ob das einzufügende Studienobjekte ein Vibrationsmuster oder ein TextBlock ist.

Model.Study.SectionElements.Test extends AbstractSectionElement

Die Klasse: public class **Test** ist ein logischer Container für Vibrationpatterns und Textblocks und Frageobjekte.

public **addStudyObject(studyObject:AbstractStudyObjectProduct** : boolean fügt ein neues Studienobjekt ein und überprüft, ob das einzufügende Studienobjekte ein Vibrationsmuster, eine Frage oder ein TextBlock ist.

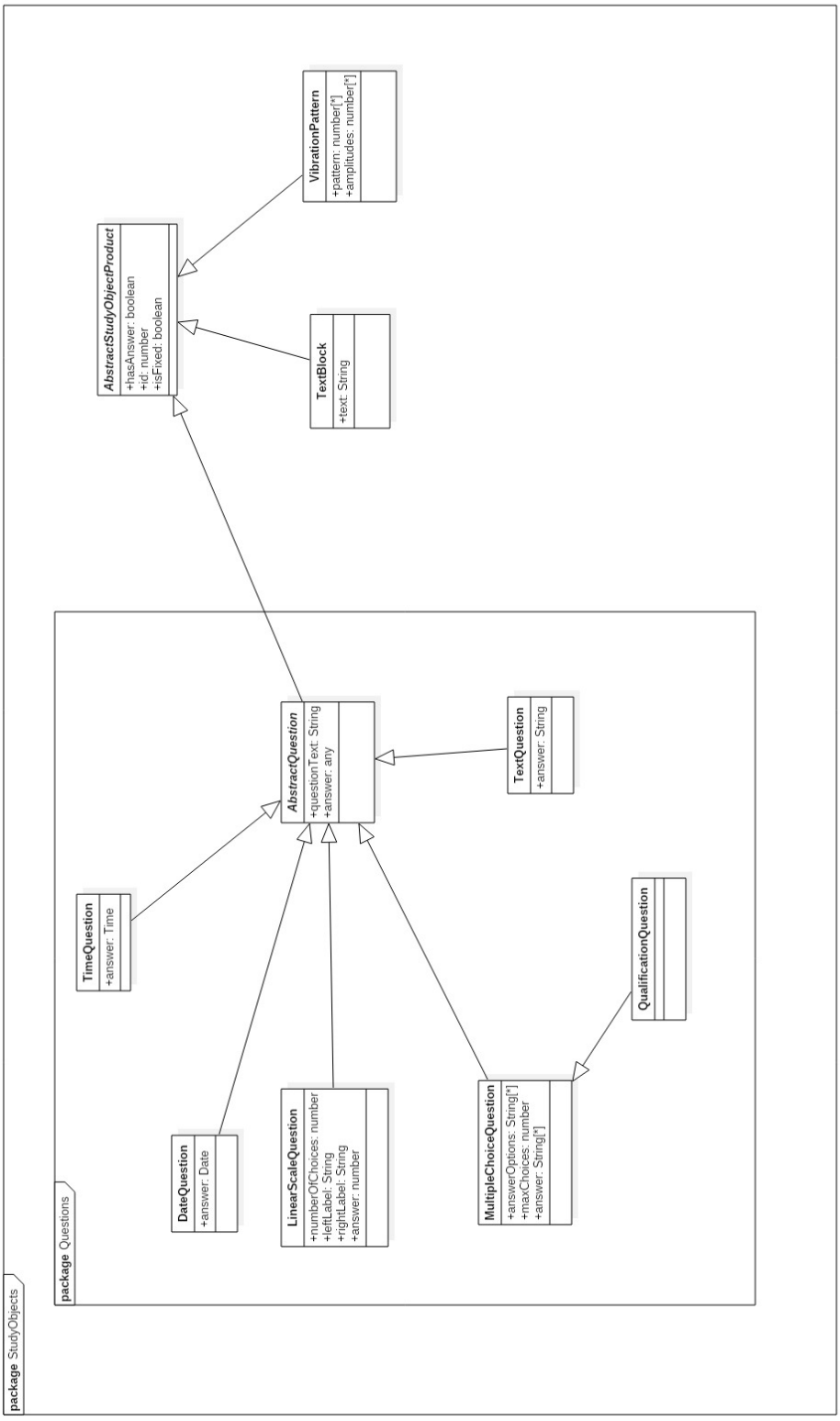

Abbildung 1.3.4: StudyObjects

Model.Study.StudyObjects.AbstractStudyObjectProduct

Die Klasse: public abstract class **AbstractStudyObjectProduct** ist das abstrakte Produkt in der Fabrikmethode.

public boolean **hasAnswer**
Ein Attribut welches anzeigt, ob das konkrete Objekt, das die abstrakte Klasse implementiert, eine Antwort enthält

public boolean **isFixed**
Ein Attribut welches anzeigt, ob das dahinterliegende Element, bei Ausführung der Studie randomisiert oder an seiner Stelle fixiert werden soll.

public number **id**
Ein Attribut, dass unter allen konkreten Objekten, die von AbstractStudyObjectProduct erben, ein eindeutiger Identifier innerhalb einer Studie ist.

Model.Study.StudyObjects.TextBlock extends AbstractStudyObjectProduct

Die Klasse: public class **TextBlock** ist eine Kindklasse, die von AbstractStudyObjectProduct erbt. Sie enthält einen Text, der in einer Nutzerstudie angezeigt wird.

public **text**: String
Ein Attribut, welches einen Text speichert, der dem Nutzer bei Ausführung einer Nutzerstudie angezeigt wird.

Model.Study.StudyObjects.VibrationPattern extends AbstractStudyObjectProduct

Die Klasse: public class **VibrationPattern** ist Kindklasse von AbstractStudyObjectProduct und enthält jene Parameter, die nötig sind, um ein Vibrationsmuster eindeutig zu bestimmen.

public **pattern** : number[*]
Dieses Array enthält Zahlen, die Zeitschlitze darstellen, in denen eine Vibration mit bestimmter Amplitude abgespielt wird.

public **amplitudes** : number[*]
Dieses Array enthält Zahlen, die als Amplitudenstärke interpretiert werden und auf einem bestimmten Zeitschlitz ausgeführt werden.

Model.Study.StudyObjects.Questions.AbstractQuestion extends AbstractStudyObjectProduct

Die Klasse: public abstract **AbstractQuestion** ist eine Kindklasse von AbstractStudyObjectProduct und gibt eine Schnittstelle für konkrete Frageklassen an.

public String **questionText**
Ein Attribut, welches eine Frage speichert.

public any **answer**
Ein Attribut, welches ein beliebiges Objekt als Antwort auf die Frage entgegennehmen kann.

Model.Study.StudyObjects.Questions.DateQuestion extends AbstractQuestion

Die Klasse: public class **DateQuestion** ist eine Kindklasse von AbstractQuestion, die als Antwort ein Datum zurückgibt.

public Date **answer** Ein Attribut, welches als Antwort ein Datum liefert.

Model.Study.StudyObjects.Questions.LinearScaleQuestion extends AbstractQuestion

Die Klasse: public class **LinearScaleQuestion** ist eine Kindklasse von AbstractQuestion, die als Antwort einen diskreten Wert auf einer einstellbaren Skala zurückgibt.

public number **numberOfChoices**
Das Attribut spezifiziert, wieviele diskrete Werte auf der Skala zur Auswahl stehen.

public String **rightLabel**
Das Attribut spezifiziert, welcher Text am rechten Ende der Skala steht.

public String **leftLabel**
Das Attribut spezifiziert, welcher Text am linken Ende der Skala steht.

public number **answer**
Das Attribut enthält die Antwort auf die Frage als Zahl.

Model.Study.StudyObjects.Questions.MultipleChoiceQuestion extends AbstractQuestion

Die Klasse: public class **MultipleChoiceQuestion** ist eine Kindklasse von AbstractQuestion. Die stellt die Möglichkeit zur Verfügung, Fragen mit mehreren Antwortmöglichkeiten zu stellen.

public String[*] **answerOptions**
Dieses Array enthält alle Antwortmöglichkeiten, die angeboten werden.

public number **maxChoices**
Dieses Attribut spezifiziert, wieviele Antworten aus **answerOptions** ansgewählt werden dürfen.

public String[*] **answer**
Dieses Array enthält alle gegebenen Antworten in Textform.

Model.Study.StudyObjects.Questions.QualificationQuestion extends MultipleChoiceQuestion

Die Klasse: public class **QualificationQuestion** gibt die Möglichkeiten, Ausschlussfragen für Studien zustellen.

public **maxChoices** = 1

public **answerOptions** = ["Ja", Nein"]

Model.Study.StudyObjects.Questions.TextQuestion extends AbstractQuestion

Die Klasse: public class **TextQuestion** ist eine Kindklasse von AbstractQuestion und stellt die Möglichkeit zur Verfügung eine Freitextantwort zu geben.

public String **answer**
Dieses Attribut enthält eine Freitextantwort.

Model.Study.StudyObjects.Questions.TimeQuestion extends AbstractQuestion

Die Klasse: public class **TimeQuestion** ist eine Kindklasse von AbstractQuestion und stellt die Möglichkeit zur Verfügung, eine Antwort im Format "HH:MM:SSßu geben.

public **answer** : Time enthält die Antwort in Form des Zeitformats.

1.3.2 Paket Model.User

Dieses Paket modelliert einen Benutzer mit seinen zugehörigen Eigenschaften der Software.

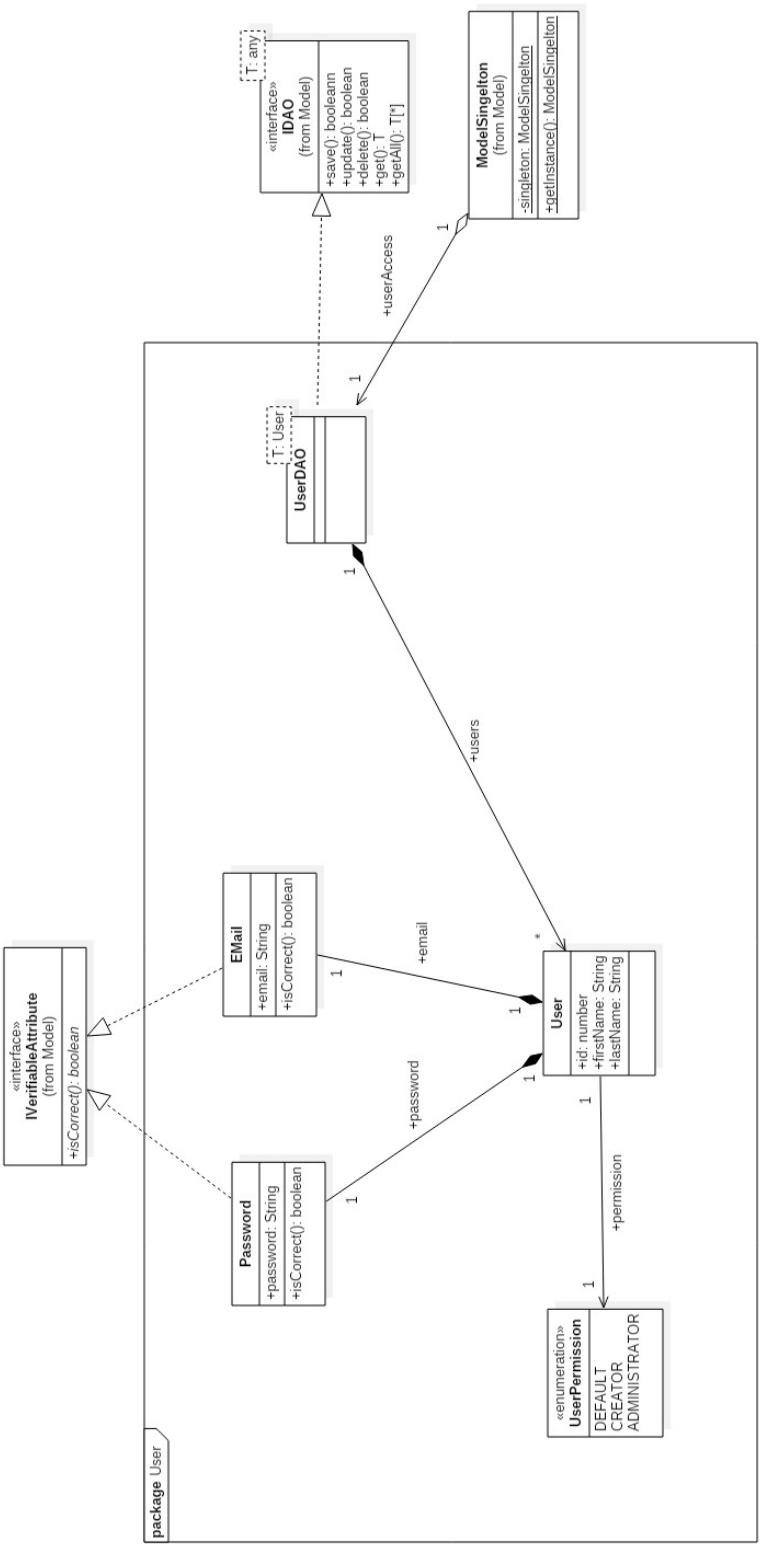

Abbildung 1.3.5: User

Model.User.EMail implements IVerifiableAttribute

Die Klasse public class **EMail** eine e-Mail Adresse dar.

public String **email**
Dieses Attribut enthält die e-Mail Adresse eines User.

public **isCorrect()**: boolean
Diese Methode verifiziert das Attribut email auf syntaktische Korrektheit

Model.User.Password implements IVerifiableAttribute

Die Klasse public class **Password** speichert den Hashwert eines Passwort.

public **password**: String Das Attribut enthält den Hashwert des Passwortes.

public **isCorrect()**: boolean
Diese Methode überprüft, ob das eingegebene Passwort den Sicherheitsstandards entspricht.

Model.User.User

Die Klasse public class **User** speichert einen konkreten Benutzer.

public **id**: number
Das Attribut enthält eine eindeutige Nummer um einen User zu identifizieren.

public **firstName**: String
Das Attribut speichert den Vornamen eines User.

public **lastName**: String
Das Attribut speichert den Nachnamen eines User.

public **email**: EMail
Das Attribut spezifiziert die e-Mail Adresse, die der User besitzt.

public **password**: Password
Das Attribut spezifiziert das Passwort des User.

public **permission** : UserPermission legt fest, welche Zugriffe ein Benutzer hat.

Model.User.UserDAO implements IDAO

Die Klasse public class **UserDAO** verwendet das Entwurfsmuster DAO und bietet eine

Schnittstelle an um auf User zugreifen zu können.

public **users** : User[*] beinhaltet alle registrierten Nutzer. public **save()**: boolean
Diese Methode speichert einen User Eintrag.

public **update()**: boolean
Die Methode updated einen User Eintrag.

public **delete()**: boolean
Die Methode löscht einen User Eintrag.

public **get()**: User
Die Methode gibt einen bestimmten User zurück.

public **getAll()**: List<User>
Die Methode gibt eine Liste aller User (einer Studie) zuück.

Model.User.UserPermission

Das Enum public enum **UserPermission** gibt an, welche Rechte die User in der Application haben.

DEFAULT
Die Standardberechtigung, die automatische jeder User erhält.

CREATOR
Die Berechtigung, Nutzerstudien selbst zu erstellen.

ADMINISTRATOR
Die Berechtigung, über Berechtigungen anderer User zu entscheiden.

1.3.3 Paket Model.Randomizing

Dieses Paket ist für die Randomisierung zuständig.

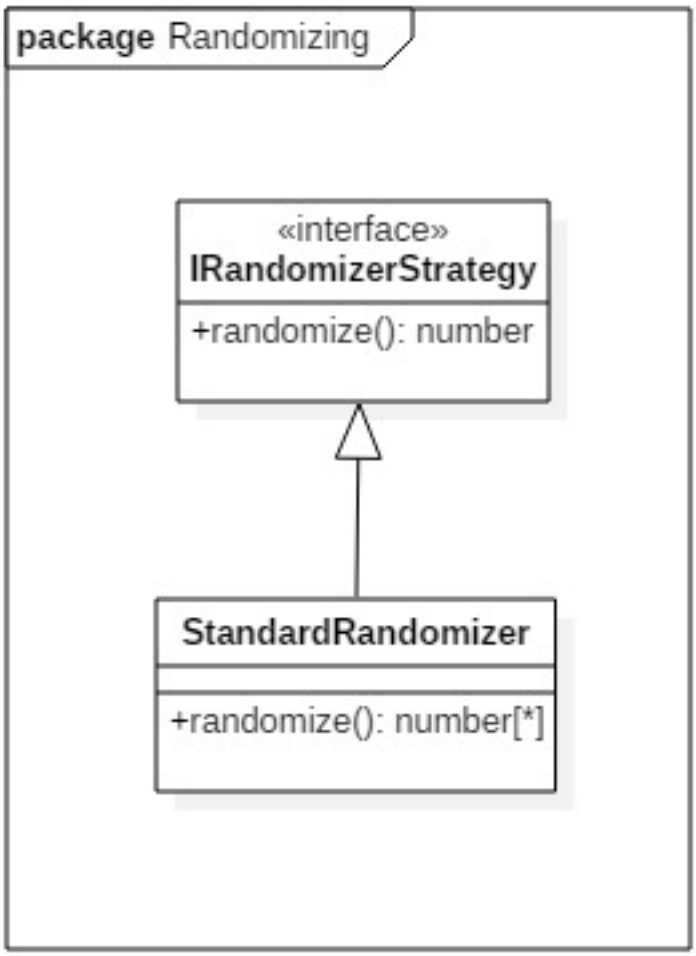

Abbildung 1.3.6: Randomisierung

Model.Randomizing.StandardRandomizer implements IRandomizerStrategy
Die Klasse public class **StandardRandomizer** randomisiert mittles simpler Zufallszahlen, die der Computer mit den Standard Pseudozufallszahlen Algorithmen generiert.

public **randomize()**: number[*]
Die Methode randomisiert über eine Liste in der Zahlen gepeichert sind. Die Zahlen repräsentieren die Reihenfolge der Objekte aus der zu randomisierenden Liste.

Model.Randomizing.IRandomizerStrategy

Diese Klasse stellt ein Interface zur Verfügung, mit dem Elemente aus StudyPrototype und AbstractSectionElement randomisiert werden können. Eine konkrete Randomisierung ist durch die Strategie austauschbar.

public **randomize()**: number[*]
Die Methode randomisiert über eine Liste in der Zahlen gepeichert sind. Die Zahlen repräsentieren die Reihenfolge der Objekte aus der zu randomisierenden Liste.

2 Programmfluss

2.1 Event Studienteilnahme

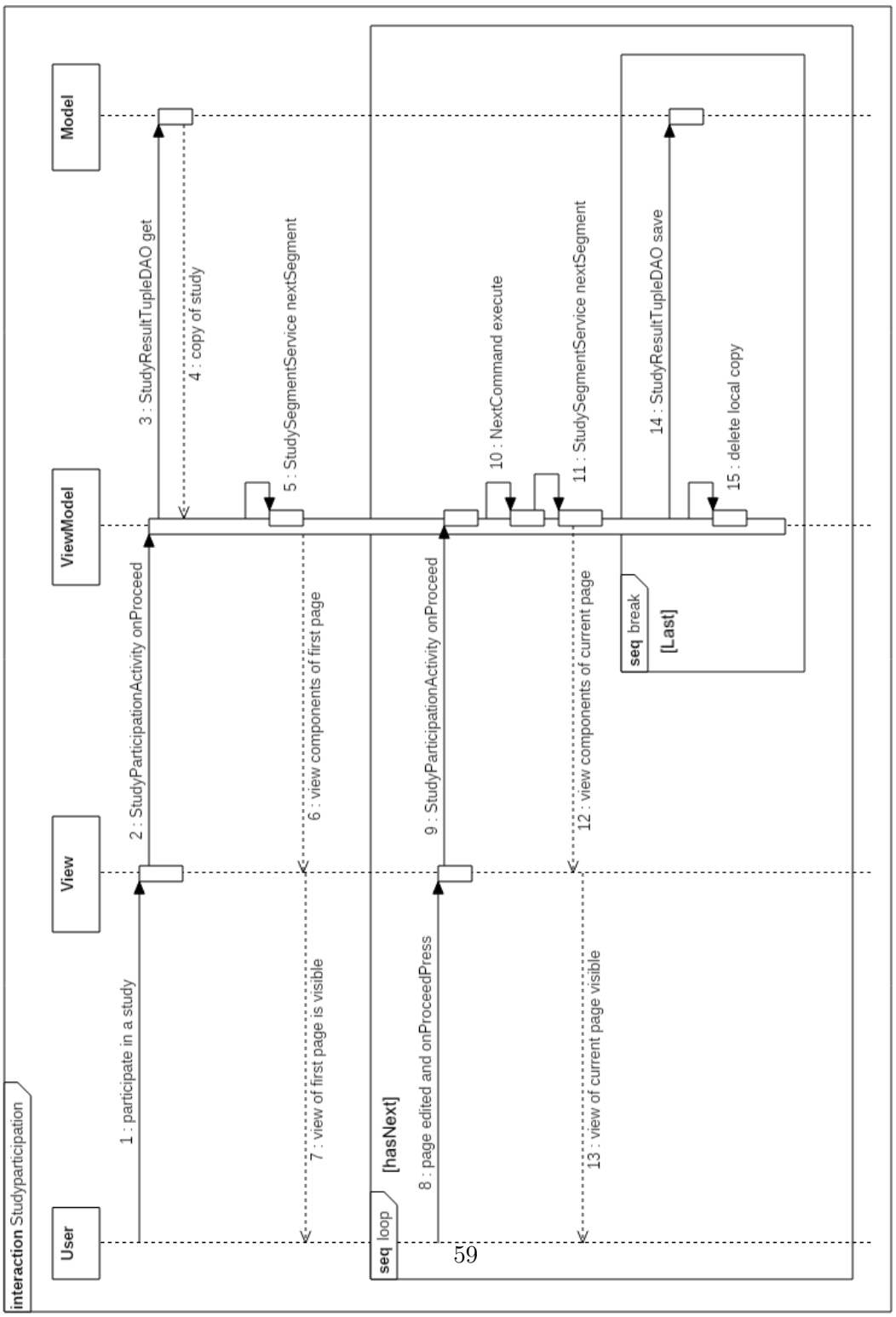

Abbildung 2.1.1: Teilnahme an Studie (keine Demoversion)

2.2 Event Studienerstellung und -bearbeitung

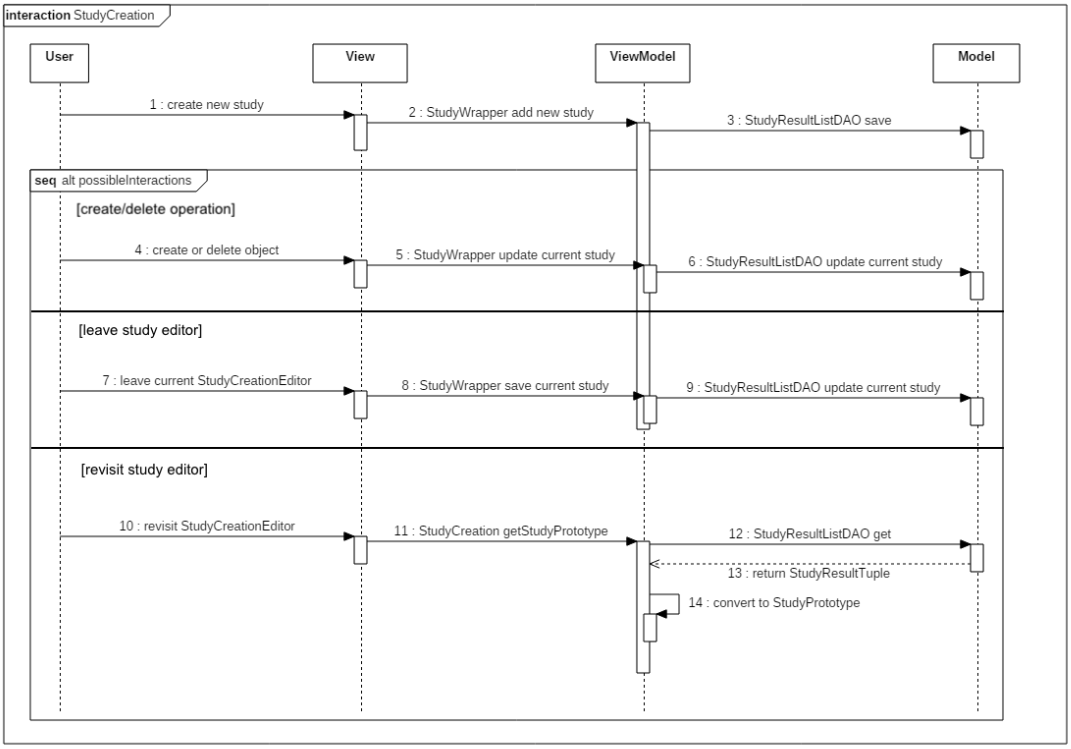

Abbildung 2.2.1: Erstellung und Bearbeitung einer Studie. Anmerkung: bei create/delete operation ist die Erstellung und das Löschen von StudyObjects, AbstractSectionElements und Sections inbegriffen.

3 Navigationsfluss

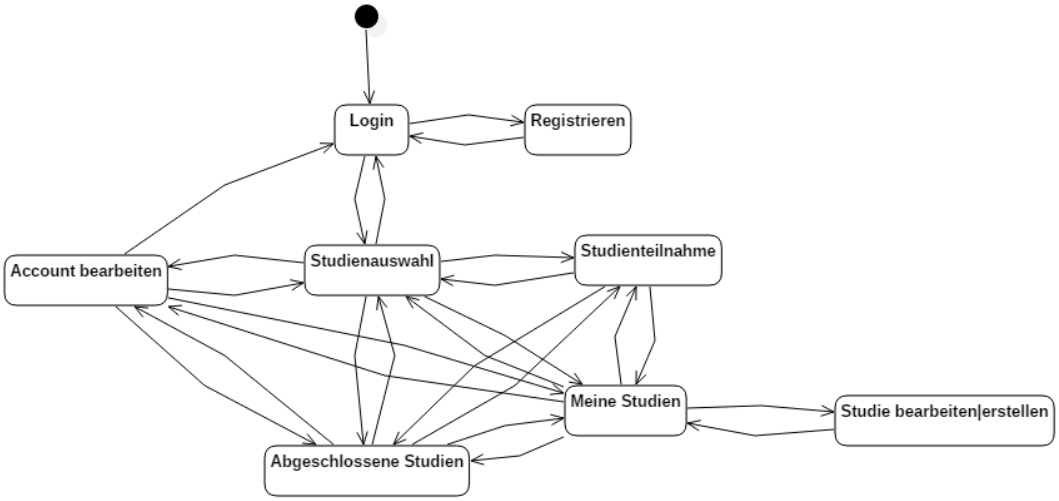

Abbildung 3.0.1: Navigation des Nutzers über die App