Entwurf Earables

Tec O'Brain

16. Dezember 2019

Auftraggeber:	Karlsruhe Institute of Technology (Teco)
Auftragnehmer:	Tec O'Brain (Entwickler: David Höglinger, Jan Ettrich, Erwin Müller, Benedikt Rittner, Valentin Quapil)
Version:	0.1

Inhaltsverzeichnis

1 Einleitung

In diesem Dokument wird der Entwurf der Cross-Platform Bibliothek, des Erweiterungsmoduls und der App spezifiziert. Außerdem wird die Interaktion der einzelnen Komponenten beschrieben. Die verwendete Notation richtet sich nach dem UML-Standard.

2 Aufbau

2.1 Architektur

2.2 Klassendiagramm

3 Klassenübersicht

4 Klassenbeschreibung Model

4.1 ServiceManager

Klassenbeschreibung:

Der ServiceManager verwaltet alle Services des Models mithilfe eines Serviceproviders. Dies wird von anderen Services des Models, sowie von dem Viewmodel benutzt. Er dient zur Delegierung der Services.
Dabei implementiert der ServiceManager das Interface IManager. Von diesem bekommt er die Methode serviceRegistration() übergeben. Der ServiceManager speichert die Services als Referenzen und achtet, dass diese ein Singleton sind. (Es existiert immer nur eine Instanz des Services). Er selbst implementriert das Singleton Muster.

Attribute:

- static instance : ServiceManager	Die Singleton Instanz, welche benutzt wird
+ static Instance : ServiceManager	Regelt die Initialisierung von instance und gibt diese zurück
+ serviceProvider : ServiceProvider	Enthält alle Referenzen auf die Services. Per GetService<T> wird der Service vom Typ T zurückgeliefert.

Methoden:

- Servicemanager() : void	privater Konstruktor, genutzt im Singleton Muster. Aufrufen der Methode serviceRegistration() zur Erstmaligen Registrierung der Services mit einer IServiceCollection und zur Erstellung des ServiceProvider.

4.2 Einstellungen Service

4.2.1 User

Klassenbeschreibung:

Die User-Klasse spezifiziert den Benutzer, der die App gerade verwendet.

Attribute:

+ username: String	Der Name des Benutzers.
+ steplength: String	Die durchschnittliche Schrittlänge des Benutzers in cm.

Methoden:

+ toString(): String	Wandelt das Objekt in einen String um. (vergleichbar mit JSON).
+ static parseUser(user: String): User	Wandelt JSON-String wieder in Objekt um.

4.2.2 ISettingsService

Interfacebeschreibung:

Das Interface ISettingsService bietet eine Schnittstelle für die Einstellungen der App. Sie implementiert Konstanten zur Identifizierung der Einstellungen und hält die aktuellen Einstellungen als Attribute.

Attribute:

+ activeLanguage: CultureInfo	Die aktuelle Sprache der App (Deutsch oder Englisch)
+ samplingRate: SamplingRate	Die aktuelle Samplingrate der Earables
- LANGUAGE_PROPERTY: String	Konstanter Bezeichner für die Einstellung der Sprache
- USER_PROPERTY: String	Konstanter Bezeichner für die Einstellung des Nutzers
- SAMPLINGRATE_PROPERTY: String	Konstanter Bezeichner für die Einstellung der Samplingrate

Methoden:

- loadSettings():void	Lädt alle Settings aus den Einstellungen in die Attribute.

4.2.3 SettingsService

Klassenbeschreibung

Die Klasse SettingsService implementiert die Schnittstelle ISettingsService. In der Implementierung der Funktionen, welche die Speicherung der Einstellungen enthalten, wird die Klasse App.Current.Properties verwendet. Dabei wird immer sofort nach einer Änderung die neue Einstellung gespeichert und nicht erst beim Beenden der Sitzung.

TODO: Sollen auch bei den konkreten Klassen die Methoden/Attribute nochmal aufgelistet werden?

4.3 Datenbank Service

4.3.1 IDataBaseConnection

Interfacebeschreibung

Das Interface IDataBaseConnection ist ein Service, welcher vom ServiceManager verwaltet wird. Dieser regelt Speicherung der Trainingsdaten mittels einer Datenbank. Zudem verwaltet sie das Importieren und Exportieren von den Trainingsdaten. Die Datenbankeinträge sind Instanzen der Klasse DBEntry.

Methoden:

+ saveDBEntry(entry: DBEntry): void	Speichert ein Datenbankeintrag in der Datenbank. Falls schon ein Eintrag zu dem Datum existiert, wird dieser aktualisiert
+ getAllEntriesAsync(): Task<List<DBEntry»	Liefert einem alle Einträge der Datenbank in Form von einer Liste mit DBEntry
+ getMostReentEntriesAsync(amount : int): void	Liefert die letzten amount Einträge der Datenbank in Form einer Liste mit DBEntry zurück
+ importTrainingData(file: FileData): void	Liest aus der angegebenen CSV-Datei die DBEnries raus und speichert sie in der Datenbank.
+ exportTrainingData(path: String): void	Exportiert die Trainingsdaten aus der Datenbank in einer CSV-Datei in den Ordner bei path.

4.3.2 DBEntry

Klassenbeschreibung

Die Klasse DBEntry ist ein Datenkontainer, welcher einen Trainingstag darstellt. Dieser wird in der Datenbank gespeichert. DBEntry beinhaltet kein selbstständiges Verhalten.

Attribute

+ Date: DateTime	Das Datum des Trainingstages. Wird als eindeutiger Bezeichner (Primary key) in der Datenbank benutzt.
+ PushUps: int	Die Anzahl der Liegestützen, welche an dem Tag gemacht wurden
+ SitUps: int	Die Anzahl der SitUps, welche an dem Tag gemacht wurden
+ Steps: int	Die Anzahl der Schritte, welche an dem Tag gemacht wurden

Methoden

+ toString(): String	Liefert die Instanz als String zurück. (CSV-Format)
+ static parseDBEntry(entry: String) : DBEntry	Liest aus einem String ein DBEntry Objekt. Falls der String nicht dem Format entspricht, wird null zurückgegeben.

5 Klassenbeschreibung View-Model

5.1 StepModeViewmodel

Klassenbeschreibung:

Die Klasse CountModeViewmodel enthält die Logik des Stepmodes und hält Attribute die per Databinding an die Viewklassen CountModeView und CountModeActiveView gebunden sind.

Attribute:

+ StepsDoneLastTime: string	Hält die Schrittanzahl des letzten aktiven Laufvorgangs.
+ DistanceWalkedLastTime: string	Hält die gelaufene Distanz während des letzten aktiven Laufvorgangs.
+ LastDatatime: string	Hält das Datum des letzten aktiven Laufvorgangs.
+ StepCounter: int	Counter für die Schrittanzahl während des aktiven Laufvorgangs.
+ DistanceWalked: int	Bisher gelaufene Distanz.
+ isRunning: boolean	True wenn der Nutzer läuft, false wenn er steht.
+ StartActivityCommand: Command	Der Command der beim Drücken des Start Buttons ausgeführt wird, um den Laufvorgang zu starten.
+ StopActivtiyCommand: Command	Der Command der beim Drücken des Stopp Buttons ausgeführt wird, um den Laufvorgang zu stoppen.

Methoden:

+ **«create» StepModeViewmodel()** Konstruktor, in dem die Commands definiert und die Instanz des Service Managers zugewiesen werden. Die Anzahl der zuletzt gelaufenen Schritte und die Distanz sowie das Datum werden durch Aufruf der Methode updateLastData über den ServiceManager von der DatabaseConnection geupdated. isRunning wird standardmäßig zunächst auf false gesetzt.

+ **StartActivity(): void** Methode die vom StartActivityCommand aufgerufen wird. Holt sich beim Service Manager den ActivityManager, der dann die StepActivity zur Verfügung stellt. Das Viewmodel registriert seine OnActivityDone Event Methode beim Event Handler in der StepActivity Klasse und wechselt zum StepModeActiveView.

+ «create» StepModeViewmodel()	Konstruktor, in dem die Commands definiert und die Instanz des Service Managers zugewiesen werden. Die Anzahl der zuletzt gelaufenen Schritte und die Distanz sowie das Datum werden durch Aufruf der Methode updateLastData über den ServiceManager von der DatabaseConnection geupdated. isRunning wird standardmäßig zunächst auf false gesetzt.
+ StartActivity(): void	Methode die vom StartActivityCommand aufgerufen wird. Holt sich beim Service Manager den ActivityManager, der dann die StepActivity zur Verfügung stellt. Das Viewmodel registriert seine OnActivityDone Event Methode beim Event Handler in der StepActivity Klasse und wechselt zum StepModeActiveView.
+ StopActivity(): void	Methode die vom StopActivityCommand aufgerufen wird. Meldet die OnActivityDone Methode vom EventHandler in der StepActivity Klasse ab und setzt isRunning auf false. Zeigt ein PopUp mit der Anzahl der gelaufenen Schritte an, welches der Nutzer wegklicken kann. Danach wird zum StepModeView zurück gewechselt und die Anzeige des letzten Laufvorgangs aktualisiert.
+ OnActivityDone(object sender, EventArgs.Empty)	Event Methode, die von der Aktivität bei Erkennung von einem Schritt aufgerufen wird. Sie erhöht den Counter um 1 und setzt isRunning auf True. Ruft danach checkRunning auf
+ updateLastData(): void	Wird vom Konstruktor und nach jedem Vorgangsende aufgerufen. Aktualisiert die Attribute StepsDoneLastTime, DistanceWalkedLastTime und LastDatatime, indem über den Service Manager von der DataBaseConnection der letzte Eintrag genommen wird.
+ checkRunning(): void	Speichert den aktuellen StepCounter Wert zwischen und vergleicht diesen nach einer Sekunde mit dem neuen StepCounter Wert. Sind diese gleich, so wird isRunning auf false gesetzt.
+ showPopUp(): void	Zeigt das PopUp mit den gelaufenen Schritten und der Distanz asynchron an, wegklickbar mit einem Button.

5.2 CountModeViewmodel

Klassenbeschreibung:

Die Klasse CountModeViewmodel enthält die Logik des Countmodes und hält Attribute die per Databinding an die Viewklassen CountModeView und CountModeActiveView gebunden sind. Sie implementiert die INotifyPropertyChanged Schnittstelle, um das Auswählen einer Aktivität durch den Benutzer in der zugehörigen Viewklasse anzeigen zu können.

Attribute:

+ PropertyChanged: PropertyChangedEventHandler	Der Eventhandler der INotifyPropertyChanged Schnittstelle, bei dem die zugehörige View registriert ist.
+ selectedActivity: String	Die aktuell vom Nutzer ausgewählte Aktivität.
+ timer: Stopwatch	Der Timer, welcher die Trainingszeit des Nutzers misst.
+ counter: int	Ein Zähler, der die Anzahl der ausgeführten Wiederholungen des Nutzers zählt.
+ Minutes: String	Anzahl Minuten der Ausführungszeit.
+ Seconds: String	Anzahl Sekunden der Ausführungszeit.
+ Milliseconds: String	Anzahl Sekunden der Ausführungszeit.
+ StartActivityCommand: Command	Der Command der beim Drücken des Start Buttons ausgeführt wird, um den Zählvorgang zu starten.
+ StopActivityCommand: Command	Der Command der beim Drücken des Stopp Buttons ausgeführt wird, um den Zählvorgang zu stoppen.
+ Manager: ServiceManager	Die Instanz des ServiceManagers, bei dem Anfragen an Services getätigt werden.

Methoden:

+ «create» CountModeViewmodel()	Konstruktor, in dem die Commands definiert und die Instanz des Service Managers zugewiesen werden.
# OnPropertyChanged(): void	Methode die beim Eintritt des PropertyChanged Events aufgerufen wird. Sie verändert bei der registrierten View Klasse die Anzeige der ausgewählten Aktivität.
+ StartActivity(): void	Methode die vom StartActivityCommand aufgerufen wird. Holt sich beim Service Manager den ActivityManager, der dann die auswählte Activity zur Verfügung stellt. Das Viewmodel registriert seine OnActivityDone Event Methode beim Event Handler in der Activity Klasse und wechselt zum CountModeActiveView; danach wird der Timer gestartet.
+ StopActivity(): void	Methode die vom StopActivityCommand aufgerufen wird. Meldet die OnActivityDone Methode vom EventHandler in der Activity Klasse ab und hält den Timer an. Zeigt ein PopUp mit der Anzahl der ausgeführten Aktivität an, welches der Nutzer wegklicken kann. Danach wird zum CountModeView zurück gewechselt.
+ StartTimer(): void	Startet den Timer.
+ StopTimer(): void	Stoppt den Timer.
+ OnActivityDone(): void	Event Methode, die von der Aktivität bei Erkennung der Ausführung von einer Wiederholung aufgerufen wird. Sie erhöht den Counter um 1.
+ ShowPopUp(): void	Zeigt das PopUp mit dem Vorgangsresultat asynchron an, wegklickbar mit einem Button.

5.3 ListenAndPerformViewmodel

Klassenbeschreibung:

Die Klasse ListenAndPerformViewmodel enthält die Logik von Listen&Perform und hält Attribute die per Databinding an die Viewklassen CountModeView und CountModeActiveView gebunden sind. Sie implementiert die INotifyPropertyChanged Schnittstelle, um das Erstellen eines Trainingsablaufplans durch den Benutzer in der zugehörigen Viewklasse anzeigen zu können.

Attribute:

+ PropertyChanged: PropertyChangedEventHandler	Der Eventhandler der INotifyPropertyChanged Schnittstelle, bei dem die zugehörige View registriert ist.
+ StartActivityCommand: Command	Der Command der beim Drücken des Start Buttons ausgeführt wird, um das Training zu starten.
+ StopActivityCommand: Command	Der Command der beim Drücken des Stopp Buttons ausgeführt wird, um das Training zu stoppen.
+ PushUpCounter: int	Zähler für aktuell ausgeführte Liegestütze.
+ SitUpCounter: int	Zähler für aktuell ausgeführte Sit-ups.
+ PushUpResult: int	Zähler für insgesamt während des aktuellen Trainings ausgeführte Liegestütze.
+ SitUpResult: int	Zähler für insgesamt während des aktuellen Trainings ausgeführte Sit-ups.
+ timer: Stopwatch	Der Timer, welcher die Trainingsdauer des Nutzers misst.
+ Minutes: String	Anzahl Minuten der Trainingszeit.
+ Seconds: String	Anzahl Sekunden der Trainingszeit.
+ Milliseconds: String	Anzahl Sekunden der Trainingszeit.
+ ActivityList: ObservableCollection<string>	Liste der Aktivitäten die der Nutzer durchführen möchte, die durch ihn anpassbar ist.
+ ActivityAmounts: ObservableCollection<int>	Anzahl der Aktivitäten die der Nutzer durchführen möchte, die durch ihn anpassbar sind.
+ SelectedActivity: string	Hilfsattribut zum Bearbeiten und Löschen eines Listeneintrags. Per Databinding an die View gebunden; es ist der Eintrag der Liste auf den der Nutzer zuletzt getippt hat.
+ AddActivityCommand: Command	Der Command der beim Drücken des AddActivity Buttons ausgeführt wird.
+ RemoveActivityCommand: Command	Der Command der beim Drücken des RemoveActivity Buttons ausgeführt wird.
+ EditActivityCommand: Command	Der Command der beim Drücken des EditActivity Buttons ausgeführt wird.

Methoden:

+ «create» ListAndPerformViewmodel	Konstruktor, in dem die Commands definiert und die Instanz des Service Managers zugewiesen werden. Die ActityList und ActivityAmounts werden initialisiert.
# OnPropertyChanged(): void	Methode die beim Eintritt des PropertyChanged Events aufgerufen wird. Sie verändert bei der registrierten View Klasse die Anzeige der ausgewählten Aktivitäten sowie deren Anzahl.
+ StartActivity(): void	Methode die vom StartActivityCommand aufgerufen wird, die zur ListenAndPerformActiveView wechselt. Ruft für jeden Listeneintrag in ActivityList nacheinander die DoActivity Methode auf, bis die Liste abgearbeitet ist.
+ StopActivity(): void	Methode die vom StopActivityCommand aufgerufen wird. Hält den Timer an und zeigt ein PopUp mit der Anzahl der ausgeführten Aktivitäten an, welches der Nutzer wegklicken kann. Danach wird zum ListenAndPerform-View zurück gewechselt.
+ startTimer(): void	Startet den Timer.
+ stopTimer(): void	Stoppt den Timer.
+ DoActivity(): void	Holt sich beim Service Manager den ActivityManager, der die entsprechende Activity der ActivityList zur Verfügung stellt, bei dem das Viewmodel seine OnActivity-Done Event Methode registriert. Liest die Activity vor und startet den Timer. Nach Erkennung der vorher festgelegten Anzahl Wiederholungen, meldet das Viewmodel seine OnActivityDone Methode ab und zählt den entsprechenden ResultCounter um die ausgeführte Anzahl hoch.
+ AddActivity(): void	Methode die vom AddActivityCommand aufgerufen wird. Ein Popup erscheint, bei dem der Nutzer auf Liegestütze, Situp oder Pause drücken kann. Nach dem Bestätigen wird erscheint ein neues Popup mit der Anzahl. Die Auswahl wird zu den jeweiligen Listen hinzugefügt.
+ RemoveActivity(): void	Methode die vom RemoveActivityCommand aufgerufen wird. Der gehaltene SelectedActivity Eintrag wird entfernt.
+ EditActivity(): void	Methode die vom EditActivityCommand aufgerufen wird.
+ OnActivityDone(sender: object, EventArgs.Empty)	Event Methode, die von der Aktivität bei Erkennung der Ausführung von einer Wiederholung aufgerufen wird. Sie erhöht den jeweiligen Counter um 1.
+ showPopUp(): void	Zeigt das PopUp mit dem Vorgangsresultat asynchron an, wegklickbar mit einem Button.
+ SpeakNextActivity(): void	Liest die nächste Aktivität der ActivityList sowie seine Anzahl asynchron vor.

6 Klassenbeschreibung View

7 Interaktionsdiagramme

7.1 Aktivitätsdiagramm Lauschen und Agieren

7.2 Sequenzdiagramme

7.2.1 Abläufe in der App

Programmstart Sprache Ändern

8 Entwurfdaten

8.1 Ressourceenverzeichnis

8.2 lokale Datenbank

8.3 App Properties

9 Klassenindex

10 Anhang