Technische Projektdokumentation

ENTWICKLUNG MOBILER ANWENDUNGEN

Prof. Dr. Stephan Kurpjuweit

Autoren:

Daniel Grosmayer
Matrikelnummer:
674695

Darwin Schulz
Matrikelnummer:
674700

Inhalt

Funktion der App

NWUP (Never Wake UP (late again)) ist eine Wecker App die zudem die Funktionen eines Timers, einer Weltuhr sowie einer Stoppuhr beinhaltet.

Weltuhr

Bei dem Tab „Weltuhr“ sieht man in der oberen Hälfte die aktuelle Uhrzeit, den Tag sowie die derzeitige Zeitzone. Klickt man auf das Plus auf der unteren Seite wird man auf eine Aktivität geführt, in der der User eine Zeitzone eingeben kann, welche im Anschluss in der Mitte des Bildschirmes angezeigt wird. Diese Zeitzonen zeigen die ausgewählte Zeitzone, die Uhrzeit sowie das Datum. Durch Swipes entfernt man diese. Alle eingegebenen Zeitzonen werden mithilfe von Shared Preferences abgespeichert.

Alarm

Im Tab „Alarm“ kann der User mithilfe des Plus Knopfes in der Ecke unten-rechts einen Alarm erstellen, hierfür wird man zu einem Timepicker geleitet wo der User die Zeit, den Intervall und den Titel des Alarms eingeben kann. Klickt der Benutzer auf den „Alarm erstellen“ Knopf, wird der User zurückgeleitet und findet alle erstellten Alarme in der Oberen Hälfte des Bildschirms, dort kann er den Alarm zudem ausschalten oder löschen. Der Müllkorb-Button auf der rechten Seite löscht alle erstellten Alarme. Alarme werden mithilfe einer Room Datenbank abgespeichert und geladen.

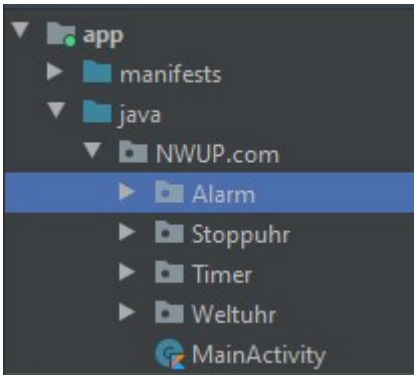
Stoppuhr

Bei der „Stoppuhr“ sieht der User auf der oberen Seite die derzeit abgelaufene Zeit. Durch den Play Button wird die Stoppuhr gestartet und die Zeit läuft hoch. Durch den Flaggen Knopf auf der rechten Seite lassen sich Zwischenergebnisse speichern. Durch den Pause Knopf wird die Zeit pausiert und der Flaggen Knopf wird zu einem Share-Button wodurch die Ergebnisse weitergeschickt werden können. Durch den Button auf der linken Seite lassen sich alle Zwischenergebnisse sowie die abgelaufene Zeit resetten.

Timer

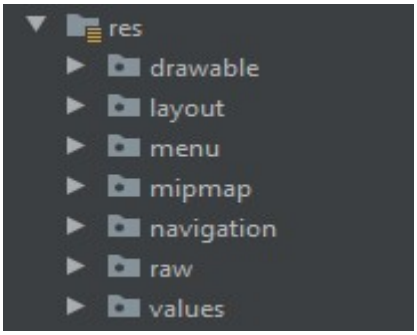
Im letzten Tab „Timer“ kann der User die Zeit einstellen, wenn er auf die Zahl in der Mitte des Bildschirms drückt, hierbei können die Minuten sowie die Sekunden eingestellt werden und diese wird im Anschluss in der Mitte angezeigt. Wird auf den Play Button auf der linken Seite gedrückt, startet der Timer und die Zeit läuft ab. Durch den Knopf in der Mitte wird der Timer pausiert. Der Button auf der rechten Seite setzt den Timer wieder auf die ursprünglich eingegebene Zeit.

Aufbau der App



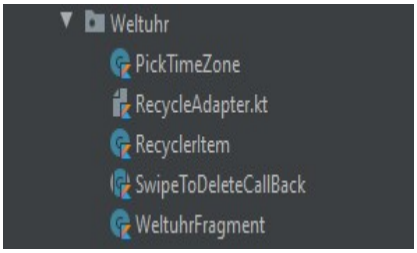
Unsere App enthält die 4 Ordner „Alarm“, „Stoppuhr“, „Timer“ sowie „Weltuhr“ in jedem Ordner befinden sich alle nötigen Dateien, um die zuvor beschriebenen Tabs laufen zu lassen. Alle 4 Tabs sind Fragmente, die durch die Auswahl der Bottomnavigation der Mainactivity geladen werden. Als Default beim Start wird die Weltuhr geladen.

Mainactivity lädt im Grunde nur die verschiedenen Fragmente, wenn auf die Bottomnavigation gedrückt wird.



In „drawable“ befinden sich alle unsere verwendeten Bilder. In „layout“ befinden sich unsere Fragmente, Activities sowie der Toolbar. „Menu“ enthält unsere „Bottom_nav“. „Mipmap“ ist der standardmäßig angelegte Ordner von Android Studio in der sich die Icons einer Android-App befinden. In „navigation“ liegt der „nav_graph“ welcher in Alarm benutzt wird, um zwischen den verschiedenen Fragmenten zu wechseln. Der „raw“ Ordner enthält unseren erstellten Alarm Ton. Und zuletzt in „values“ befinden sich unsere Styles, Farben und Strings.

Weltuhr



PickTimeZone ist die Aktivität, die aufgerufen wird, wenn auf das Plus gedrückt wird und enthält das Laden einer autotextView, dass alle vorhandenen Zeitzonen angezeigt werden und besteht zudem aus einem ItemClickListener, welcher die ausgewählte Zeitzone in den Recyclerview im Fragment hinzufügt.

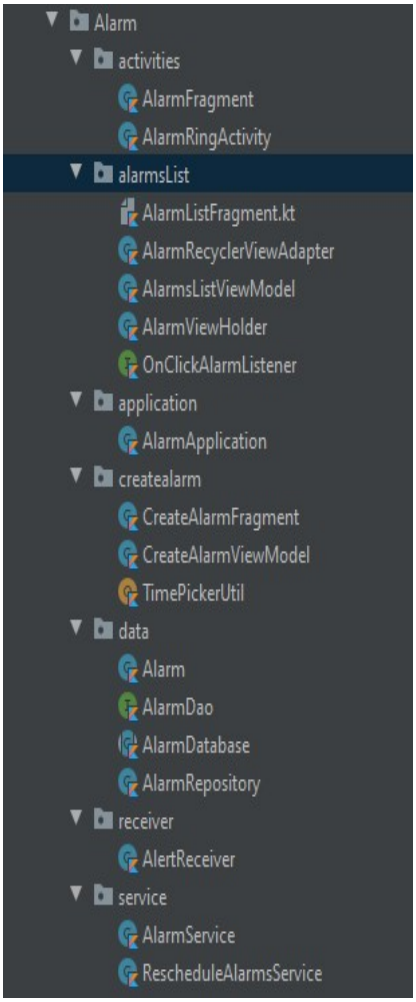
RecycleAdapter.kt beinhaltet den RecyclerAdapter der Weltuhr.

RecyclerItem ist eine Data Klasse, welche als Item für den RecycleAdapter dient.

SwipeToDeleteCallBack ist eine Abstrakte Klasse, welche im WeltuhrFragment erstellt wird, damit die Items im RecyclerView durch ein Swipe gelöscht werden.

WeltuhrFragment beinhaltet das Fragment, welches aufgerufen wird, wenn auf der Navigation „Weltuhr“ gedrückt wird. Hierbei wird die derzeitige Zeit sowie Zeitzone gelesen und angezeigt, dies findet durch die Funktion „setCurrentDate“ statt, welche in „onStart“ aufgerufen wird. Weiterhin beinhaltet das Fragment einen „setOnClickListener“ in „onCreateView“ welcher beim Knopfdruck auf das Plus die Aktivität „PickTimeZone“ startet. Bei „onAttach“ werden die Daten von den Shared Preferences gelesen, dies findet in der Funkton „loadData“ statt, welche nach der SharedPreference „weltuhr preferences“ sucht und die Daten im companion object „RecyclerItems“ einträgt. Nach „setCurrentDate“ wird zudem der Adapter sowie das LinearLayoutManager gesetzt und das Objekt „SwipeToDeleteCallBack“ erstellt, welches im Adapter die Löschfunktion aufruft und die Daten im Anschluss speichert.

Alarm



Der Alarm beinhaltet die im Bilde angezeigten Dateien und Ordner. Hierbei steht jeder Ordner für eine Funktionalität.

„activites“ beinhaltet alle Aktivitäten bzw. Fragmente, die beim Alarm aufgerufen werden.

„alarmsList“ beinhaltet alle Klassen, die dafür da sind, dass alle Alarme in einem Fragment angezeigt werden können.

„application“ beinhaltet eine Application welche den NotificationChannel beim Start erstellt

„createAlarm“ beinhaltet alle Klassen, damit man einen Alarm in einem Fragment erstellen kann

„data“ beinhaltet eine Klasse welche einen Alarm darstellt (Alarm), ein Interface mit den Befehlen (AlarmDao), eine Instanz von der Datenbank (AlarmDatabase) sowie eine Klasse die auf die Datenbank zugreift (AlarmRepository).

„receiver“ enthält die Klasse die Notifications annimmt und daraufhin weiterleitet

„service“ beinhaltet die Klassen, welche den empfangenen Alarm verarbeiten

AlarmFragment ruft das Layout „fragment_alarm_main“ auf welches ein Fragment beinhaltet, welches mit einer navigation verbunden ist. Dadurch können wir durch verschiedene Fragmente wechseln

AlarmRingActivity ist die Aktivität, die aufgerufen wird, wenn der Alarm angekommen ist und man den Wecker ausschalten will. Dies beinhaltet Zwei onClickListener, einer zum Ausschalten, hierbei wird ein neuer Intent erstellt und weitergeleitet wird, sodass der Alarm gestoppt wird. Der andere Listener ist dafür da, dass wenn auf „Snooze“ gedrückt wird, ein neuer Alarm weitergeleitet wird, der 10 minuten später losgeht.

AlarmListFragment ist das Fragment, welches alle Alarme anzeigt. Hierbei werden in „onCreateView“ 2 onClickListener erzeugt, „deleteAllButton“ löscht mithilfe der Funktion „onDeleteAll“ alle Alarme aus der Datenbank, „addAlarm“ löst den Navigator aus, sodass zum CreateAlarmFragment gewechselt wird. Weiterhin werden hier die 2 Funktionen „onToggle“ und „onItemDelete“ überschrieben, welche dazu dienen, dass man auf die Items im RecyclerView selbst zugreifen kann und die Daten somit ändert. „onToggle“ wechselt von den Zuständen an und aus und startet einen Alarm oder bricht diesen ab. „onItemDelete“ und „onDeleteAll“ müssen beide auf die Daten zugreifen, damit der Mainthread dies nicht tun muss, wird ein lifecycleScope gestartet, der auf die Datenbank zugreift, den Zustand der Alarme herausfindet, diese Alarme abbricht, wenn sie gestartet haben und diese anschließend von der Datenbank löscht.

AlarmRecyclerViewAdapter beinhaltet den RecyclerAdapter des AlarmListFragment

AlarmsListViewModel ist die Klasse, welche die LiveData abruft und der Zwischenweg, um mit der Datenbank zu kommunizieren.

AlarmViewholder entnimmt Alarme aus der Datenbank und setzt die UI anhand dieser um und verwaltet die Listener für die Listitems mit den zuvor genannten Funktionen „onToggle“ und „onItemDelete“.

OnClickAlarmListener ist das Interface für die beiden Funktionen des AlarmViewHolder.

AlarmApplication erstellt den NotificationChannel beim Start der App mit der Channel ID „ALARM_SERVICE_CHANNEL“

CreateAlarmFragment ist das Fragment, wo der User einen Alarm erstellt. Hierbei wird ein Alarm Objekt mit allen angegebenen Informationen erstellt (Zeit durch den Timepicker, Wochentage durch die Checkboxen) und in die Datenbank eingetragen. Weiterhin wechselt das Fragment zurück zum AlarmListFragment wenn der Alarm erstellt wird.

CreateAlarmViewModel ist der Zwischenweg, damit CreateAlarmFragment mit der Datenbank kommunizieren kann.

TimePickerUtil ist das Objekt, welches die eingegebene Zeit im Timepicker zurückgibt.

Alarm ist die Klasse, welche einen Alarm darstellt und in der Datenbank abgespeichert wird. Ein Alarm besteht hierbei aus der AlarmId, seinem Titel, der Zeit und den Wochentagen wo der Alarm losgehen soll. In der Funktion „schedule“ wird nun aus den genannten Informationen ein Intent erstellt und in einen alarmManager gepackt sowie eine Toastmessage angezeigt wird, für wann der Alarm angelegt wurde. „cancelAlarm“ erstellt einen Intent zum Beenden des Alarms und schickt eine Toastmessage welchen Alarm der User abgebrochen hat. „getRecurringDaysText“ gibt einen String zurück, der angibt für welche Tage der Alarm gesetzt wurde, um die Toastmessage besser zu gestalten.

AlarmDao enthält die Befehle, um Daten aus der Datenbank zu entnehmen/einzufügen.

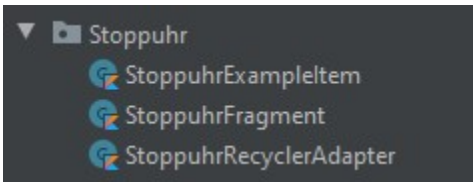
AlarmDatabase erstellt den „databaseWriteExecutor“, um Datenbankbefehle auszuführen und gibt die Datenbank zurück in der Funktion „getDatabase“, falls keine existiert wird eine erstellt.

AlarmRepository enthält alle Funktionen, die die Befehle in der AlarmDao ausführen.

AlertReceiver ist der BroadcastReceiver, welcher Notifications entgegennimmt und abhängig von diesen den Alarm verarbeitet. „onReceive“ schaut nach ob der Intent ein Boot des Handys ist und plant den Alarm neu mithilfe von „startRescheduleAlarmService“ oder startet einen neuen Alarm und schickt einen neuen Intent mit AlarmService.

AlarmService startet die Aktivität „AlarmRingActivity“ und lässt das Handy klingeln.

Stoppuhr

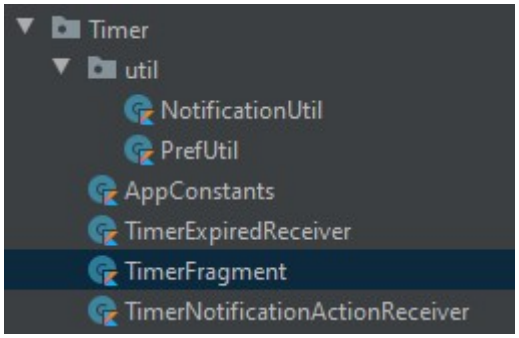


StoppuhrExampleItem ist eine Data Klasse, welche als Item für den StoppuhrRecyclerAdapter dient.

StoppuhrFragment beinhaltet das Fragment, welches aufgerufen wird, wenn auf der Navigation „Stoppuhr“ gedrückt wird. Hierbei gibt es 4 „setOnClickListener“. „stopwatchStart“ lässt die ablaufen und zeigt dem User die beiden Knöpfe für Stop und Lap an. „stopwatchReset“ setzt die abgelaufene Zeit zurück auf 0, versteckt die Knöpfe links und rechts vom Pause Knopf und setzt diesen zurück zum Playbutton. „stopwatchShare“ ist der Share Knopf auf den rechten Seiten des Play Knopfes und gibt einem die Möglichkeit die gestoppte Zeit an weitere Personen zu schicken. „stopwatchLap“ speichert die vergangene Zeit in ein Recyclerview, sodass die Zeit öfters gestoppt werden kann.

StoppuhrRecyclerAdapter beinhaltet den Adapter für das StoppuhrFragment und setzt die Laps in ein RecyclerView.

Timer



In NotificationUtil werden die jeweiligen Notifications gebaut je nach Zustand des Timers z.B., wenn der Timer auserhalb der App läuft, wird die Meldung „showTimerExpired“ angezeigt.

PrefUtil besteht aus Funktionen, welche die Werte aus den Shared Preferences ablesen und setzen.

AppConstants beinhaltet die vier Konstanten (ACTION_STOP, ACTION_PAUSE, ACTION_RESUME und ACTION_START).

TimerExpiredReceiver wird aufgerufen, wenn der Timer abgelaufen ist und zeigt die Notification das der Timer abgelaufen ist und setzt den Zustand des Timers auf „Stopped“ und setzt die Zeit auf 0.

TimerFragment ist das Fragment, welches den Timer anzeigt, hierbei gibt es drei Knöpfe und ein Klickbares Textview. Wenn auf das Textview geklickt wird, öffnet sich ein Popup indem der User die Minuten und Sekunden des Timers eingeben kann. Der mittlere Knopf startet den Timer, der linke pausiert ihn und der rechte setzt den Timer auf die zuvor eingegebene Zeit zurück.

TimerNotificationActionReceiver beinhaltet die Funktion, damit die Buttons auf der Notifciaton ausgeführt werden. Dazu gehören der „Stop“, „Pause“, „Resume“ und „Start“.

Erweiterungsmöglichkeiten

Allgemeine Erweiterungen

- Lightmode
- Barrierefreiheit für Menschen mit Behinderung

Erweiterungen für den Alarm

- Klingelton auswählen
- Verschiedene Weckoptionen (z.B. Matheaufgaben beim Klingeln)
- Swipe to delete für die Alarme
- Vibration auswählen
- Lautstärke schrittweise erhöhen
- Individuelle „Snooze“ Länge einstellen

Erweiterungen für die Weltuhr

- Spezifische Suche nach Städten
- Spezifische Suche nach Ländern
- Temperatur von den Städten
- Wetter von den Städten anzeigen

Erweiterung für den Timer

- Stunden hinzufügen
- Mehrere Timer
- Notifikationston einstellen
- Lautstärke einstellen
- Vibration
- Lautstärke schrittweise erhöhen

Zeiterfassung

Daniel

Daniel	Tätigkeit	Dauer (in min)	Datum(Fertigstellung)
	Projektskizze	120	06.04.2020
	Erstellung des Alarms	1300	30.08.2020
	Erstellung der Weltuhr	340	30.08.2020
	Hilfestellung bei Timer	120	30.08.2020
	Technische Doku	120	30.08.2020
	Refactoring	120	30.08.2020
	Ingesamte Stunden	35,3	Stunden

Darwin

Darwin	Tätigkeit	Dauer (in min)	Datum (Fertigstellung)
	Projektskizze	120	06.04.2020
	Einrichtung des Projekts in Github	60	08.04.2020
	Erstellung der Stoppuhr	300	30.08.2020
	Erstellung des Timer	1200	30.08.2020
	Design des Alarms + Pairprogramming	360	30.08.2020
	Technische Doku	120	30.08.2020
	Refactoring	120	30.08.2020
	Erstellung der Demo	30	30.08.2020
	Ingesamte Stunden	38,5	Stunden