Universität Stuttgart

Studienprojekt **GarmentOS** der Abteilung für Mensch-Computer-Interaktion

Entwurf

Kunde:
Prof. Dr. Albrecht Schmidt

Betreuer:
Jun.-Prof. Dr. Niels Henze
M. Sc. Stefan Schneegass
Dipl.-Inf. Markus Funk

Autoren:
Tamara Müller, Lucas Röhrle, Tobias Linn, Sophie Ogando, Ferdinand Pfähler, Velihan Bulut, Martin Root, Oliver Röhrdanz, Vincenz Pauly, Manuel Lorenz

28. April 2015

Inhaltsverzeichnis

1 Einleitung

1.1 Zweck des Dokuments

Der in diesem Dokument festgelegte Entwurf für das Projekt GarmentOS soll den Entwicklern eine schematische Zusammensetzung liefern, welche als Leitfaden hinsichtlich der Implementierung der Software dienen soll. Es sind die essentiellen Entschlüsse bzgl. der Zusammensetzung der Software vermerkt.
Durch UML-Diagramme werden einzelne Grundbausteine des vollständigen Systems ausführlich dargestellt. Durch dieses Dokument soll den Entwicklern die Wartbarkeit und Erweiterbarkeit in der Zukunft leichter gemacht werden.
Die Betreuer erhalten durch dieses Dokument ein besseren Gesamtüberblick über die Software.

1.2 Leserkreis

Zum Leserkreis gehören:

- die Betreuer
- das Entwicklungsteam des Studienprojekts GarmentOS

Der Leserkreis kann in Zukunft durch Personen ausgedehnt werden, die für die Weiterentwicklung durch den Kunden eingesetzt werden.

1.3 Überblick zum Entwurf

Zu Beginn werden die generellen Entscheidungen für den Entwurf, die angewandten Entwurfs- und Architekturmuster ausgeführt. Dadurch soll eine Übersicht über die Zusammensetzung und Vorgehensweise des Entwurfs bereitgestellt werden.

1.4 Referenzierte Dokumente

Dieses Dokument referenziert die Spezifikation des Projekts GarmentOS.

1.5 Konventionen

Durch die agile Entwicklung in kleinen Teams legt sich das gesamte Entwicklerteam selbst strikte Konventionen auf. Bei der Entwicklung des Eclipse-Plugins ist die Orientierung an den Eclipse UI Guidelines. Die gesamte Entwicklung sieht vor strikt die Java Code Conventions anzuwenden.

2 Sensormodul

2.1 Package: sensors

2.1.1 Klassendiagramm

2.1.2 Beschreibung

Sensor:
Verwaltet die Parameter eines realen Geräte-internen oder -externen Sensors und symbolisiert diesen in der API. Speichert die aktuellsten Werte des Sensors im Hauptspeicher und lagert ältere Werte in das Storage Modul aus.

SensorType:
Hält alle vom System unterstützten Sensortypen (z.B. Heartrate, Accelerometer, Gps-Sensor...)und ordnet diesen ein MeasurementSystem, eine standardmäßige Dimension der erwarteten Werte und ein Icon zu.

SensorData:
Speichert die Werte aller Dimensionen eines Sensorwerts und den dazu gehörenden Zeitstempel, wann diese Werte aufgenommen wurden.

SensorManager:
Verwaltet eine Liste aller aktiven und inaktiven Sensoren.

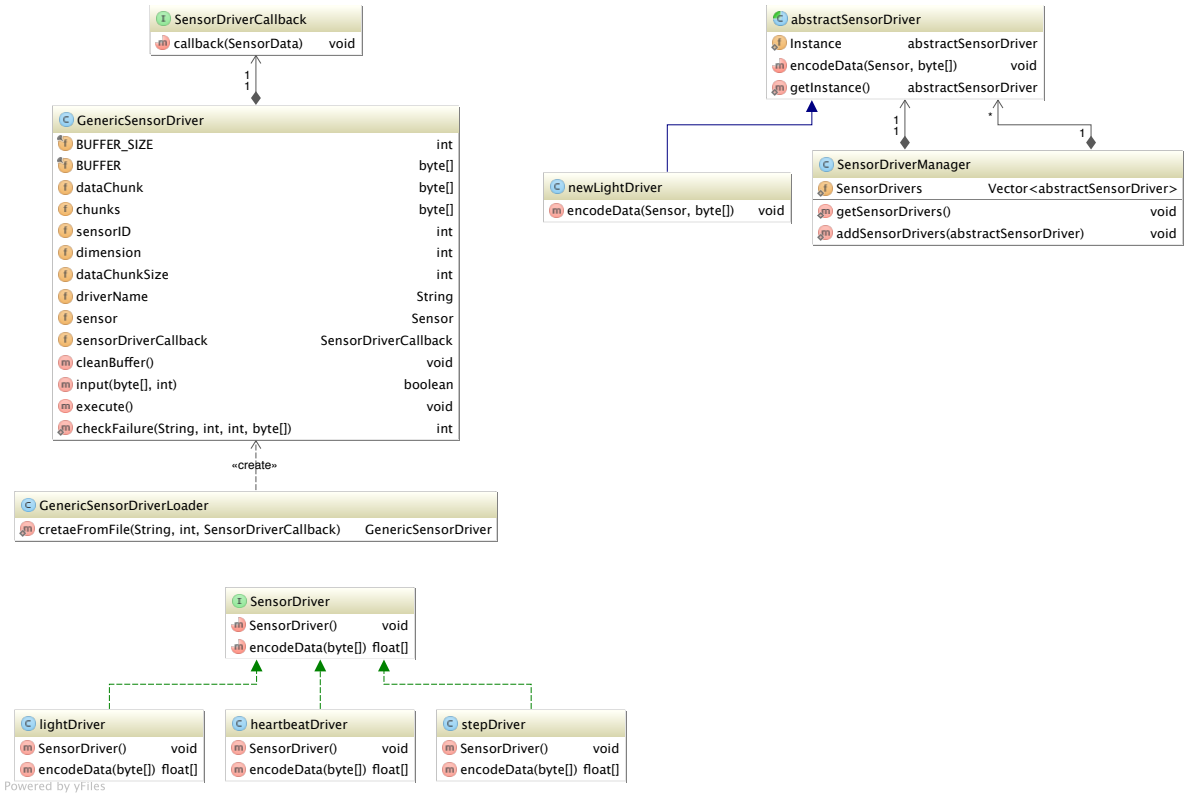
InternalSensors:
Stellt eine Schnittstelle zu den Gerät internen Android Sensoren dar. Für jeden von dem Gerät unterstützten Android Sensor wird ein Sensor Objekt angelegt.

MeasurementSystems:
Hält alle vom System unterstützten Einheiten (z.B. Lux, Pascal, Hertz...), in denen Sensorwerte angenommen oder angezeigt werden können.

MeasurementUnits:
Hält alle vom System unterstützten Multiplizitäten (z.B. Kilo, Mega, Hecto...), ordnet diesen MeasurementSystems zu, für die sie zulässig sind, und speichert einen Umrechnungsweg in die Grundeinheit.

2.2 Package: sensorDriver

2.2.1 Klassendiagramm



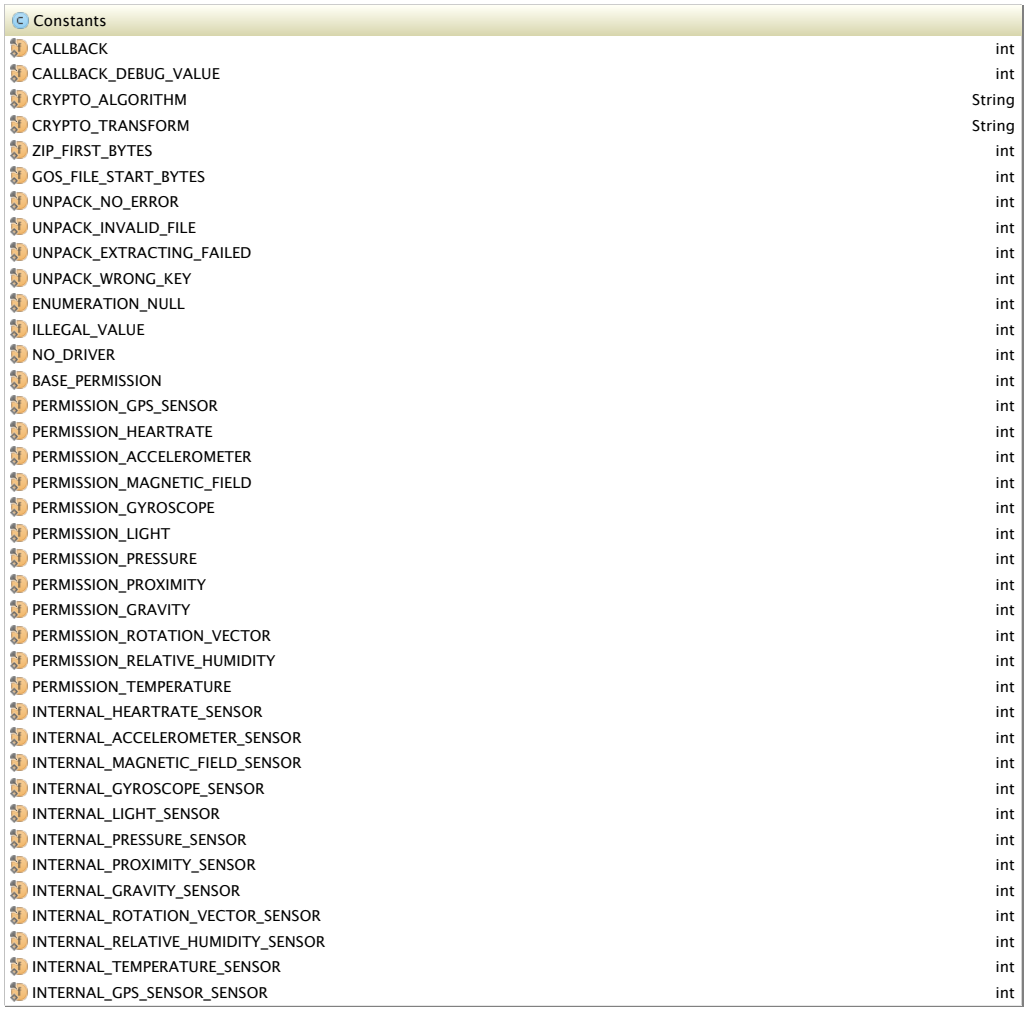
2.2.2 Beschreibung

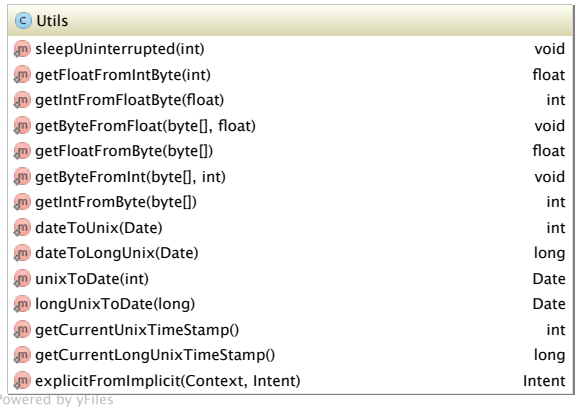
Dieses Package verwaltet die Treiber der zu verbindenen Sensoren.

3 Verarbeitungsmodul

3.1 Package: utils

3.1.1 Klassendiagramm





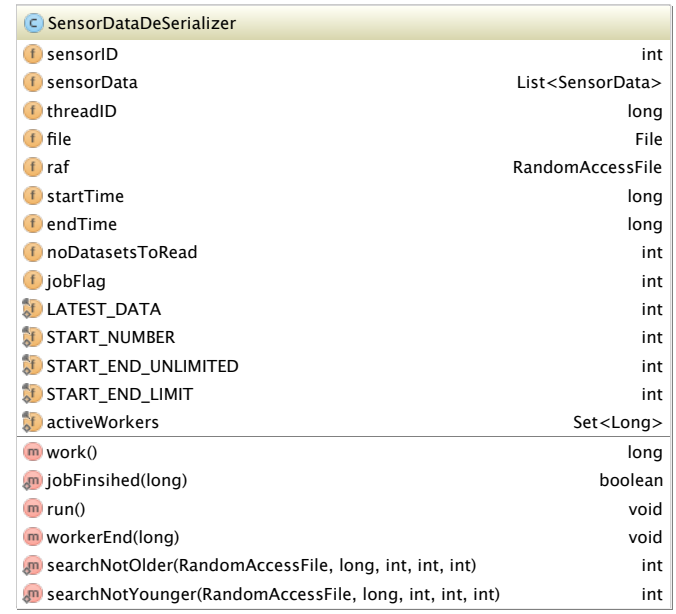
3.1.2 Beschreibung

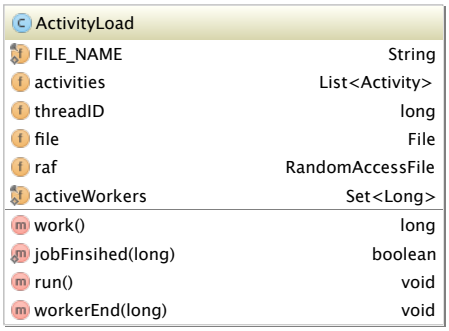
Dieses Paket stellt Hilfsfunktionen bereit, die in verschiedenen Paketen eingesetzt werden.

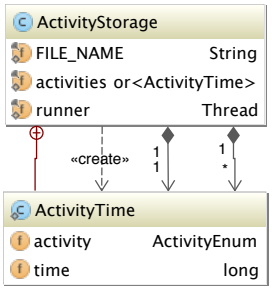
4 Speicher- und Verschlüsselungsmodul

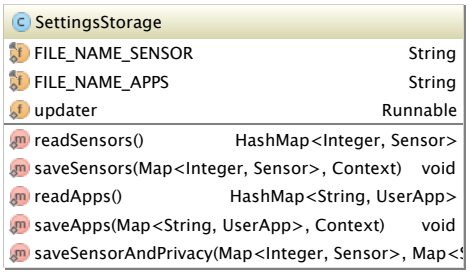
4.1 Package: storage

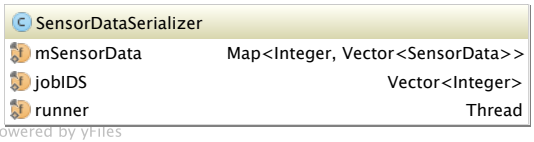
4.1.1 Klassendiagramm









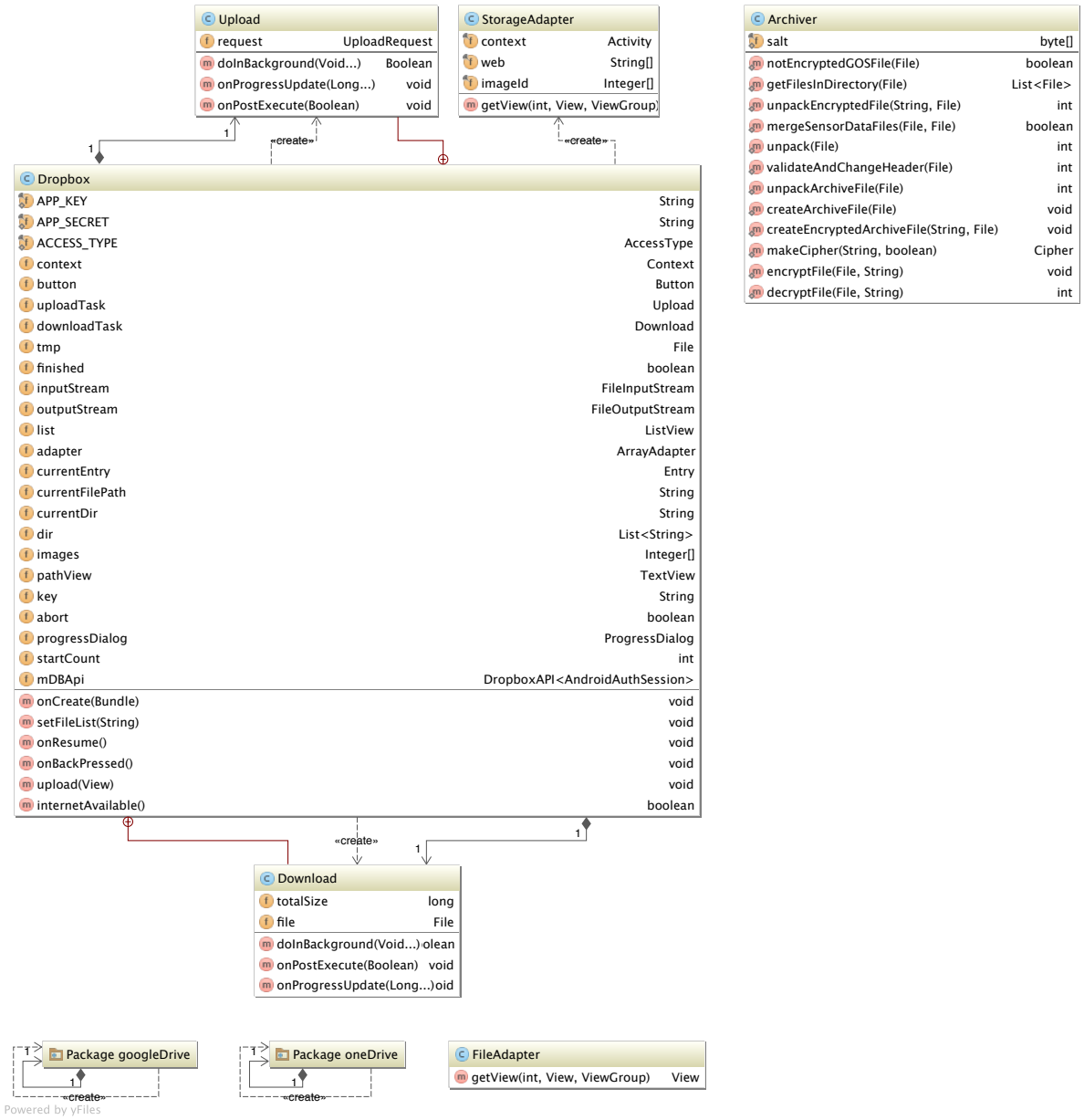


4.1.2 Beschreibung

Das Storage Paket dient zum Speichern und Laden von Einstellungen, Aktivitäten und Sensordaten.

4.2 Package: cloud

4.2.1 Klassendiagramm



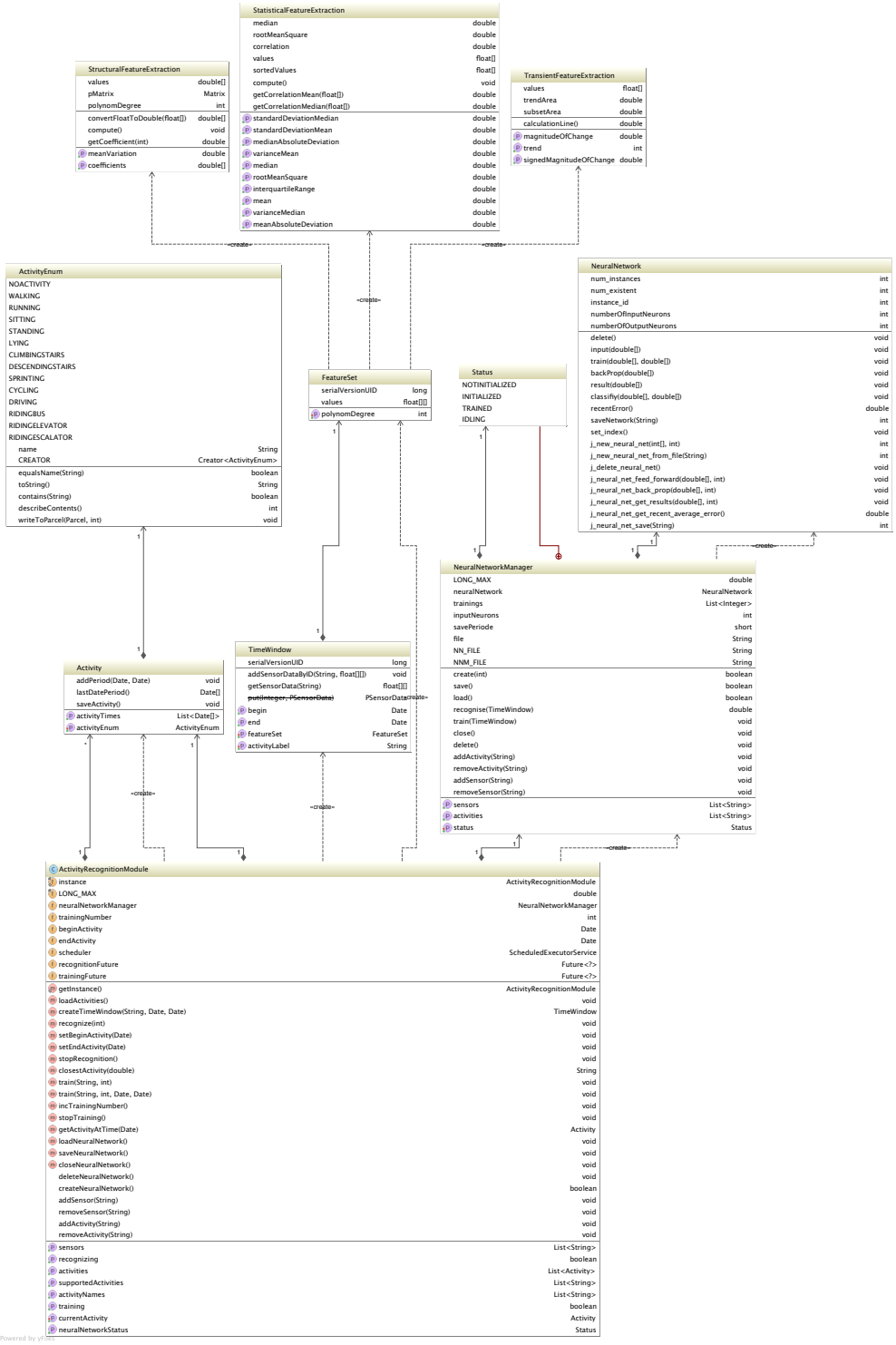
4.2.2 Beschreibung

Bietet Implementierungen für den Upload von selbsterstellten Archiven mit Sicherungsdateien der aktuell lokal gehaltenen Sensordaten.
Zudem ist es für den Download und anschließenden Import von Archivdateien mit zuvor gesicherten Sensordaten zuständig, die bei den bekannten Cloud-Services Google Drive, DropBox oder One Drive gelagert wurden.

5 Activity Recognition Modul

5.1 Package: activityRecognition

5.1.1 Klassendiagramm



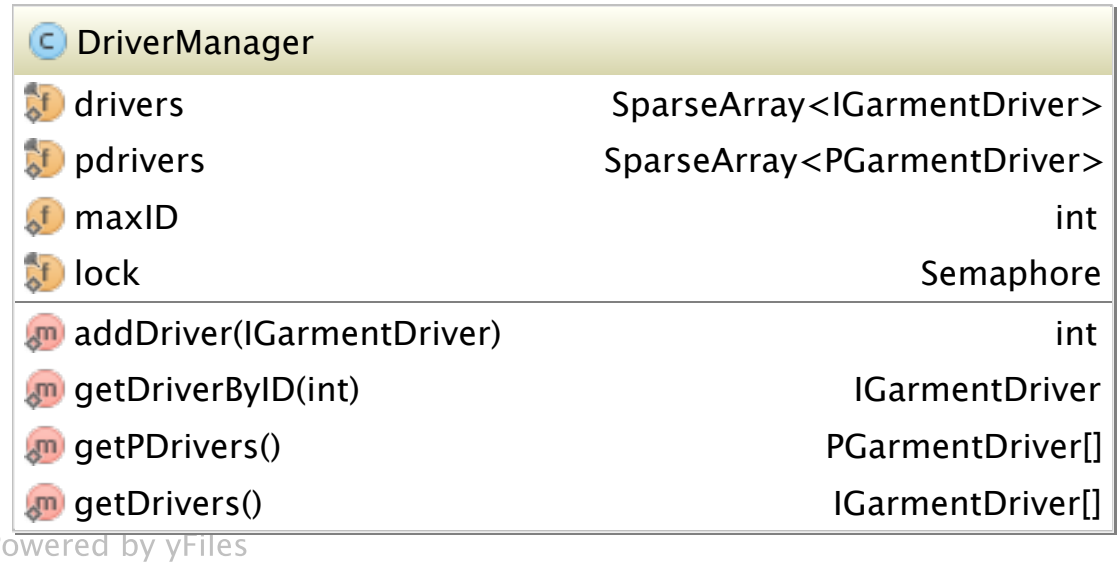
5.1.2 Beschreibung

Das „Activity Recognition Module“ übernimmt die Aufgabe, Aktivitäten zu erkennen mit allen dazu nötigen Mitteln. Hierbei dient das Modul selber als Schnittstelle, um mit dem Neuronalen Netzwerk zu interagieren. Es verwaltet die Aktivitäten und startet neue Threads, die entweder das neuronale Netzwerk trainieren oder es klassifizieren lassen. Hierzu werden Zeitfenster erstellt, die in einem gewissen Zeitraum alle von den unterstützten Sensoren angefallene Daten enthalten und diese dann an den „NeuonalNetworkManager“ weitergeben.
Der „NeuonalNetworkManager“ verwaltet das neuronale Netzwerk und die nötigen zusätzlichen Informationen, wie unterstützte Sensoren und Aktivitäten. Auch sind hier die eigentlichen Trainier- und Erkennungsmethoden, die vom „Activity Recognition Module“, aufgerufen werden, enthalten. Diese rufen dann wiederum die entsprechenden Methoden in der API für das neuronale Netzwerk auf.
Die Klasse „TimeWindow“ enthält wie erwähnt alle Daten der Sensoren und zusätzlich ein „FeatureSet“, in welchem die berechneten Features der Daten gespeichert sind. Die Features wiederum werden von den drei Klassen „StatisticalFeatureExtraction“, „TransientFeatureExtraction“ und „StructuralFeatureExtraction“ berechnet.
Die Klassen „ActivityEnum“ und „Activity“ dienen dazu die Aktivitäten zu speichern, wobei „ActivityEnum“ die Labels der Aktivitäten enthält und „Activity“ die zusätzlich zu den Labels entsprechenden Zeitstempel.

6 Sensor Kommunikationsmodul

6.1 Package: driver

6.1.1 Klassendiagramm

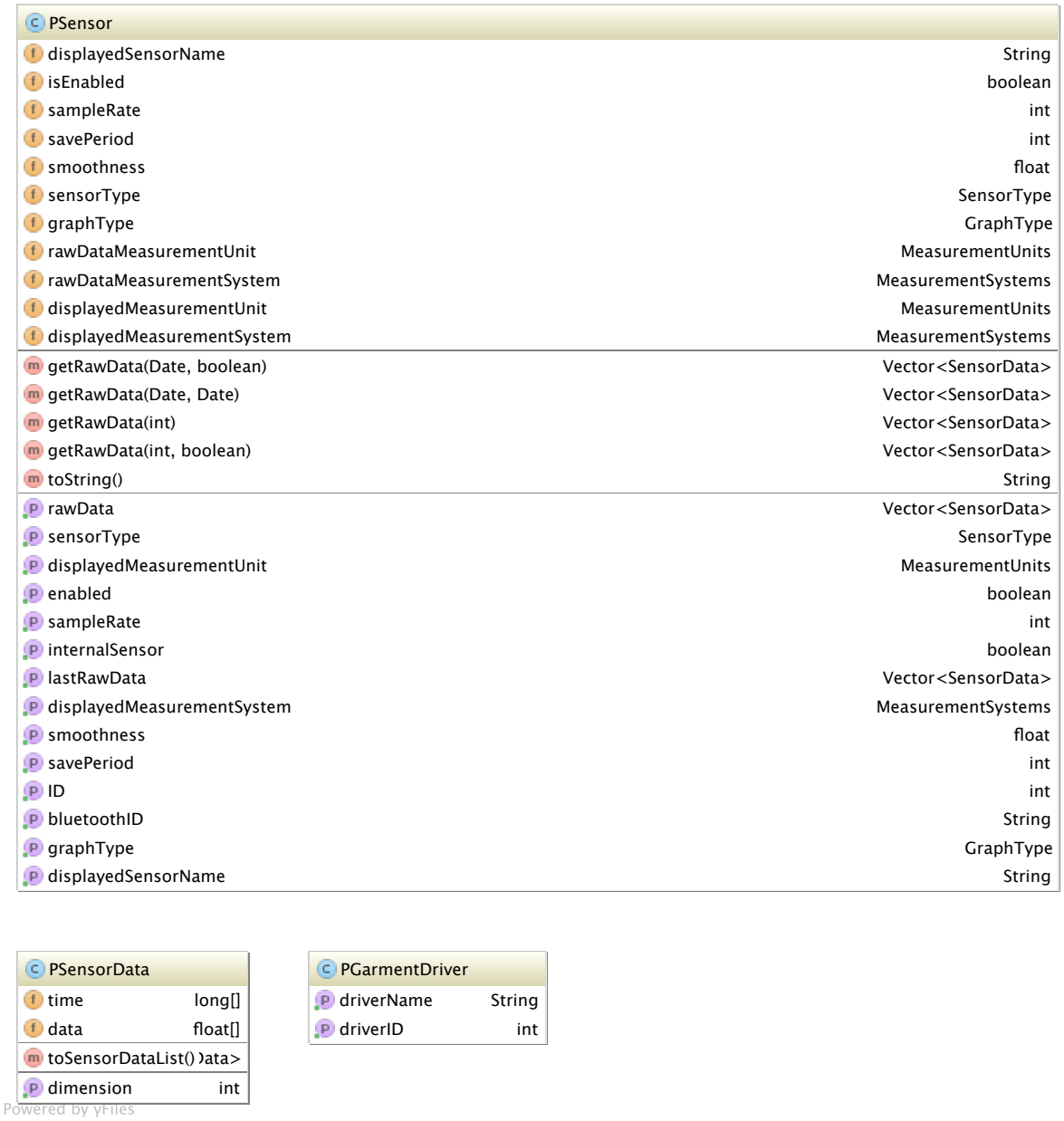


6.1.2 Beschreibung

Das driver Paket stellt das Treiberinterface bereit und eine abstrakte Implementierung eines Basistreibers.

6.2 Package: parcel

6.2.1 Klassendiagramm



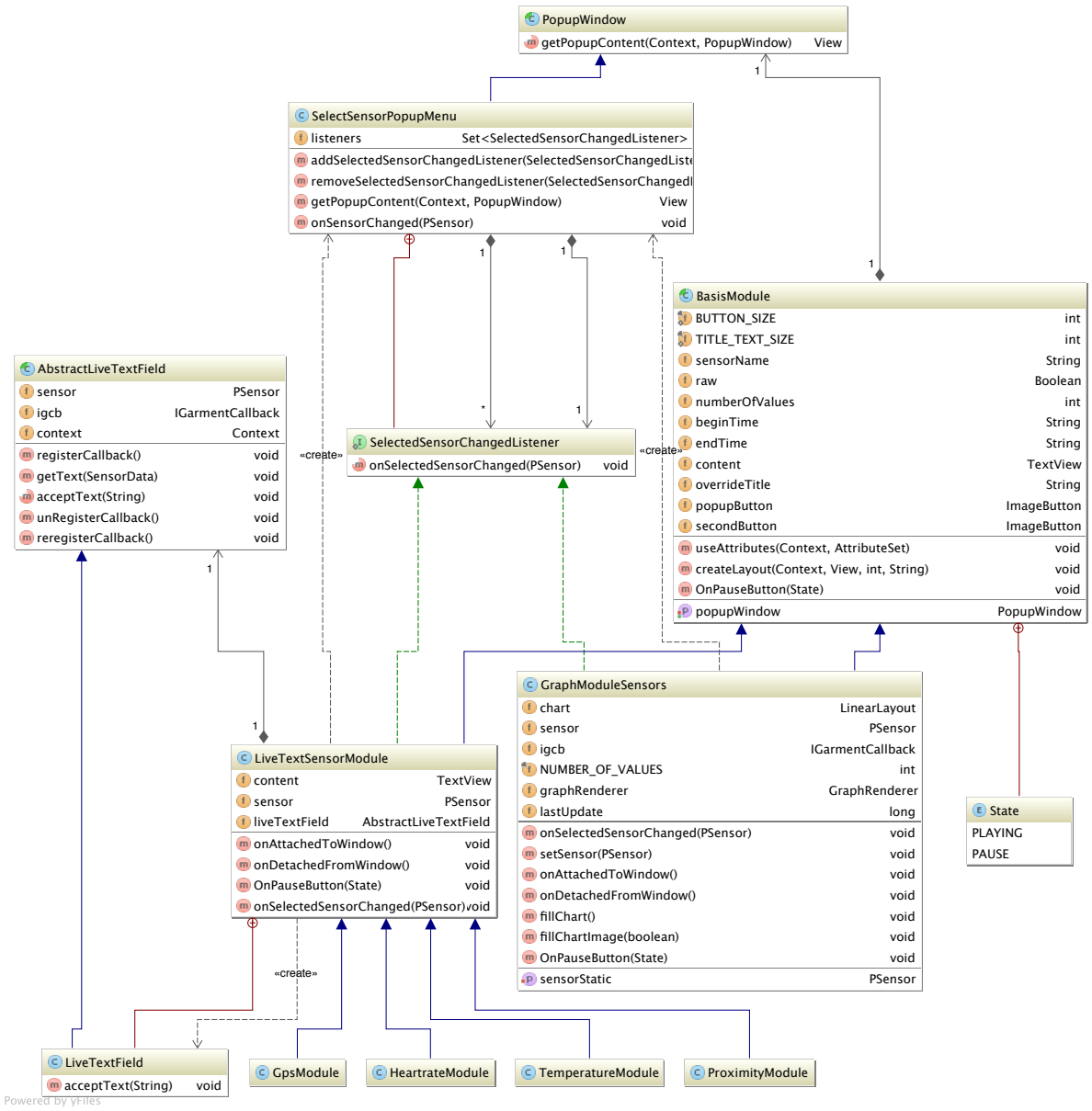
6.2.2 Beschreibung

Das Parcel Paket stellt Basisklassen bereit, die sowohl im SDK als auch im internen SDK verwendet werden.

7 Entwicklungsmodul

7.1 Package: developmentModule

7.1.1 Klassendiagramm



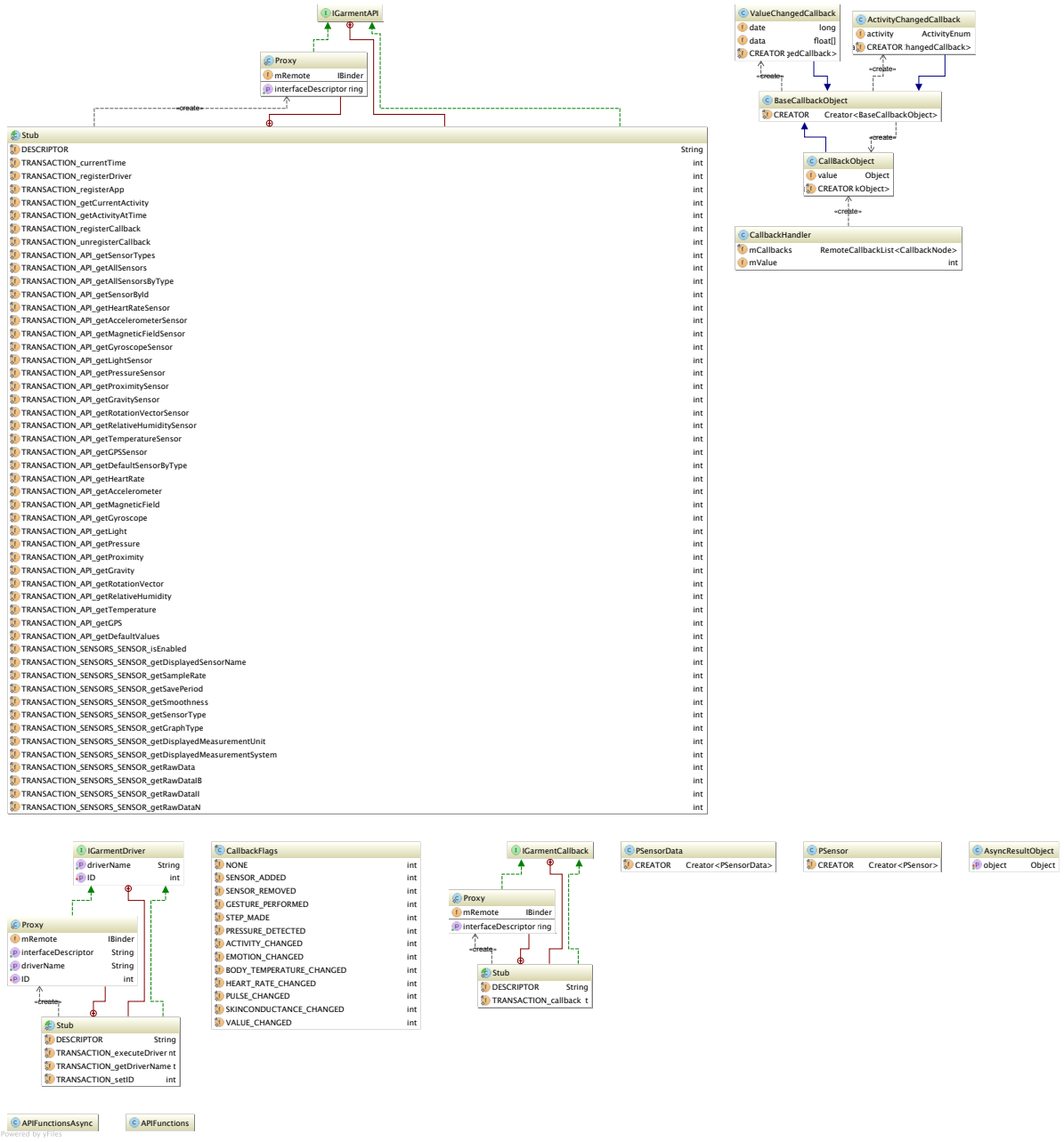
7.1.2 Beschreibung

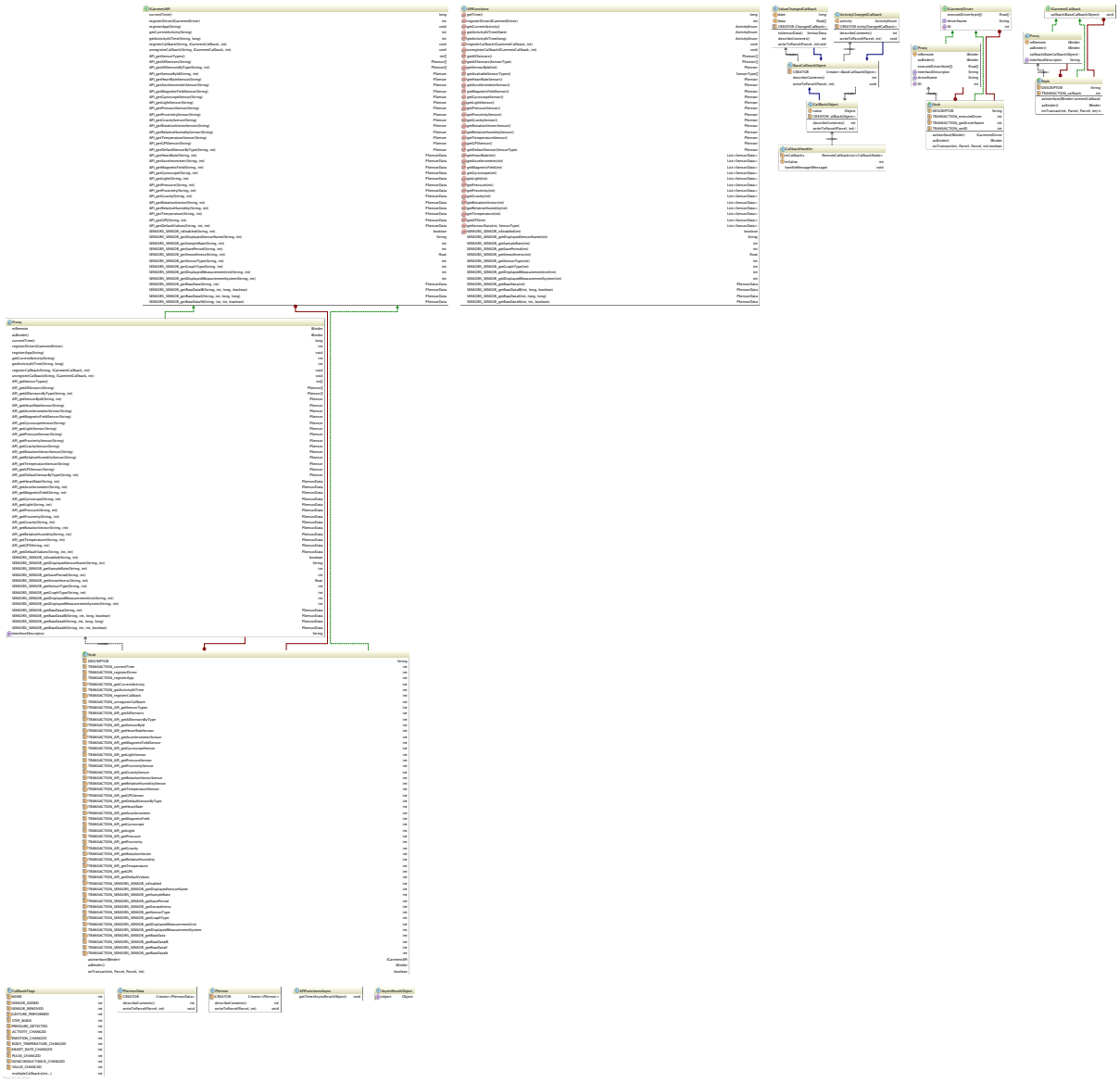
Im der Klasse BasisModul wird das grundlegende Aussehen der einzelnen Module definiert. Die Module bestehen aus einem Layout, einem Bild auf der linken Seite, einem Titel und zwei Buttons am rechten Rand. Der obere Button klappt das Poppup-Menü auf und der untere Button ist eine Play/Pause Button mit dem man das Anzeigen neuer Sensorwerte pausieren und wieder starten kann. Der Popup-Menü Button erscheint nur wenn in den Graph- und LiveTextSensor Modulen kein fester Sensor definiert ist und somit der Benutzer im Menü auswählen kann welcher Sensor angezeigt werden soll.
Die Klassen HeartrateModule, TemperatureModule und GPSModule sind Beispiel Implementierungen für ein LiveTextSensor Module. Sie zeigen die Werte der Sensoren als Text an. Der Sensorwert wird live aktualisiert. Im Konstruktor eines LiveTextSensor Moduls werden der Sensortyp, ein Bild und ein Titel übergeben.
Die Klasse GraphModuleSensors erstellt ein Modul in dem die Sensorwerte in einem Graphen angezeigt werden.
In der Klasse SelectSensorPopupMenu wird das Popup-Menü erstellt um Sensoren auszuwählen. Das Popup-Menü wird im BasisModul mit der Methode setPopupWindow() gesetzt.

7.2 Package: api

7.2.1 Klassendiagramm

Aufgrund der Größe des Klassendiagramms sind hier die Methoden ausgeblendet. Die Methoden sind auf der nächsten Seite in einem größeren Diagramm aufgelistet.





7.2.2 Beschreibung

Dieses Paket stellt die benötigten Klassen, Funktionen und Interfaces bereit, die für die korrekte Funktionsweise des API verwendet werden.

7.3 Package: service

7.3.1 Klassendiagramm

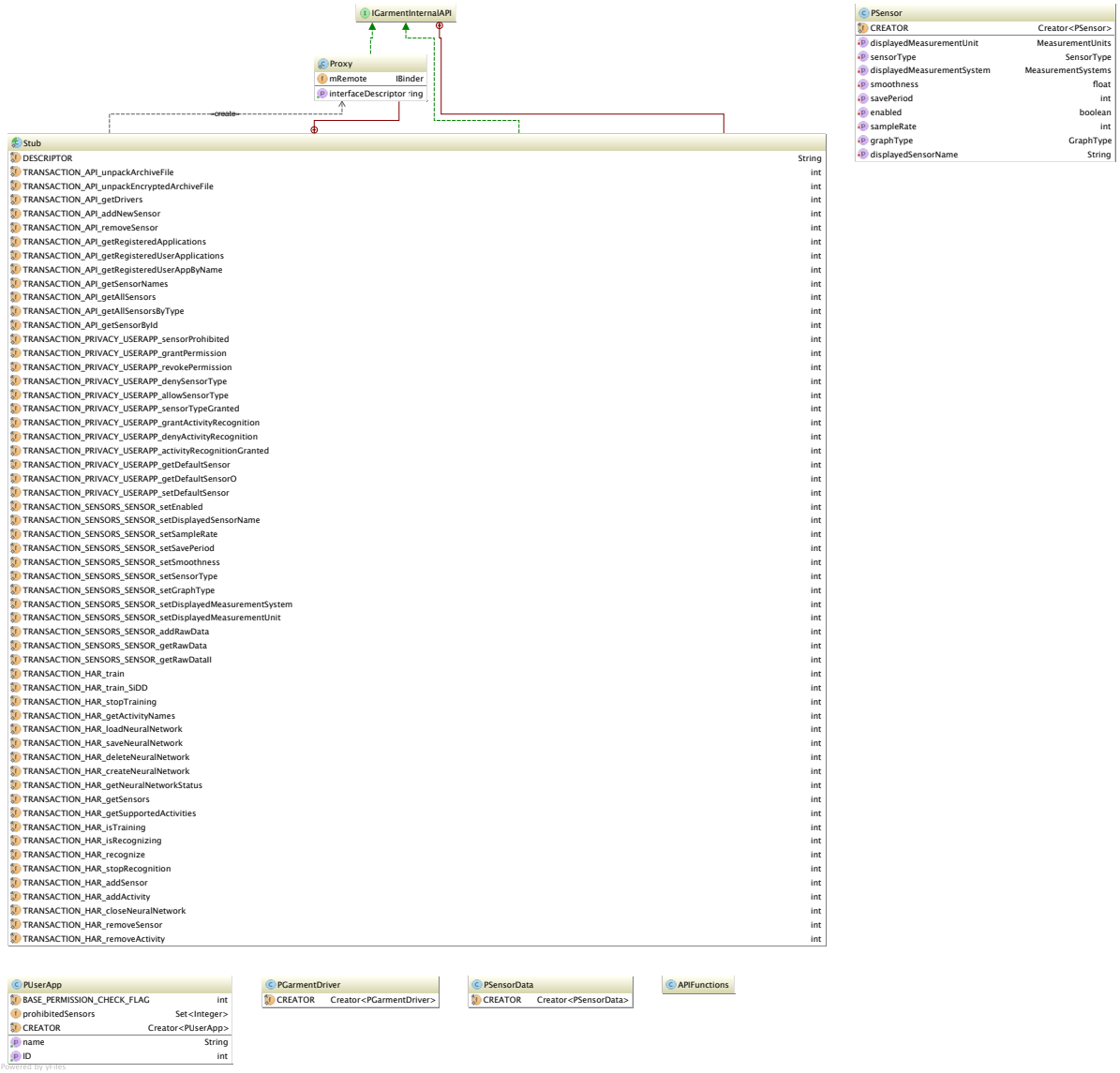
7.3.2 Beschreibung

Das Service Paket beinhaltet die Service Klasse, die das System am laufen hält, Daten verwaltet und die implementierten Funktionen anstößt.

7.4 Package: internal_api

7.4.1 Klassendiagramm

Aufgrund der Größe des Klassendiagramms sind hier die Methoden ausgeblendet. Die Methoden sind auf der nächsten Seite in einem größeren Diagramm aufgelistet.

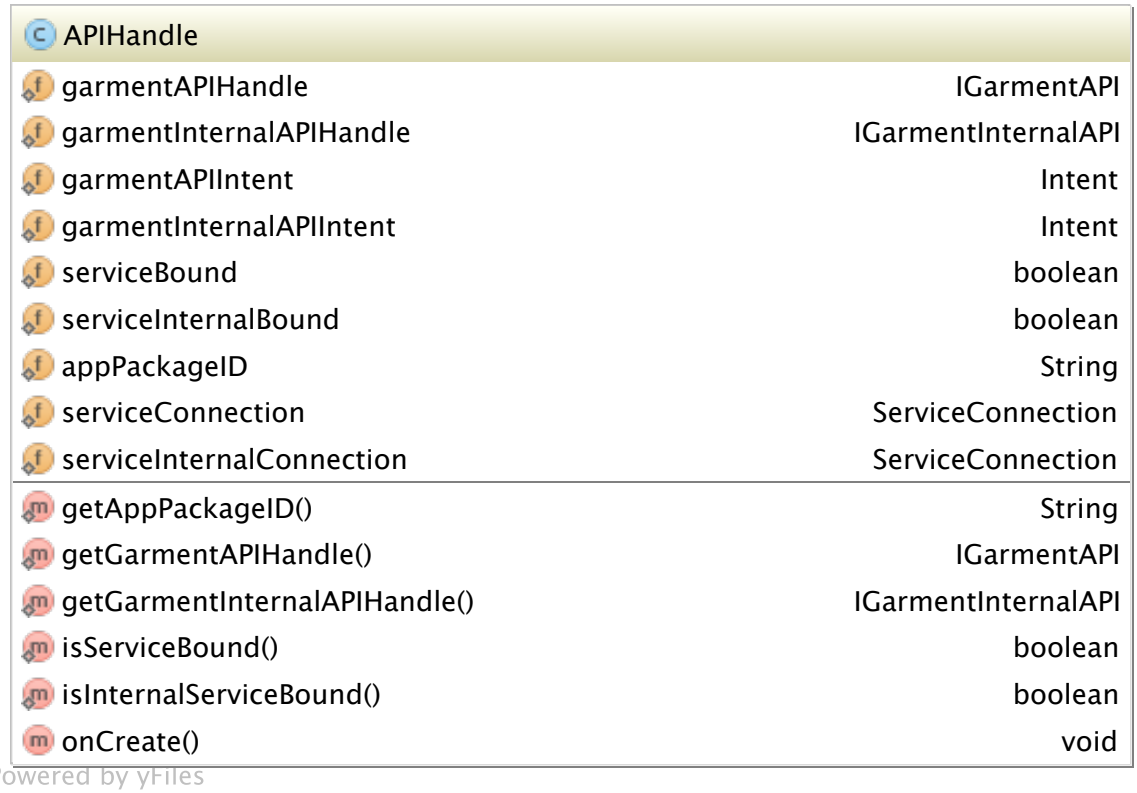


7.4.2 Beschreibung

Das internalapi legt die Funktionen fest und bietet Schnittstellen an, über die die SettingsApp mit dem Service kommuniziert.

7.5 Package: handle

7.5.1 Klassendiagramm



7.5.2 Beschreibung

Das Handle Paket kümmert sich um die Herstellung der Verbindung zwischen App und Service, sowohl für die SettingsApp als auch für das SDK.

8 Settings-App

8.1 Package: app

8.1.1 Klassendiagramm

8.1.2 Beschreibung

MainActivity:
Hier befindet man sich im Hauptmenu der Settings-App. Es gibt 5 Auswahlmöglichkeiten, Sensors (SensorsActivity), Activity Recognition (HARActivityManager), Privacy (PrivacyActivity), Storage (StorageActivity) und About (AboutActivity).

SensorsActivity:
Hier werden alle Sensoren in einer Liste angezeigt. In der Listenansicht können bereits die beiden Werte Smoothness und Poweroptions abgelesen werden. Zusätzlich gibt es auch in der Listansicht einen Switch mit dem der Sensor aktiviert oder deaktiviert werden kann. Jedes der Listviews hat einen Click- und Longclicklistener. Bei einem Click öffnet sich die SensorDetailActivity, durch einen Longclick öffnet sich ein Dialog durch den dieser Sensor gelöscht werden kann. Dieser Longclicklistener ist für interne Sensoren deaktiviert. Zusätzlich befindet sich oben rechts ein Button mit einem Plus. Dieser ermöglicht das Hinzufügen eines neuen Bluetooth Sensors (AddSensorActivity).

SensorDetailActivity:
In dieser Activity können alle, vom Benutzer definierbaren Eigenschaften für einen Sensor, bearbeitet werden.

AddSensorActivity: In dieser Activity können alle, vom Benutzer definierbaren Eigenschaften für einen Sensor, auf einen beliebigen Wert gesetzt werden. Wird die Eingabe mit dem Save-Button bestätigt, befindet man sich wieder in der SensorsActivity und dort erscheint auch der neu angelegte Sensor.

HARActivity:
In dieser Activity ist eine kurze Erklärung zum Activity Recognition Module und den Ablauf des Trainings bzw. der Erkennung. Zusätzlich besteht die Möglichkeit in die Einstellungen des neuronalen Netzwerkes und, sofern schon trainiert, in das Trainingsmenü zu wechseln.

HARActivityManager:
In dieser Activity ist es dem Nutzer möglich die unterstützten Sensoren und Aktivitäten zu ändern. Diese werden beim drücken auf Entsprechendem Button aufgelistet und sind je nach Unterstützung markiert. Das Erstellen von neuen neuronalen Netzwerken, sowie auch das Löschen ist auch aus dieser Aktivity möglich.

HARActivityTraining:
In dieser Activity können Einstellungen für das Training vorgenommen werden. Es kann eingestellt werden, ob das Training Live oder mit Hilfe von bereits gespeicherten Daten durchgeführt werden soll. Bei Letzterem muss zusätzlich noch der Zeitraum der Daten mit angegeben werden, was durch ensprechende Dialoge vereinfacht wird. Zusätzlich müssen die zu trainierende Aktivität und die Länge der Zeitfenster angegeben werden. Die verschiedenen Aktivitäten werden hierzu aufgelistet.

PrivacyActivity:
In dieser Activity werden alle Apps aufgelistet die auf die API der Settings-App zugreifen.

Durch einen Click auf das Listview Element öffnet sich die Detailansicht (AppDetailActivity).

AppDetailActivity:
In dieser Activity werden alle angelegten Sensoren der Settings-App aufgelistet. Falls mehrere Sensoren des gleichen Typs vorhanden sind kann der Default Sensor für die aktuelle App über einen Spinner ausgewählt werden. Über ein Longclick auf das Listview Element öffnet sich ein Dialog, durch den ein kompletter Sensortyp, für diese App deaktiviert oder aktiviert werden kann.

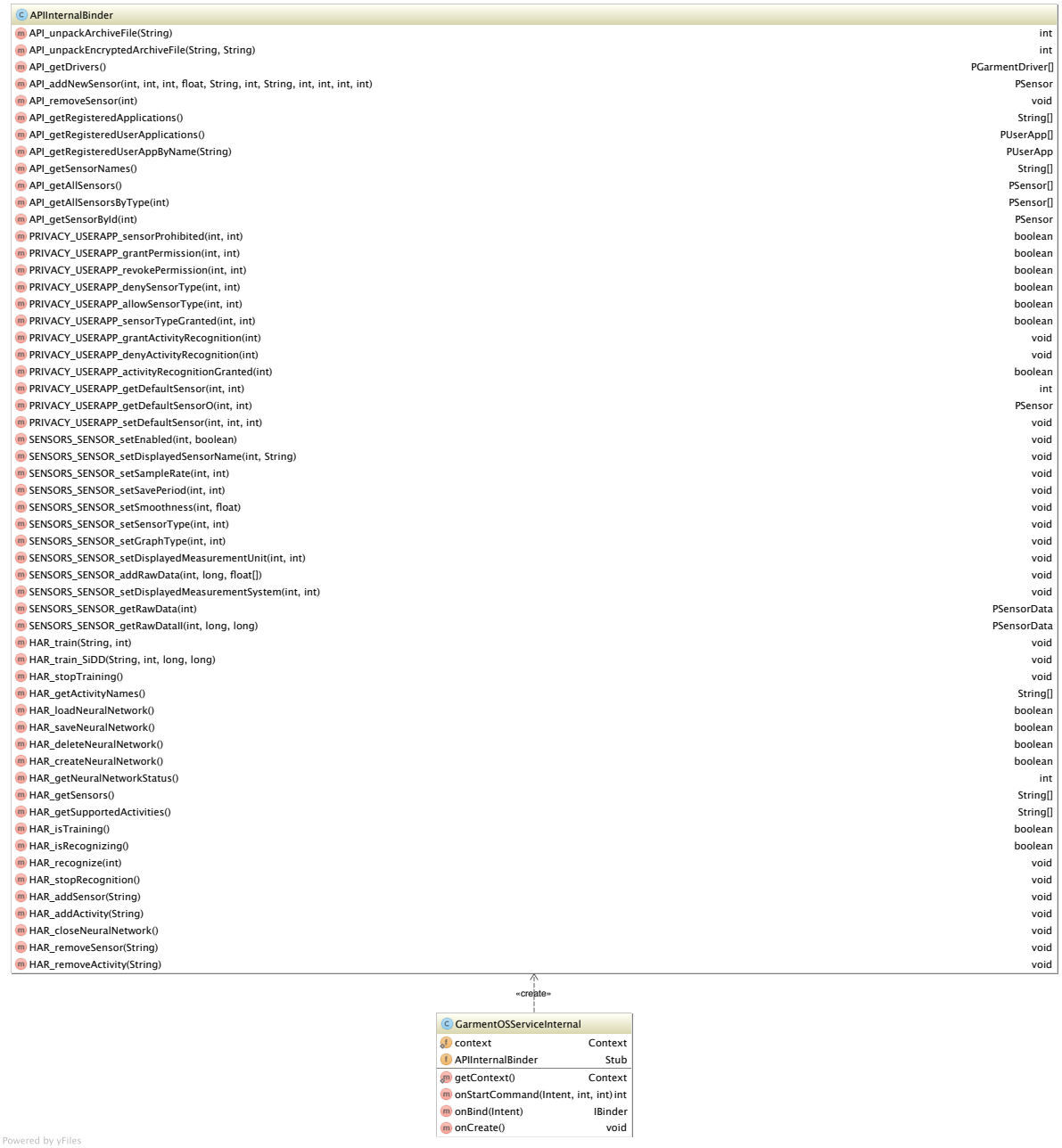
StorageActivity:
In dieser Activity werden vier verschiedene Arten von Export- und Importmöglichkeiten angezeigt. Durch diese können die aktuellen Daten exportiert oder bereits bestehende Daten importiert und gemerged werden. Es gibt die Möglichkeit die Daten zu verschlüsseln oder sie unverschlüsselt zu exportieren. Durch die Auswahl einer der vier Möglichkeiten öffnet sich die ImportExportArchiveActivity in der der Speicherort in Form der Ordnerstruktur des gewählten Ziels ausgewählt werden kann.

AboutActivity:
In dieser Activity befindet sich das Impressum und Verweise auf verwendet Bilder.

ImportExportArchiveActivity:
Hier wird die Möglichkeit geboten, auf einen externen Datenträger ein selbsterstelltes Archiv mit Sicherungsdateien der aktuell lokal gehaltenen Sensordaten zu erstellen oder ein solches zuvor erstelltes Archiv von einem externen Datenträger zu importieren.

8.2 Package: internalservice

8.2.1 Klassendiagramm

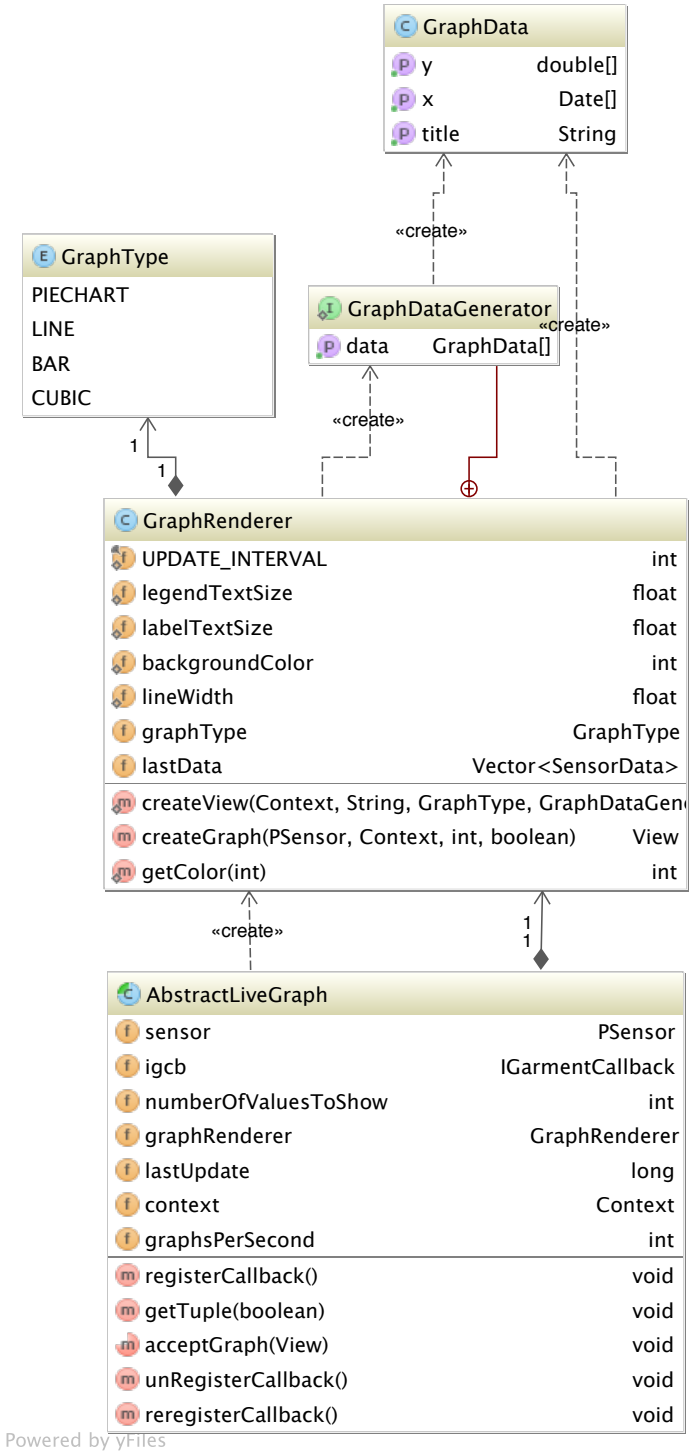


8.2.2 Beschreibung

Das Paket internalservice kümmert sich um die Kommunikation zwischen der SettingsApp und dem Service.

8.3 Package: graph

8.3.1 Klassendiagramm



8.3.2 Beschreibung

Dieses Paket ist für die visuelle Darstellung der Sensordaten zuständig.

8.4 Package: privacy

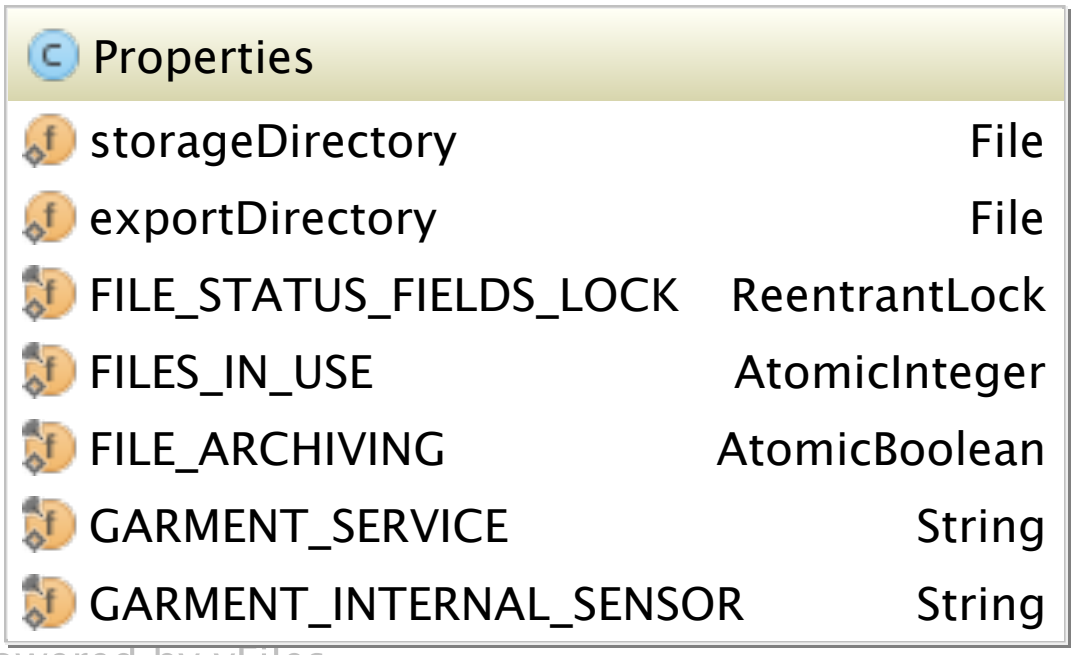
8.4.1 Klassendiagramm

8.4.2 Beschreibung

Das Paket Privacy verwaltet die Rechtevergabe von GarmentOS.

8.5 Package: properties

8.5.1 Klassendiagramm



8.5.2 Beschreibung

Das Paket Properties speichert den Export- und Importpfad für die Dateienverwaltung.

9 Versionshistorie

Version 0.1 - 01.03.2015

- Dokument erstellt

(Velihan Bulut)

Version 0.2 - 24.03.2015

- UML-Klassendiagramme eingefügt

(Velihan Bulut)

Version 0.2.1 - 25.03.2015

- UML-Klassendiagramme neu angeordnet

(Velihan Bulut)

Version 0.2.2 - 26.03.2015

- Bildgrößen angepasst
- Fehler in den Überschriften verbessert

(Velihan Bulut)

Version 0.3 - 14.04.2015

- UML Bilder aktualisiert
- Section Beispiel-App entfernt
- Beschreibungen für Entwicklungsmodul, Sensormodul und Activity recognition Module eingefügt.

(Velihan Bulut)

Version 1.0 - 28.04.2015

- UML Bild des Packages app aktualisiert
- Beschreibungen für die restlichen Packages eingefügt.

(Velihan Bulut)