Open Diabetes UAM
Heuristik Algorithm

Pflichtenheft UAM

Gruppe 11:	Aino Schwarte <aino.schwarte@stud.tu-darmstadt.de> Anna Mees <anna.mees@stud.tu-darmstadt.de> Jan Paul Petto <janpaul.petto@stud.tu-darmstadt.de> Paul Wolfart <paul.wolfart@stud.tu-darmstadt.de> Tom Großmann <tom.grossmann@stud.tu-darmstadt.de>
Teamleiter:	Benedikt Schneider <schneider-benedikt@gmx.net>
Auftraggeber:	M.Sc. Jens Heuschkel <heuschkel@tk.tu-darmstadt.de> Telecooperation Smart Urban Networks
Abgabedatum:	31.03.2019

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Bachelor-Praktikum WS 2018/2019
Fachbereich Informatik

Inhaltsverzeichnis

1 Zielbestimmung

Das Projekt Open Diabetes UAM Heuristik Algorithmen entwickelt ein Programm zur richtigen Erkennung von Mahlzeiten anhand von Blutzuckerwerten, die in einer Nightscout Datenbank gespeichert sind. Nightscout ist eine Onlineplattform zur grafischen Darstellung von Blutzuckerwerten, Insulindosierungen und Mahlzeiten.

Folgende Punkte müssen implementiert bzw. erstellt werden:

Must-Have:

- Skript zum Lesen und Schreiben von Daten in eine Nightscout Instanz
- Kommandozeilentool zum Lesen, Schreiben und Synchronisieren von Nightscout
- Parser zum Überführen von Datensätzen aus dem Skript in Java
- (mind. 1) Algorithmus zur korrekten Berechnung von Kohlenhydraten
- Modifikation von Nightscout um angekündigte und berechnete Kohlenhydrate getrennt speichern zu können.
- Plotten des resultierenden Blutzuckerverlaufs
- Kommandozeilentool, das Daten einliest und berechnete Kohlenhydrate und Plots ausgibt
- Docker-Container der zum Programmstart einen Datensatz einliest, den Algorithmus ausführt und berechnete Kohlenhydrate und Plots ausgibt
- Wikiartikel:
  - Anleitung für das Kommandozeilentool
  - Erklärung wie die Daten aus Nightscout auf unsere Daten abgebildet werden
  - Erklärung zu Algorithmen und mögliche Einstellungsfaktoren
  - Anleitung zum Aufsetzen und Einstellen von lokalen Nightscout Instanzen die mit dem Tool funktionieren

Should-Have:

- Modifikation von Nightscout um die Daten und die berechneten Kohlenhydrate in Nightscout im Tagesprofil korrekt anzuzeigen
- Dokumentation der Nightscout Modifikationen

2 Einsatz

Die Open Source Community OpenAPS unterstüzt Versuche eine künstliche Bauchspeicheldrüse zu entwickeln. Bisherige Systeme können die Grundversorgung an Insulin automatisch an die gemessenen Blutzuckerwerte anpassen. Eine komplett automatische Insulinversorgung ist allerdings noch nicht gelungen, hauptsächlich wegen des Unannounced-Meal (kurz UAM) Problems. Zu jeder Mahlzeit muss zusätzliches Insulin verabreicht werden, um zu hohe Blutzuckerwerte zu verhindern. Da die üblichen Insulinarten langsamer auf den Blutzuckerspiegel wirken als Mahlzeiten, ist es essenziell, dass diese Dosis so früh wie möglich verabreicht wird. In einem vollständig automatischen System besteht also die Notwendigkeit Mahlzeiten zu erkennen.

Unser Projekt dient als Prototyp zur Lösung des UAM Problems. Wir liefern einen beispielhaften Algorithmus, der die Menge einer Mahlzeit anhand der Steigung des Blutzuckerspiegels annähert. Unser Programm kann außerdem mit weiteren Algorithmen erweitert werden, und bietet so eine Umgebung zum Testen von verschiedenen Ansätzen. Unser Projekt ist auf Nightscout ausgelegt, auf dem viele künstliche Bauchspeicheldrüsen-Systeme basieren. In Nightscout können kostenlos mehrere Monate an Daten speichert werden, an denen die Mahlzeitenerkennung getestet werden kann, ohne die Gesundheit einer Person direkt zu gefährden.

Außerdem können die einzelnen Bauteile, wie das Synchronizations-Tool für Nightscout Instanzen auch separat wiederverwendet werden.

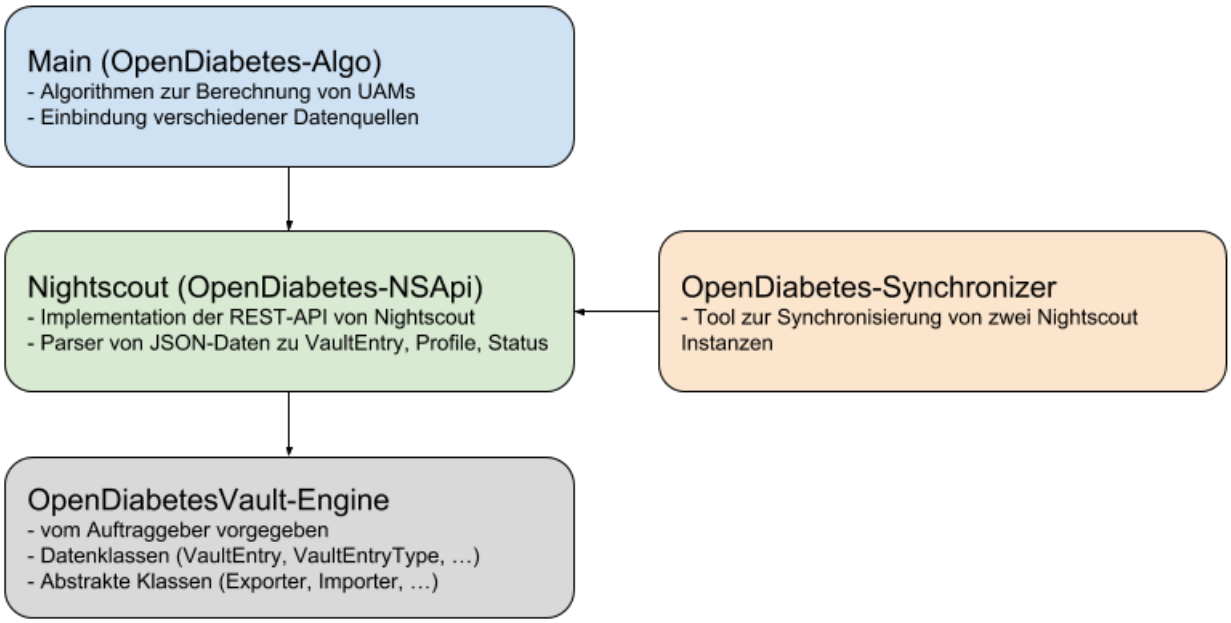
3 Ausgangslage

Für das Projekt stehen folgende Infrastrukturen zur Verfügung:

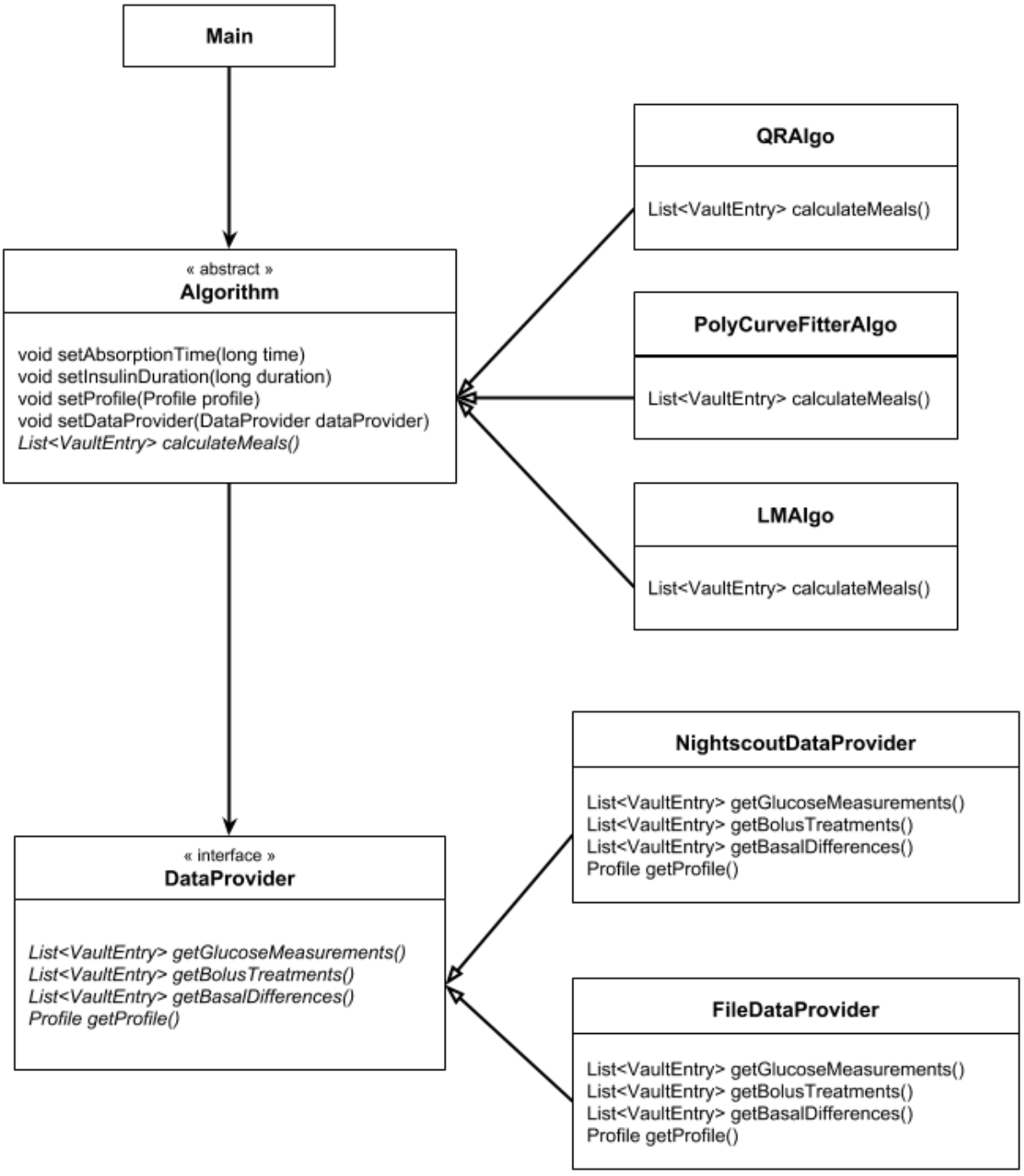
- Anonymisierte Patientendaten zum Testen der Ansätze
- Nightscout um eigene Instanzen aufzusetzen
- Beschreibungen der Tools
- Paper mit Modell zum Einfluss von Insulin und Kohlenhydrate auf den Blutzuckerspiegel
- Der VaultEntry Datentyp, der zum Parsen verwendet werden soll

4 Produktübersicht

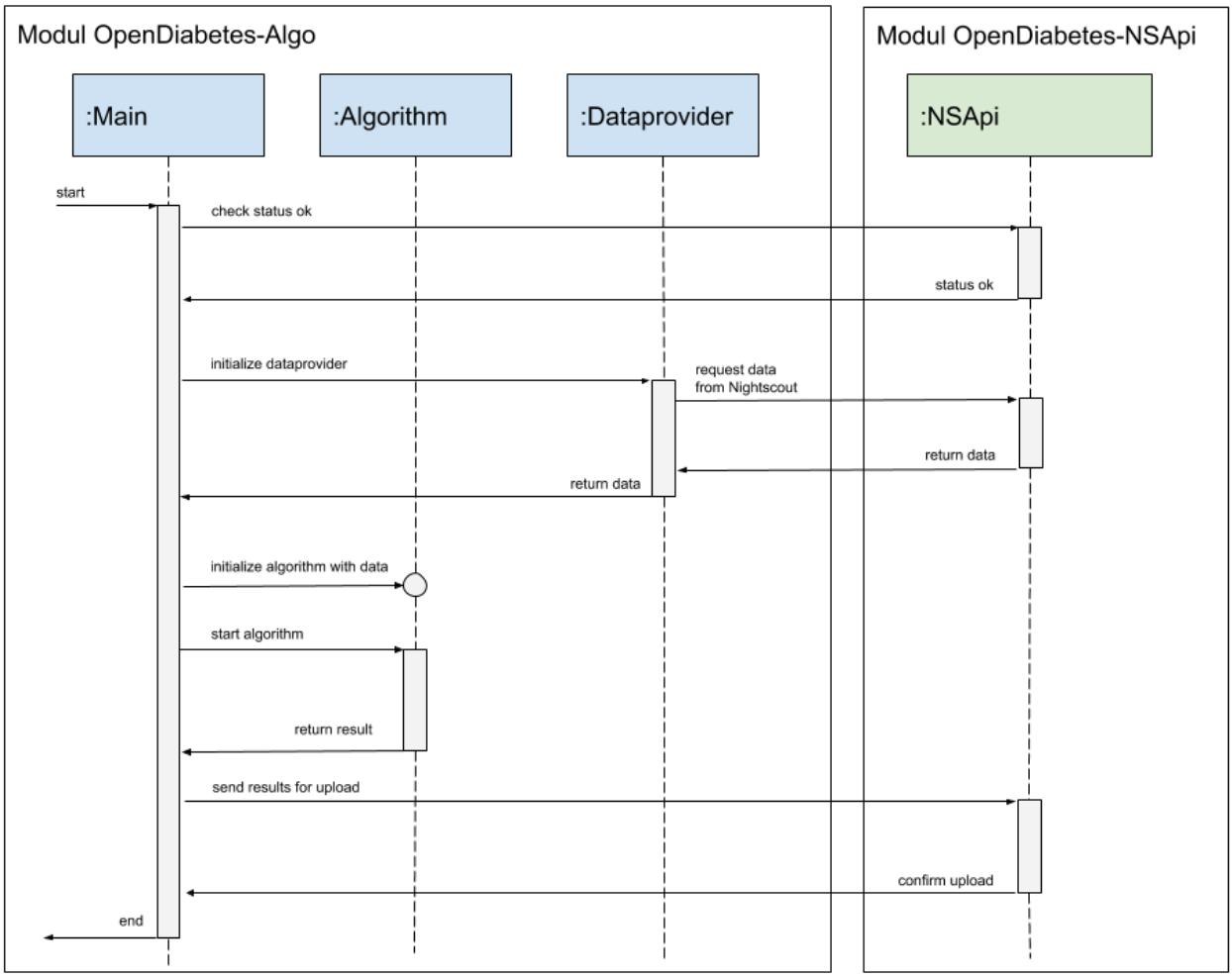
Aufbau der Module



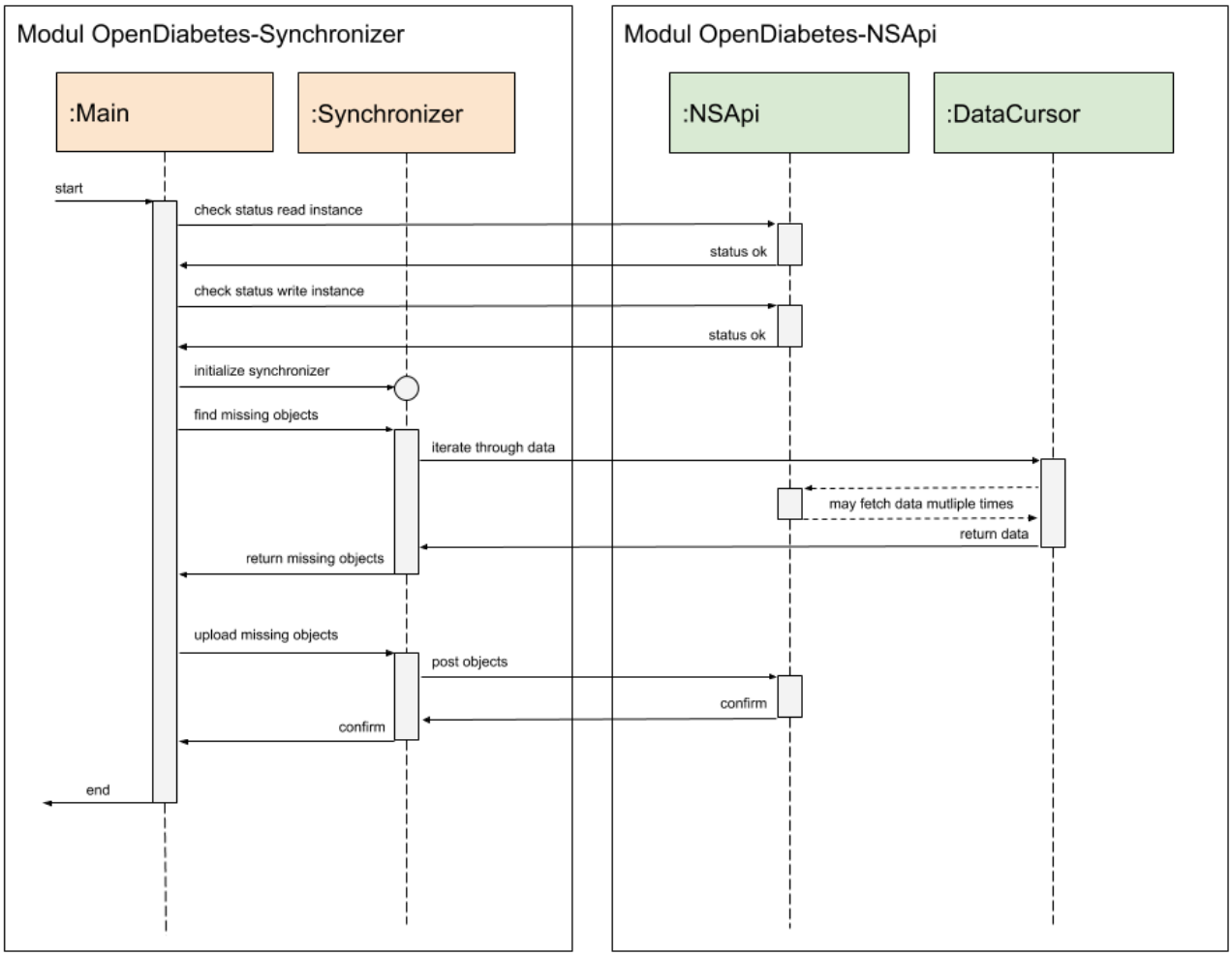
UML-Diagramm Algorithm:



Ablauf Mahlzeitenerkennung:



Ablauf synchronisieren von Daten:



5 Anforderungen

5.1 funktional

- Der Algorithmus muss die Größe der stattfindenden Mahlzeiten (Kohlenhydrate in Gramm) korrekt erkennen. Überprüft wird das an dem gegebenem Datensatz, auf Abschnitten in denen über einen Zeitraum von mindestens 6 Stunden durchgehend Blutzuckerdaten vorhanden sind. In diesen finden alle 5 Minuten Messungen statt. Da die zu erkennenden Mahlzeiten nicht markiert sind, und dadurch ein direkter Vergleich nicht möglich ist, wird ein Blutzuckerverlauf anhand der bekannten Events und der berechneten Mahlzeiten generiert. Dieser soll zu jedem Zeitpunkt höchstens 10% von dem originalen Blutzuckerverlauf abweichen. Eine höhere Abweichung wird toleriert, wenn der Abschnitt eindeutig seltsames Verhalten aufweist, z.B. einen plötzlichen Einbruch des Blutzuckerspiegels.
- Der Synchronizer muss zwei Nightscout Insztanzen vergleichen und alle Einträge aus dem ersten, die aus dem zweiten fehlen, in das zweite einfügen.
- Verschiedene Datenquellen müssen einbindbar sein. Lesen aus Nightscout und einer Datei soll unterstützt sein.
- Der Parser soll Nightscout Daten im JSON Format auf einheitliche, vom Arbeitgeber vorgegebene, Datentypen abbilden.
- Darstellung der Daten/Mahlzeiten in Nightscout siehe User-Stories
- Darstellung der entstandenen Blutzuckerkurve siehe User-Stories

5.2 nicht-funktional

- Das Programm muss auf einem PI Zero in angemessener Zeit laufen. Angemessen bedeutetd dabei unter 5 Minuten.
  - Anmerkung: Da uns kein PI Zero zur Verfügung steht, und der Arbeitgeber nicht allzu großen Wert darauf legt, wird diese Anforderung während des Projektes fallen gelassen.
- Das Wiki in GitHub muss eine Einführung und Übersicht für das Projekt und Anleitungen für zukünftiges Einbinden von neuen data providern und Algorithmen enthalten. Außerdem muss das Vorgehen von den genutzten Algorithmen erklärt werden.

6 Qualitätssicherung

Siehe QS-Dokument.

7 Rechtliches

Wir entwickeln unter der AGPLv3-Lizenz und verwenden nur Open-Source Quellen. Dadurch vermeiden wir Copy-Right-Verletzungen und schließen jede Garantie an unserem Code aus.