

**FH Oberösterreich / Campus Hagenberg**

MUS – Projekt

Smart Desk

|  |  |
| --- | --- |
| **Autor:** | Fabian Loris  Philipp Haider |
| **Abgabetermin:** | 25.6.2016 |
| **Semester:** | 2. Semester |
| **Übungsleiter:** | Philipp Pendelin MSc |

**Inhaltsverzeichnis**

[1 Aufgabenstellung und Ziele 1](#_Toc454633711)

[1.1 Must-Haves 1](#_Toc454633712)

[1.2 Nice-To-Haves 1](#_Toc454633713)

[2 Verwendete Werkzeuge 2](#_Toc454633714)

[3 Architektur 3](#_Toc454633715)

[4 Quelltext 4](#_Toc454633716)

[5 Anwendung 4](#_Toc454633717)

[6 Ergebnisse 7](#_Toc454633718)

# Aufgabenstellung und Ziele

Heutzutage zeichnen schon viele Menschen ihr Bewegungsprofil mit einem „intelligenten“ Armband auf. Ihr Ziel ist eine bewusstere und gesündere Lebensweise. Doch was ist mit der vielen Zeit die wir fast regungslos im Büro sitzen. Hier kann ein (elektrischer) Stehtisch Abhilfe schaffen. Aber auch permanentes stehen ist auf Dauer ungesund. Darum möchten wir ein Gerät entwickeln, dass die Steh- und Sitzzeiten aufzeichnet und auf dieser Basis dem Anwender Empfehlungen und Auswertungen anbieten kann. Für die Interaktion mit dem Benutzer kommen viele neuartige Möglichkeiten in Frage. Nachfolgend die geplanten Teilbereiche dieses Projekts.

## Must-Haves

* Entwicklung einer Arduino/Pi/Netduino Anwendung die über einen Abstandssensor erkennt ob der Anwender sitzt oder steht
* Verbindung zum PC, sodass die Software erkennen kann ob der Anwender aktiv arbeitet oder sich nicht am Arbeitsplatz befindet
* Übermitteln der Daten in die Cloud (z.B. Microsoft IoT-Hub)
* Speichern, aufbereiten und aggregieren der Daten für spätere Auswertungen
* Visualisierung der Daten in Form einer Web-Seite (Tages-Verlauf, Aktivitäts- und Inaktivitätszeiten, Wochenverlauf, ...)
* Senden einer Benachrichtigung, wenn der Anwender zu lange sitzt (z.B. durch Browser-Notification)

## Nice-To-Haves

Je nach Aufwand der Must-Have Arbeitspakete wären auch noch folgende Funktionen interessant:

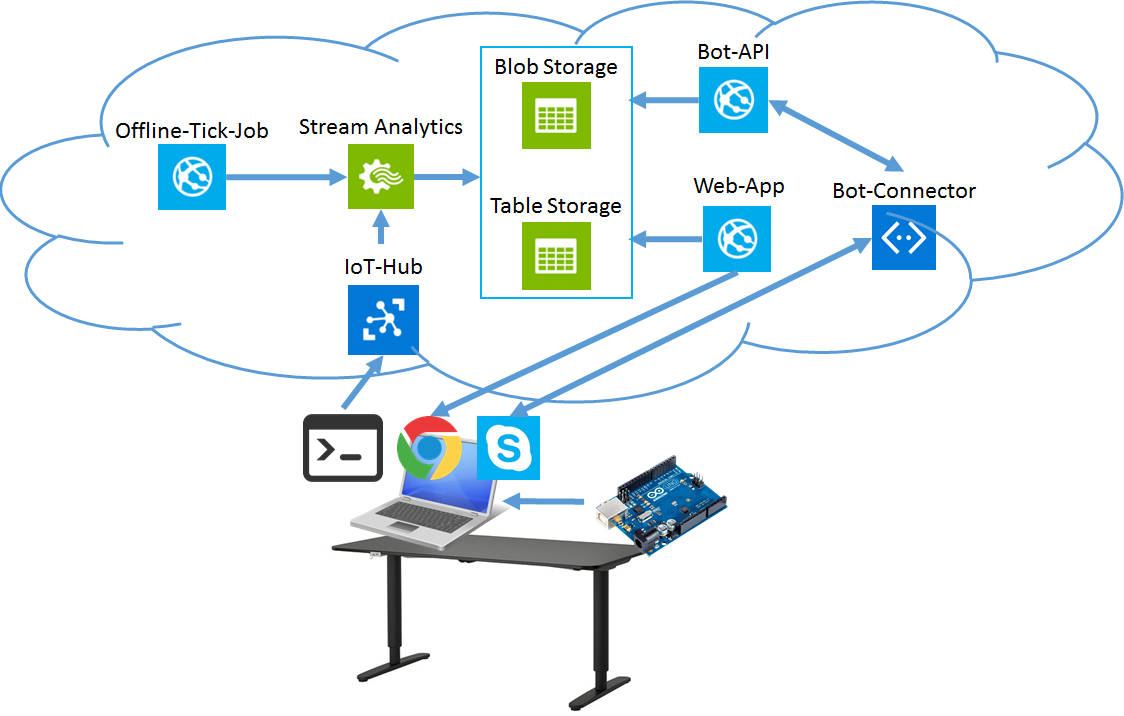
* Benachrichtigung durch einen Chat-Bot (z.B. über Skype, ...)
* Es wäre auch interessant den Chat-Bot so intelligent zu gestalten, dass er verschiedene Fragen beantworten kann.
* Interessant wäre auch eine Steuerung des Tisches, z.B. über einen Sprachbefehl. Oder eine automatische Positionsveränderung, wenn der Anwender eine Pause macht. (Kann natürlich auch gefährlich/unerwünscht sein)

# Verwendete Werkzeuge

Schlussendlich verwendete Werkzeuge und Frameworks:

* Visual Studio
* Arduino
* Microsoft Azure
* Azure IoT Hub
* Microsoft Bot Connector
* Microsoft Bot Framework
* Azure Web App Service
* Azure Web Jobs
* ASP.NET Core 1.0
* Azure Stream Analytics
* Skype
* Azure Storage

# Architektur



Die gesamte Applikation baut auf den Azure-Stack auf und ist vollständig in der Cloud gehostet. Die Clients verbinden sich mithilfe des IoT-Client SDKs zum IoT-Hub und senden ihre Informationen via HTTP und JSON.

Der Stream Analytics Job liest vom IoT-Hub und verarbeitet diese Daten weiter. Ein kontinuierlicher Job ist für die korrekte Arbeitsweise der Offline-Erkennung erforderlich. Dieser Job stellt sicher, dass der Analytics Job aktiv bleibt, da dieser nur aktiv ist, wenn zumindest 1 Client verbunden ist. Die Konfiguration der einzelnen Clients ist in einem Blob Storage abgelegt, da Stream Analytics nur von einem Blob Storage Referenzdaten lesen kann. Die Konfiguration wird unteranderem für die Offline-Erkennung benötigt, weil dem Job alle Clients bekannt sein müssen. Die durch Stream Analytics aufbereiteten Daten werden in einem Table Storage abgelegt.

Eine auf ASP.NET Core aufbauende Webapplikation greift über die Azure Storage SDKs auf die abgespeicherten Daten zu und liefert diese über eine REST-API aus. Diese Applikation liefert ebenfalls die statischen Ressourcen für das Dashboard aus.

Der mit dem Bot Framework entwickelte Bot, greift ebenfalls auf den Table Storage zu, um dem jeweiligen Benutzer Auskunft über seine Daten zu geben. Der SmartDesk-Client ist mit dem Arduino verbunden und liest die Messwerte über die serielle Schnittstelle aus. Zusammen mit der Information, ob der User gerade aktiv ist, wird der Messwert in fixen Zeitintervallen an den IoT-Hub gesendet.

# Quelltext

Der Quelltext ist unter folgender Adresse abrufbar:

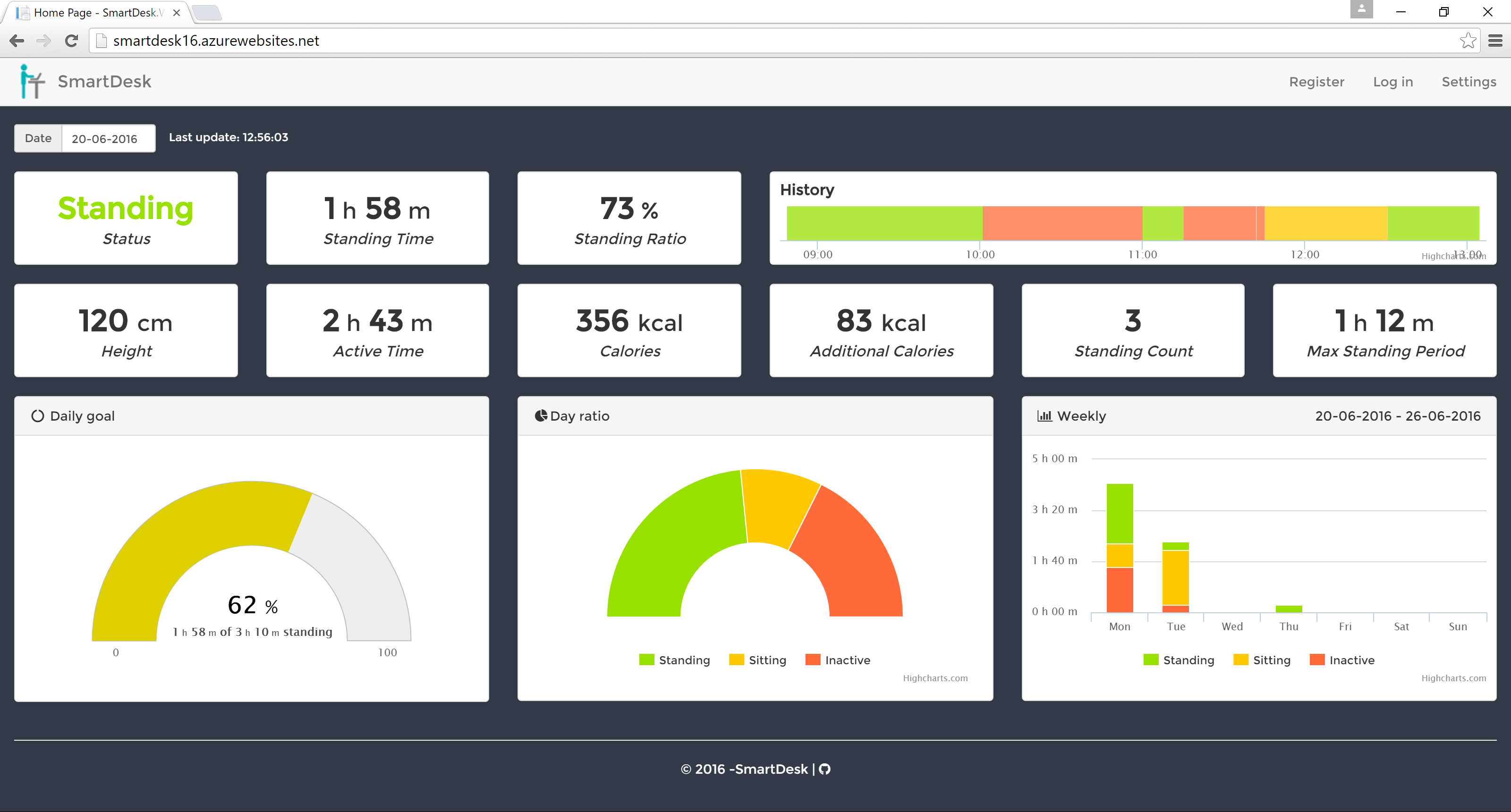
<https://github.com/pihai/SmartDesk>

# Anwendung

Ein Benutzer kann sich seine Daten auf einer Webseite anzeigen lassen. Es kann jeder aufgezeichnete Tag abgerufen werden. Folgende Informationen sind enthalten:

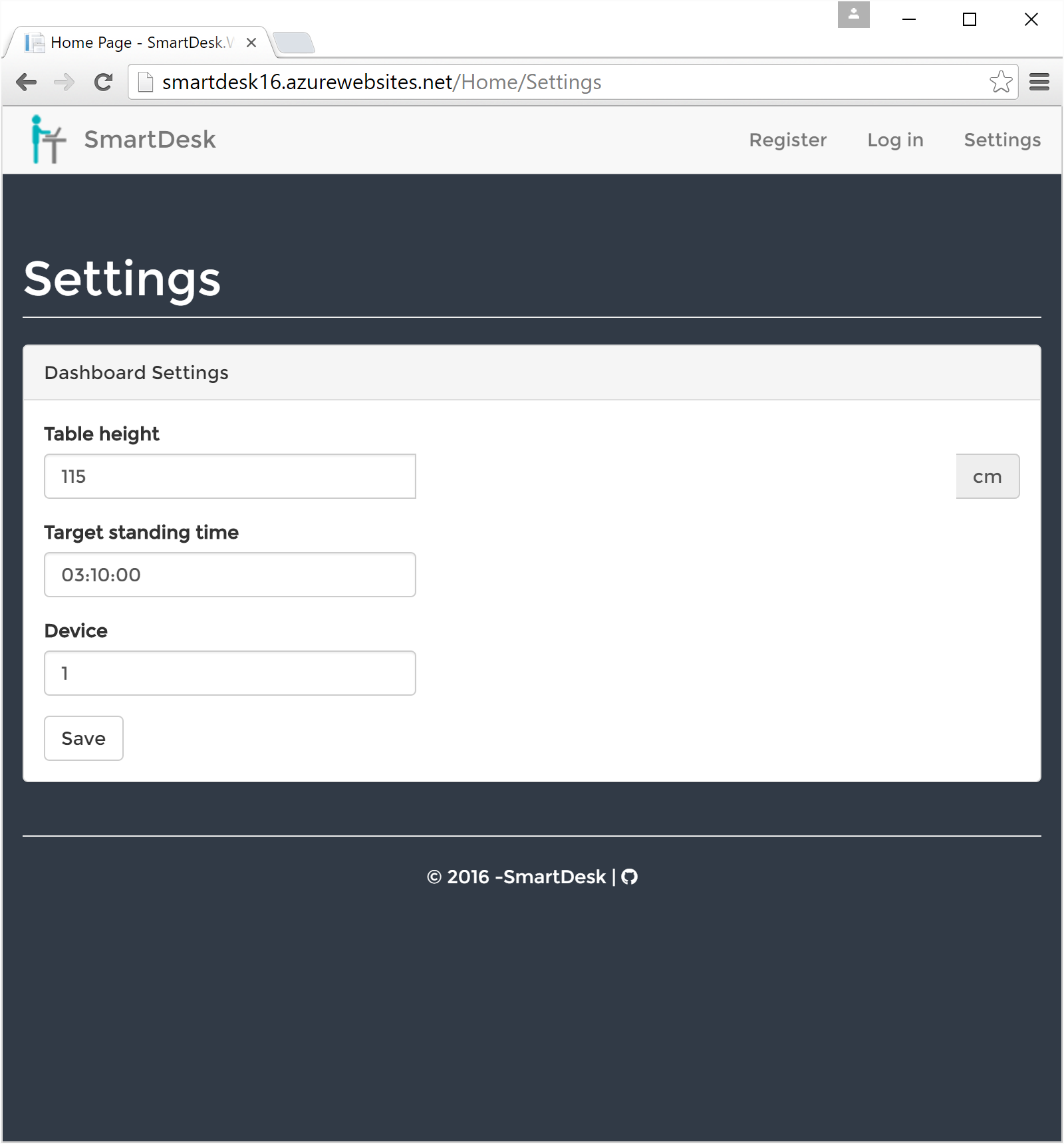
* Der aktuelle Status (Stehen, Sitzen, Inaktiv, Offline)
* Summe der Stehzeit an diesem Tag
* Verhältnis zwischen sitzen und stehen in Prozent
* Tagesverlauf, wann welcher Zustands war
* Aktuelle Tischhöhe in cm
* Gesamte „Arbeitszeit“ (sitzen und stehen, aber nur wenn der Benutzer aktiv gearbeitet hat)
* Verbrannte Kalorien an diesem Tag durch sitzen oder stehen
* Da stehen mehr Kalorien verbrennt als sitzen werden die zusätzlich verbrannten Kalorien angezeigt
* Wie oft der Benutzer gestanden ist
* Längste durchgehende Steh-Periode
* Wie viel von seinem persönlichen Stehdauer-Ziel der Benutzer bereits erreicht hat
* Verhältnis zwischen sitzen/stehe/inaktiv
* Verlauf der ganzen Woche

Über die Datumsauswahl kann jeder beliebige Tag visualisiert werden.

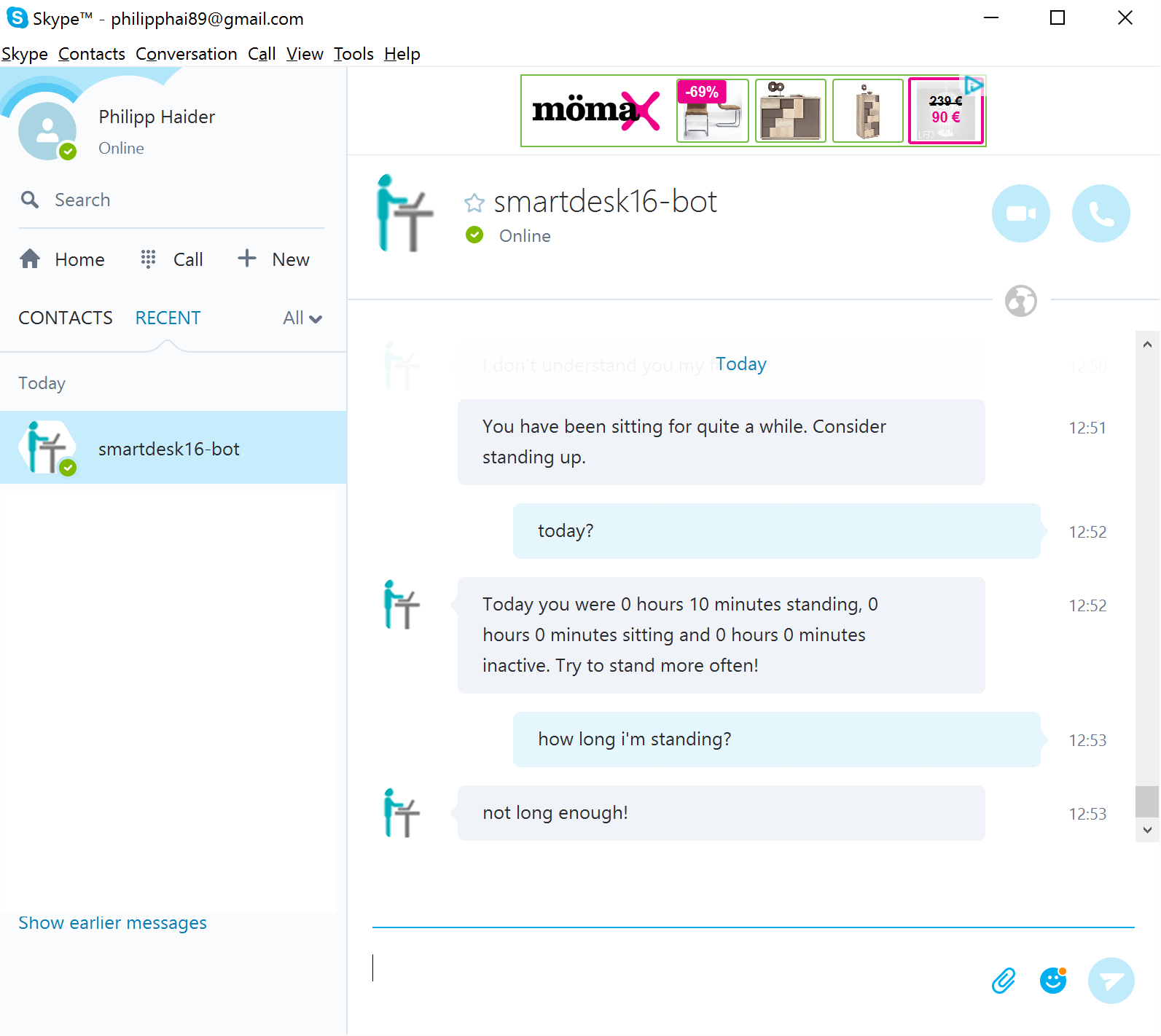


Zusätzlich gibt es noch eine Einstellungsseite in der die folgenden Einstellungen vorgenommen werden können:

* Schwellwert der Höhe, ab wann der Benutzer steht
* Tagesziele der Stehzeit
* ID des Devices dieses Benutzers (falls man von einem Multi-User-System ausgeht)



Eine weitere innovative Interaktionsmöglichkeit bietet der ebenfalls entwickelte Chat-Bot. Damit kann der Benutzer benachrichtigt werden, falls dieser schon zu lange am Stück sitzt. Es ist aber auch möglich, die Zeiten des aktuellen Tages vom Bot aktiv durch einen einfachen Befehlt abzufragen. Nachfolgend der Chat-Bot in Skype.



# Ergebnisse

Das Projekt selbst verlieft durchwegs positiv. Es wurden nicht nur alle Mussziele, sondern auch einige Nice-To-Haves umgesetzt. Ein Großteil des Gesamtsystems konnte problemlos in der Microsoft Azure Cloud betrieben werden. Der Arduino-Sensor selbst braucht eine Verbindung zu einem PC, auf dem eine Applikation läuft, die dann die Daten in die Cloud überträgt.

Wünschenswert war es, vollkommen ohne zusätzlich Applikation auf einem PC auszukommen. Jedoch verhinderten zwei Punkte diese Ziele. Zum ersten soll festgestellt werden ob der Benutzer überhaupt am Arbeitsplatz ist. Anstelle eines Bewegungssensors oder ähnlichem kann die Applikation am PC sehr leicht feststellen ob der Benutzer aktiv ist oder nicht. Zum zweiten ist die Azure IoT Hub Client Bibliothek zu groß für den eingesetzten Arduino Uno.

Eines der aufwändigsten Arbeitspakete war die Verarbeitung der Sensordaten mit einem Stream-Analytics Job in der Cloud. Hier wird deklarativ, ähnlich einer SQL-Abfrage, die Ereignisverarbeitung definiert. Das war schlussendlich schwieriger als erwartet. Eine robuste Ereignisverarbeitung, die mit unregelmäßigen, unterschiedlich schnellen Ereignissen zurechtkommen soll, ist einigermaßen schwierig. Außerdem muss dieser Job mit Referenzdaten versorgt werden. Z.B. der Grenzwert ab welcher Höhe der Benutzer steht, oder welche Sensoren online sein sollten, damit eine Offline-Erkennung gemacht werden kann.

Schlussendlich konnten aber alle Herausforderungen bewältigt werden. Bei den ersten Tests im realen Umfeld fiel auf, dass diese Auswertungen tatsächlich motiviert mehr zu stehen und somit seiner Gesundheit einen Gefallen tut. Die Vorbereitungen das System Multi-User fähig zu machen und in der Firma zu installieren laufen bereits.