IotWeight Android App

Ron Altboum 039563838 ron018@gmail.com

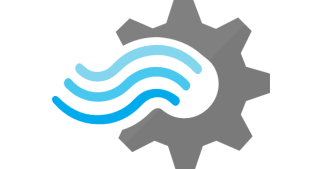
Bar Angel TODO

Ramy Lerner TODO

Data Flow



**IoT hub**



**Stream Analytics**



**Android App**

**Raspberry Pi**

**DRP Protocol**

Overview

In this project, we turned a typical digital bathroom scale I (Ron) had, into a scale that's connected to the internet. The user's weighs are stored in the Azure cloud in a SQL database. This allows the user to weigh himself/herself and keep track of his/her weight history from any such IotWeight device in the world. The user can use an IotWeight weight in his home, in the local clinic, the local mall etc'.

The user doesn't have to buy an IotWeight. All the user needs to do is to download the IotWeight android app, and this allows him to use any IotWeight in the world. Of course the user can also purchase an IotWeight to put in his home.

The Android App

The app is a Xamarin.Android app which uses Azure services such as SQL database, IoTHub , Stream Analytics and Authentication via Facebook. The mobile backend is a javascript backend managed by Azure. The mobile backend url is:

<http://iotweight.azurewebsites.net>

The client side is Xamarin.Android (C#).

User Profile

The app uses Facebook login. In the first time the user uses the app, he is presented with a Facebook login screen where the user enters the email he enrolled to Facebook with, and his Facebook password. Then, a unique id (called sid) is generated by Azure for the user based on his entered email address. This sid is stored in a SQL database in the cloud , and is used to uniquely identify the user. By doing the authentication via Facebook, we avoid storing email addresses and passwords in our database, and therefore this implementation is safer than the alternative (as explained in the following online book by Microsoft developer:

<https://adrianhall.github.io/develop-mobile-apps-with-csharp-and-azure/chapter2/custom/>

). The user can add data to his profile if he wishes by entering his height, and then he will be able to use the BMI calculator in the app (more on that later).

The users profiles are stored in a SQL table called UsersTable that contains the following fields:

string UniqueUsername - which is the previously mentioned sid

float height - optional

Start Weight

When the user wishes to weigh himself, he presses the Start Weight button. Then he needs to scan the barcode which is on the Raspberry. Once the android application has the barcode, it sends a query to a SQL table in the cloud named RaspberryTable. The table has the following fields:

string QRCode - barcode on the Raspberry

string IPAddress - IP address of the Raspberry

The query returns the IP Address of the Raspberry (the Raspberry program previously inserted this information to RaspberryTable, and keeps it updated).

Then, the android app sends to that IP address a message in the format of the DRP protocol we designed, in which it lets the Raspberry know that the user wishes to weigh himself, and sends the Raspberry the user's sid which uniquely identifies the user of the android app (see the section on the DRP protocol for more details). This way the Raspberry knows who is the user that wishes to weigh himself. After the user weighs himself, the Raspberry sends a DRP message to the android app with the weight, and the weight is displayed on the user's screen in the android app. In addition, the Raspberry sends a message to the IoT hub. The message contains the user's unique id (sid), the weight, and the current Date. The message is transferred from the IoT hub to the SQL table named weighTable via **Azure's Stream Analytics**.

The weighTable has the following fields:

string username - user's unique id (sid)

float weigh

DateTime createdAt - Date of weigh

Weight History

The user can view his weight history by pressing the Weight History button in the app. He can choose the display format (List or Graph), and the time period (Last month, Last 3 months, Last 6 months). The android app retrieves the previous weighs via a SQL query to the weighTable mentioned earlier.

Calculate BMI

The user can find out his BMI based on his last weight by pressing the Calculate BMI button in the android app. At the first time the user calculates his BMI, he is asked to enter his height. The user's height is inserted to the SQL usersTable mentioned earlier and saved there. The next times the user wishes to find out his BMI, he will not be asked to enter his height, and the height will be retrieved from the SQL table.

In addition to the BMI, the application also lets the user know to which BMI category he belongs to, where Normal ranges between 18.5 and 25. It also gives a link to the Wikipedia page which contains more details on all the BMI categories.

אתגרים ודרכי פתרון באפליקציה

אתגר גדול היה בבחירת הפרוייקט, בחירת דרך המימוש שלו ובחירת הרכיבים שצריך להזמין. חשבנו על מספר דרכי מימוש, ביררנו על כל דרך באינטרנט, עד שהחלטנו על הדרך הטובה ביותר.

רוב האתגרים היו בהתחלה. היו הרבה בעיות של התקנת כל הספריות של azure ו xamarin ב visual studio. הבעיות נבעו משינויים בגרסאות השונות והתנגשות של ספריות. לאחר שיטוט באינטרנט וניסוי של מספר פתרונות הכוללים מחיקת גרסאות מסוימות של ספריות ושל visual studio, הצלחנו להתגבר על בעייה זו.

בעייה נוספת הייתה בבאגים שהיו קיימים בספריות של מייקרוסופט בצד השרת הכתוב בשפת C#. הייתה בעייה מפורסמת שלא היה ניתן להוסיף טבלה חדשה לבסיס הנתונים SQL, (זה נקרא Table Controller). לאחר שיטוט ממושך באינטרנט, התברר כי זו בעייה ידועה והיו הרבה הצעות לפתרונות שלא עבדו. שלחתי שאלה בפורום של מייקרוסופט ושם נאמר לי על ידי עובדי מייקרוסופט שזו בעייה בגרסה החדשה של visual studio, והמליצו לי להתקין גרסא ניסיונית חדשה (VS 2017 Preview). גם פתרון זה לא עבד. לבסוף, החלטתי לממש את צד השרת בשפת Node.Js, ושם הבעייה לא הייתה קיימת. הוספת טבלה חדשה מתבצעת בקלות על ידי Easy Tables בפורטל של Azure.

**בר TODO:** לכתוב על אתגרים ודרכי פתרון ברספברי. למשל: האתגרים עם הסנסור שבתחילה לא היה מדוייק, ושתרגמת את הספרייה שכתובה לארדואינו לרספברי פיי. ולאחר מכן העובדה שלא עבד לנו החלק שסיוון בנה לנו. והפתרון הוא שבסוף הזמנו מספארקפאן את הקומבינטור. לכתוב על כל ה QA שעשית בשביל לגלות איפה הבעייה.