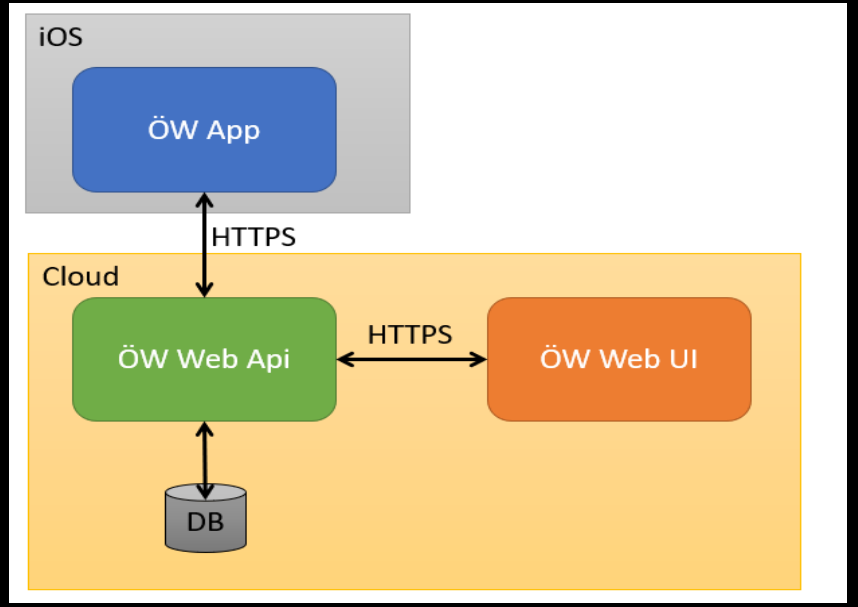
ÖSSUR WALKABOUT

APP PROOF OF CONCEPT

HANNES SVERRISSON

OBJECTIVE

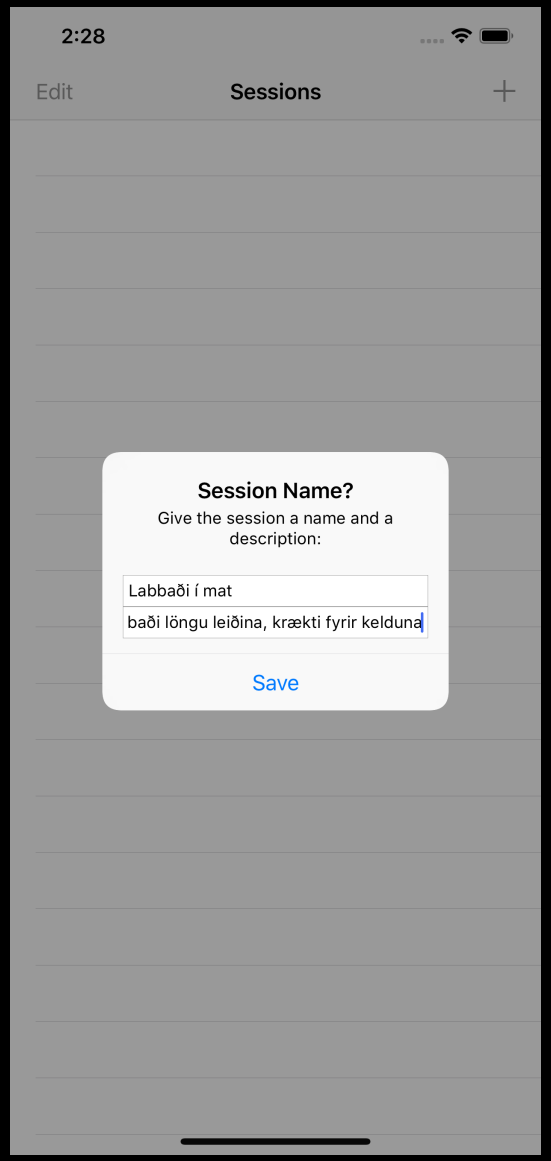
- Design a system to analyse walking motion.
- Consists of three parts:
  - APP
  - API + Database Storage
  - WEB UI

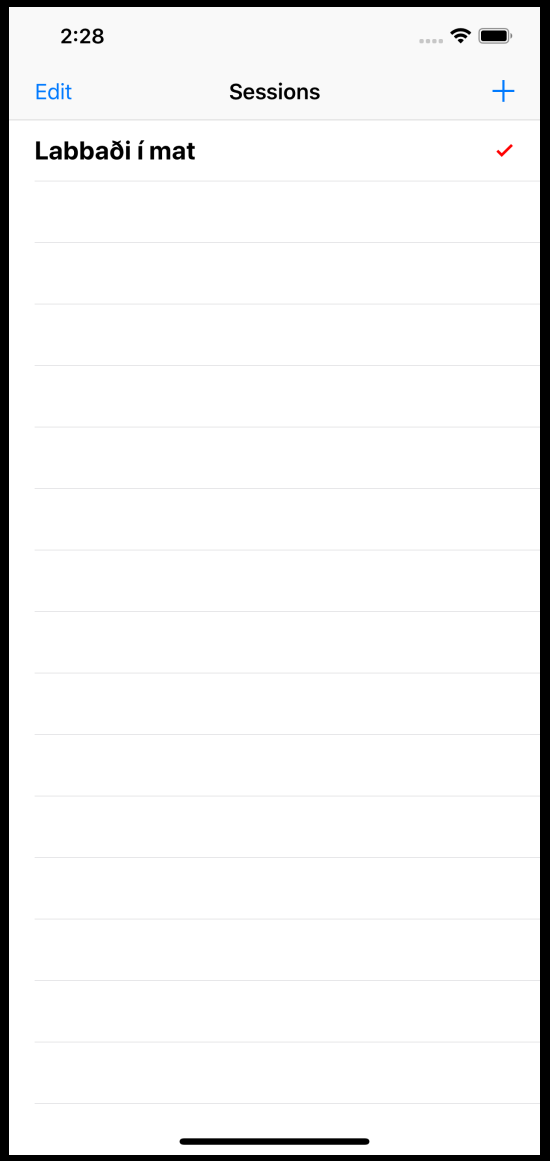


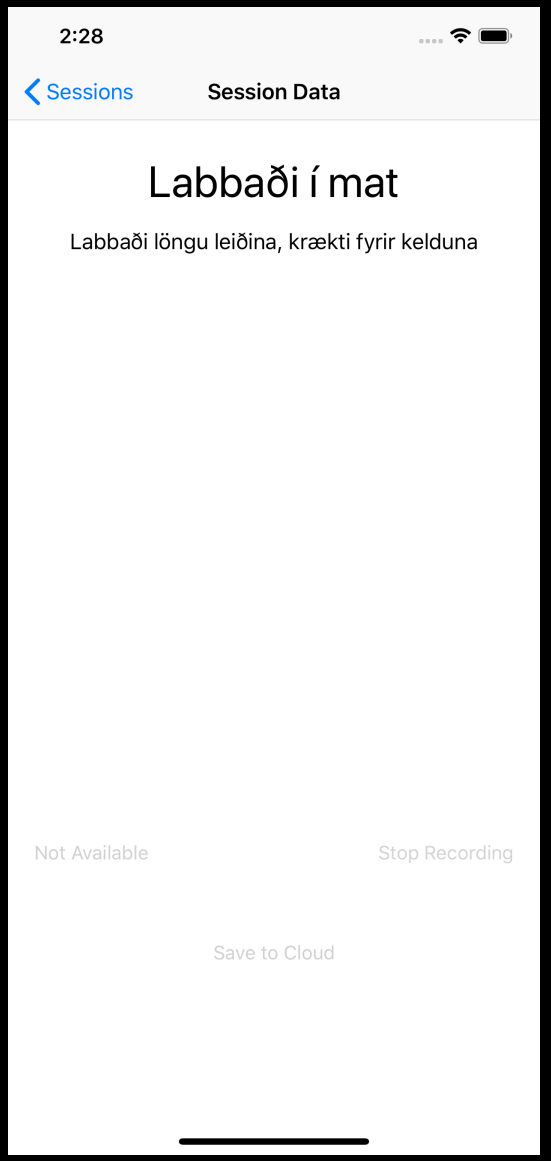
DESIGN

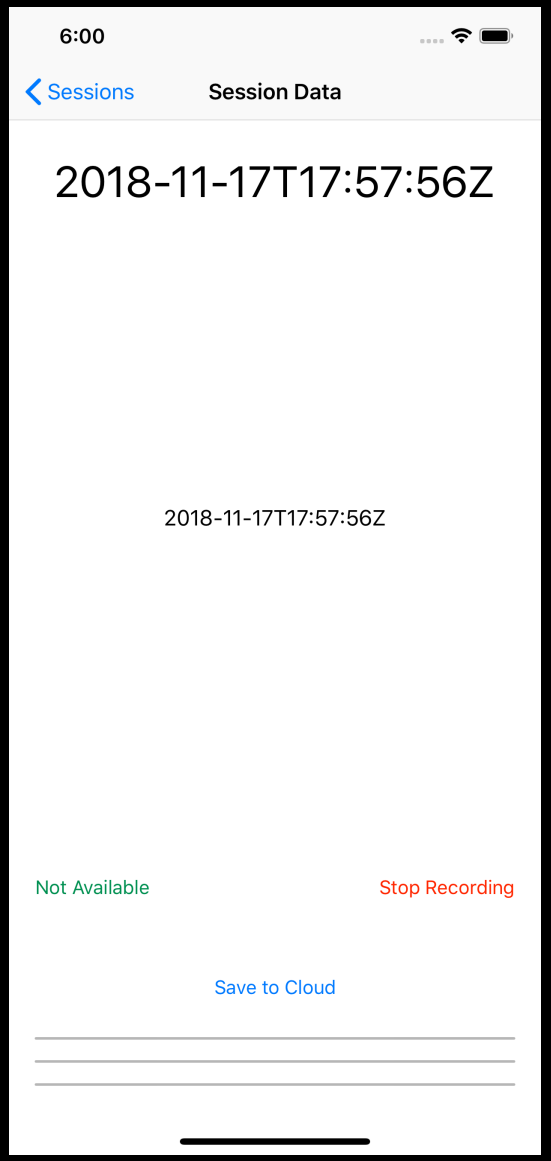
- Walkabout APP
  - Allows the user to create sessions and store data.
  - Sends data to API server.
  - SQLITE3 database for storage.
- Walkabout Server
  - Nodejs express server.
  - SQL Server 2017 from microsoft for linux.
  - Docker used to create server container.
  - Docker compose to set up server and SQL Server db.
- Walkabout Web Client
  - React framework.
  - Redux backend for data storage and flow.
  - Note: not yet implemented.

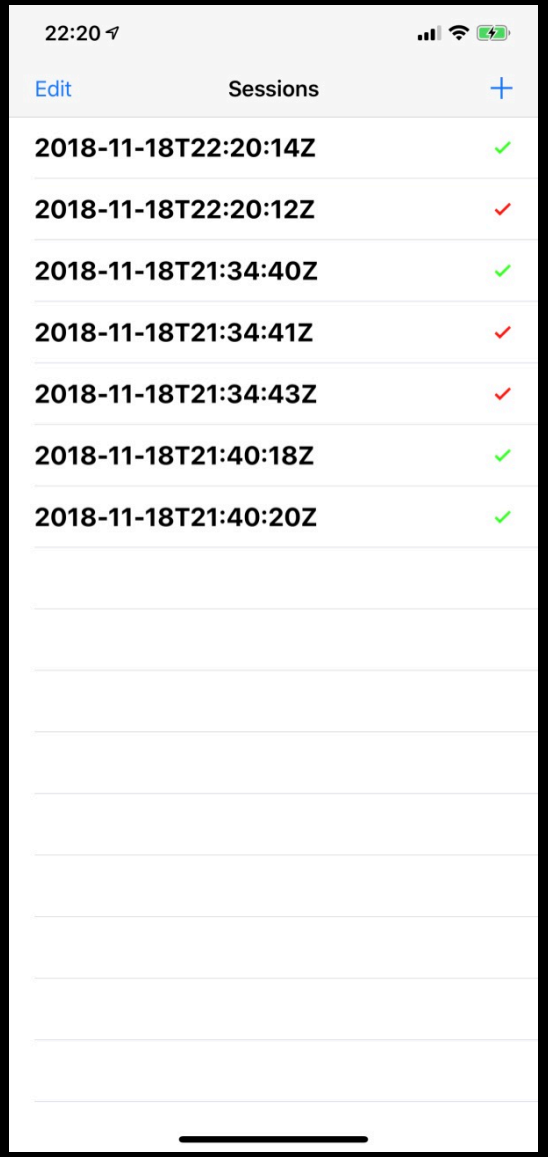
WALKABOUT APP

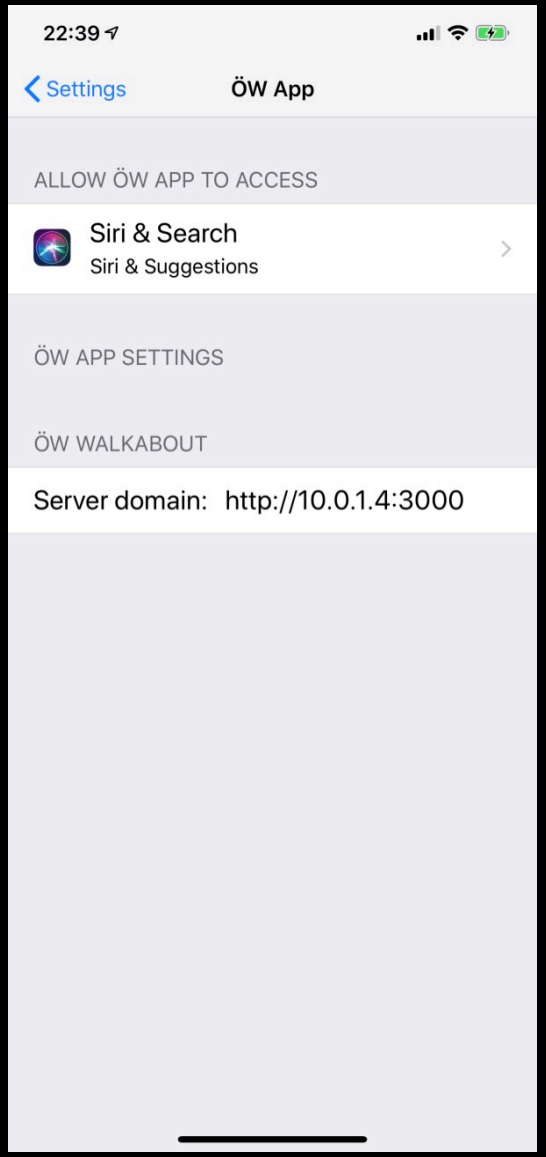

2:28
Edit
Sessions
Session Name?
Give the session a name and a description:
Labbaði í mat
baði löngu leiðina, krækti fyrir kelduna
Save


2:28
Edit
Sessions
Labbaði í mat


2:28
Sessions
Session Data
Labbaði í mat
Labbaði löngu leiðina, krækti fyrir kelduna
Not Available
Stop Recording
Save to Cloud


6:00
Sessions
Session Data
2018-11-17T17:57:56Z
2018-11-17T17:57:56Z
Not Available
Stop Recording
Save to Cloud


22:20
Edit
Sessions
2018-11-18T22:20:14Z
2018-11-18T22:20:12Z
2018-11-18T21:34:40Z
2018-11-18T21:34:41Z
2018-11-18T21:34:43Z
2018-11-18T21:40:18Z
2018-11-18T21:40:20Z


22:39
Settings
ÖW App
ALLOW ÖW APP TO ACCESS
Siri & Search
Siri & Suggestions
ÖW APP SETTINGS
ÖW WALKABOUT
Server domain: http://10.0.1.4:3000

WALKABOUT APP

- ALLOWS THE USER TO CREATE SESSIONS AND STORE DATA.
- SENDS DATA TO API SERVER.
- CODABLE PROTOCOL USED FOR JSON SERIALIZATION.
- SQLITE3 DATABASE FOR STORAGE.

model.swift

```swift
// Client stored iPhone info.  The ID is GUID.
struct Client: Codable, CustomStringConvertible {
    let id: String
    let at: Date
    let name: String
    let type: String
    let systemVersion: String

    var description: String {
        return "(id: \(id), at: \(at), name: \(name), type: \(type), version: \(systemVersion))"
    }
}

// Session stores name („Labbaði í mat“) and description („Labbaði löngu leiðina, krækti fyrir kelduna“)
struct Session: Codable, CustomDebugStringConvertible {
    let id: Int
    let clientID: String
    let at: Date
    let name: String
    let description: String
    let saved: Bool

    var debugDescription: String {
        return "(id: \(id), clientID: \(clientID), at: \(at), name: \(name), "
            + "description: \(String(describing: description)), saved: \(saved)"
    }
}

// Metadata stores x,y,z accelerometer data collected.
struct Metadata: Codable {
    let id: Int
    let sessionID: Int
    let at: Date
    let accX: Int
    let accY: Int
    let accZ: Int
}

// Payload is used to transfer data to server
struct Payload: Codable {
    let client: Client
    let session: Session
    let data: [Metadata]
}
```

WALKABOUT APP

- SQLITE3 DATABASE FOR STORAGE.
- ADDED SYSTEMVERSION IN CLIENT TABLE FOR STORING, I.E. IOS 12.1.

```
BEGIN TRANSACTION;

-- CREATE DATABASE Walkabout;
-- DROP TABLE IF EXISTS Client;
-- DROP TABLE IF EXISTS MSession;
-- DROP TABLE IF EXISTS Metadata;

-- Table: Client
CREATE TABLE IF NOT EXISTS Client
(
ID CHAR(36) NOT NULL PRIMARY KEY,
At datetime,
Name VARCHAR(45),
Type VARCHAR(30),
SystemVersion VARCHAR(30)
);

-- Table: Session
CREATE TABLE IF NOT EXISTS MSession
(
ID INTEGER NOT NULL PRIMARY KEY,
ClientID CHAR(36) NOT NULL REFERENCES Client(ID),
At datetime,
Name VARCHAR(55),
Description TEXT,
Saved INTEGER
);

-- Table: Metadata
CREATE TABLE IF NOT EXISTS Metadata
(
ID INTEGER NOT NULL PRIMARY KEY,
SessionID INTEGER NOT NULL REFERENCES MSession(ID),
At datetime,
AccX INTEGER,
AccY INTEGER,
AccZ INTEGER
);

COMMIT TRANSACTION;
```

WALKABOUT APP

- DATA SENT SYNCHRONOUSLY TO CONTINUE IN THE BACKGROUND IF THE USER CLOSES THE APP.
- REPLICATION CONFLICT
  - THE SERVER RESPONDS IF THE PAYLOAD WAS SUCCESSFULLY STORED, IF NOT THEN THE CLIENT DATA IS NOT DELETED FROM THE LOCAL DATABASE.
  - SAME APPLIES IF THE APP CRASHES OR RUNS OUT OF TIME, THE CALLBACK RETURNS FALSE OR NOTHING.

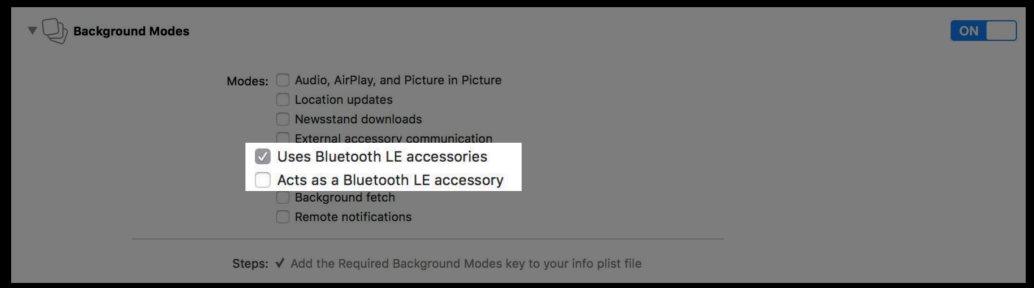
```swift
// Sends data to the cloud for the given session and deletes the data if successful
func sendToCloud(session: Session, callback: @escaping (Bool) -> ()) {
    os_log("Send data to API server in background if needed", type: .info)
    DispatchQueue.global().async {
        let networkClient = NetworkClient.shared
        self.backgroundTaskID = UIApplication.shared.beginBackgroundTask(withName: "Finish Network
            // End the task if time expires.
            DispatchQueue.main.async {
                callback(false)
            }
            UIApplication.shared.endBackgroundTask(self.backgroundTaskID!)
            self.backgroundTaskID = UIBackgroundTaskIdentifier.invalid
        }

        // Create the Payload and send the data to the API Server synchronously.
        if let client = self.readClient(), let meta = self.readMetadataFor(sessionID: session.id) {
            let payload = Payload(client: client, session: session, data: meta)
            let encoder = JSONEncoder()
            if let data = try? encoder.encode(payload) {
                // Send the data synchronously!
                let result = networkClient.sendDataSynchronously(data)
                DispatchQueue.main.async {
                    callback(result)
                }
            }
        }

        // End the task assertion.
        UIApplication.shared.endBackgroundTask(self.backgroundTaskID!)
        self.backgroundTaskID = UIBackgroundTaskIdentifier.invalid
    }
}
```

WALKABOUT APP

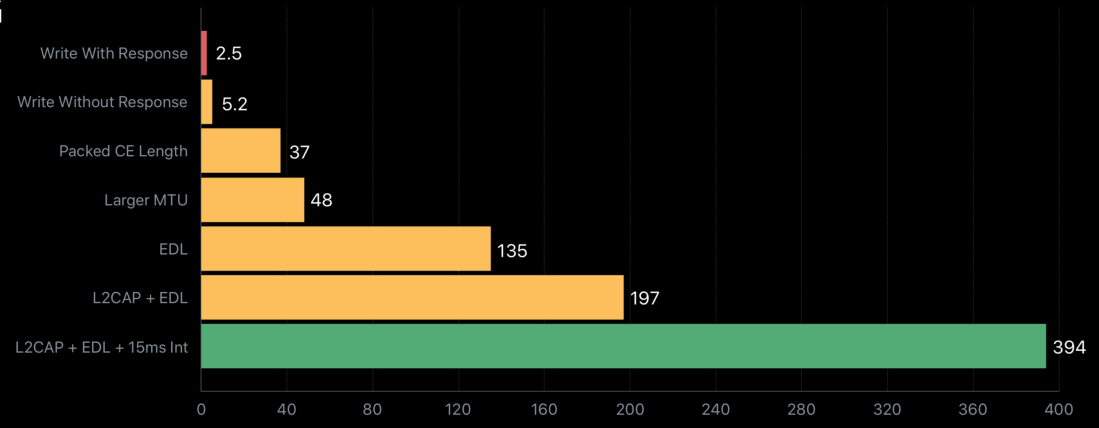
- Background data collection on iOS
  - After a user exits the app, it has 5 seconds to finish any work in the background before being shut off.
  - This can be addressed by registering for background work and monitoring Bluetooth LE devices.
  - Might be Possible to apply for using the CMMovementDisorderManager that collects motion data on the user in the background over long time. Used for Parkinsons tremor & kinesias analysis.

Backgrounded Apps

iOS Apps can continue using Core Bluetooth in the background



Throughput (kbps)



WALKABOUT APP

- SQLite vs. CoreData
  - Coredata is kind of a blackbox with no access to the code base.
  - Changes to the database in an app update can be challenging.
  - Strongly typed is an advantage as is support by Apple.
  - Coredata uses SQLite database.
  - Coredata can be called from other than the main thread.
- SQLite is a c-codebase which can be tricky to communicate with.
- There are some open source packages to allow interactions with it in a more typesafe and swiftier way.
- All code is accessble and thus easier to understand what it going on and the dataflow.

WALKABOUT APP

- Challenges
  - With more time - implement the sqlite with type safety instead of using schema (schema is valuable in development for quick changes).
  - Improved user interface with more icons and graphics would improve the look of the app.
  - Realtime updating the Acc data on a Trigraph chart.
  - Complete CRUD not finished in time.
  - Metadata primary key.

WALKABOUT SERVER

Walkabout Server

- Walkabout Server
  - NodeJS Express server.
  - SQL Server 2017 from Microsoft for Linux.
  - Docker used to create server container.
  - Docker compose to set up server and SQL Server db.

WALKABOUT SERVER

- Challenges
  - Session keyword, thus Session table renamed to MSession.
  - Lacking support for Nodejs Express integration.
  - With more time would prefer to implement with .NET core or use Postgres sql db.
  - Complete CRUD not finished in time.
  - Currently a payload can only be sent once to the server. Next payload will collide with a previous Client.

walkabout.sql

```sql
-- Walkabout Tables

CREATE TABLE Client
(
  ID char(36) NOT NULL PRIMARY KEY,
  At datetime NOT NULL,
  Name varchar(50) NOT NULL,
  Type varchar(50) NOT NULL,
  SystemVersion VARCHAR(30)
);
GO

CREATE TABLE MSession
(
  ID int NOT NULL PRIMARY KEY,
  ClientID char(36) NOT NULL,
  At datetime NOT NULL,
  Name varchar(100),
  Description text,
  FOREIGN KEY (ClientID) REFERENCES Client(ID)
);
GO

CREATE TABLE Metadata
(
  ID int NOT NULL PRIMARY KEY,
  SessionID int NOT NULL,
  At datetime NOT NULL,
  AccX int NOT NULL,
  AccY int NOT NULL,
  AccZ int NOT NULL,
  FOREIGN KEY (SessionID) REFERENCES MSession(ID)
);
GO
```

WALKABOUT SERVER

- Replication conflict
  - The **server** should always check if session is already stored, before trying to store anything and respond to the client. If data is seen before the **server** should verify the data is identical and then send a success response to the **client.**
  - The **server** should not respond to the **client** until the database has finished storing the data with a successful return and then send a success response to the **client.**

Walkabout Web Client

- React Framework.
- Redux backend for data storage and flow.
- Note: Not yet implemented.

GENERAL PROJECT IDEAS

DATA SAFETY

- HTTPS FOR ALL COMMUNICATIONS OVER THE NETWORK.
- CLIENT SHALL REGISTER AT INITIAL START AND SEND A PUBLIC KEY TO THE SERVER.
- ALL DATA SENT OVER NETWORK SHALL BE ENCRYPTED, SERVER CAN USE THE INITIAL KEY TO DECRYPT.
- ALL DATA STORED IN DATABASES SHALL BE ENCRYPTED WITH A PASSWORD AND A SALT. THUS, IF THERE IS A DATA BREACH, IT IS STILL COSTLY TO BREAK THE ENCRYPTION AS EACH SALT IS UNIQUE.
- ALLOW THE USER TO TURN ON: LOG IN WITH FACE ID OR TOUCH ID ON THE PHONE.

OTHER IDEAS

- Using and contributing to Researchkit. Apple framework for medical research.
- Using the Apple watch to gather more data and could also be attached to limbs to get more measurement points on the rotation of the limb, as well as accelerometer data.
- Use bluetooth le to wake up app, for sending data in the background to the server.
- Utilising Healthkit for more data on the user.
- Utilising sportkit, i.e. workoutkit, for possible modes of running in the background.

QUESTIONS?

THANK YOU