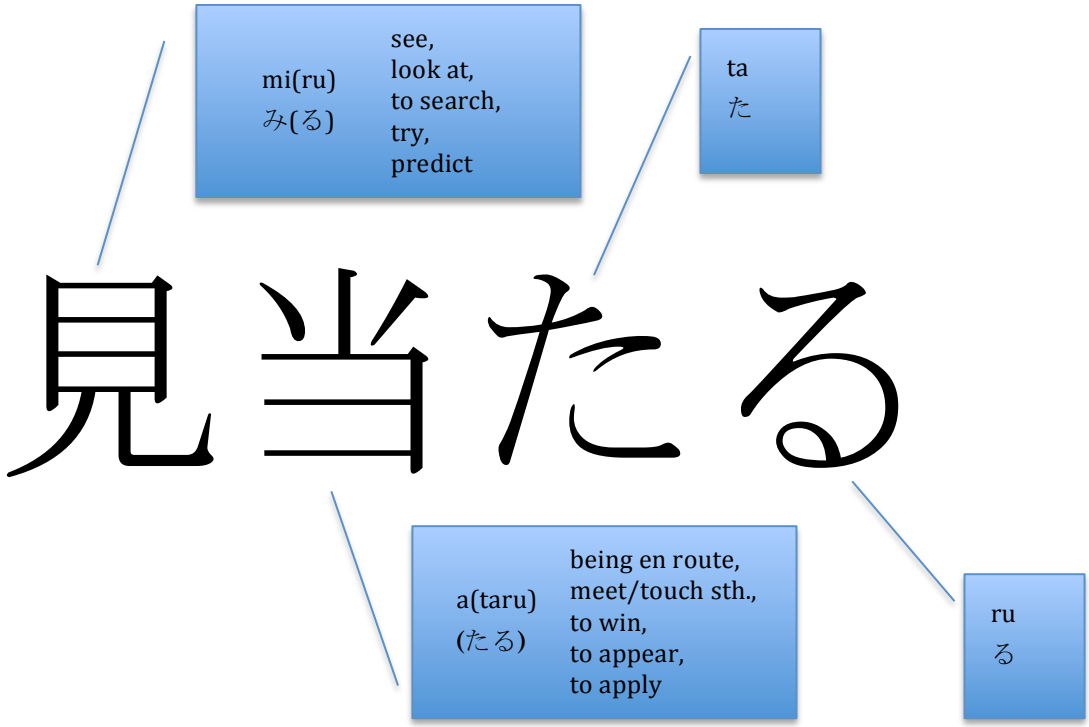
Miataru



Document version 1.3 / 20. August 2013 / Daniel Kirstenpfad

Miataru Specification

Miataru?

When Google closed it's Latitude service it suddenly became apparent to me how much I used that service and how many processes in daily live are already relying on the location of persons.

I can give some examples:

- Being able to tell where family members are right now helps a lot when organizing your day and work – it minimizes communication overhead a lot. How many times did you call people "where are you now?"
- It's a blessing in home automation to have that meta data to control systems. Think of your house knowing you are approaching and therefore increases the heating on it's own so it's warm when you arrive or it lowers the heating while you are away?
- Proximity Alerts help to avoid surprises (for all those who don't like them)
- Knowing the metrics of your day, week, month, year makes a lot of sense when you think about lowering transportation costs and personal energy consumption
- You're a photographer and you want to geo-tag your pictures after the shooting or you found a nice spot and you want to find it again in the future?

Miataru or 見当たる is Japanese and means "be found" or "to come across" and it's meant to be a set of tools to allow the user to track locations and choose how to work with the data as well as how data is stored if at all.

What is it supposed to be?

Miataru is:

- Server
    - A server that stores location data (sent in using JSON and a HTTP POST request) for a given amount of seconds
        - After the pre-configured amount of seconds the location data for a user is lost
    - A server that makes available a given amount of location history, if the user wants that (not by default)
    - A server that offers through simple GET requests the location of a particular device and eventually a location history of configurable length/duration
    - The server component will be released as open source software (BSD license, free-for-all) – the protocol itself will be documented so that any self-written server can be used as well

- Client
  - The client software is preferably installed on a mobile device like an iPhone or an Android phone. It's constantly recording and reporting location updates, configurable to the likings of the user. The main featureset of a mobile client is:
    - Runs continuously and reports locations to the Miataru server depending on the following strategy patterns:

Miataru Client App State	Battery / Power	Reporting Strategy
Running In Background	On Battery >=50% capacity (configurable)	**Best Effort**
Running in Background	On Battery <50% capacity (configurable)	**Saving Power**
Running in Background	Power Supply connected, loading <70% capacity (configurable)	**Best Effort**
Running in Background	Power Supply connected, loading >=70% capacity (configurable)	**Best Precision**
Running in Foreground	Power Supply connected, loading >=70% capacity (configurable)	**Best Precision, Best Actuality**
Running in Foreground	Power Supply connected, loading <70% capacity (configurable)	**Best Precision**
Running in Foreground	On Battery >=50% capacity (configurable)	**Best Precision, Best Actuality**
Running in Foreground	On Battery <50% capacity (configurable)	**Best Precision**

    - Strategies detail:
      - **Best Effort** – only minimum precision to not power up the GPS. Rather using GSM cell and WLAN location data to determine position and position changes. Frequently the App increases the requested location precision to find force an update (should be configurable)
      - **Saving Power** – record location changes through GSM/WLAN, do not power up GPS. Do not report locations to the Miataru server. Instead record a location history on the device to not power up the GSM modem unnecessarily
      - **Best Precision –** power up the GPS frequently and report position changes according to a higher requested location precision. Reporting should take place at a maximum rate of 30 seconds (configurable)
      - **Best Precision, Best Actuality –** by running the app in the foreground the user signals that he wants to have the best precision possible and the most frequent updates to the server. Whenever more than 50m move happened a position change is stored on the device and every 10 seconds (configurable) transmitted to the Miataru server.

  - The Client User Interface is as simple as possible and allows the user to manage a list of other devices he can display on a map view as well as all the basic configuration steps.
  - A first-run set-up is included so a new user can immediately start using the service → a pre-defined Miataru server is available in the UI

Miataru Protocol Specification

To report and retrieve location data a very simple protocol is being used. This protocol transports JSON formatted data using HTTP / HTTPS.

Posting Location Updates

A single location update request:


```
{
    "MiataruConfig": {
        "EnableLocationHistory":"False",
        "LocationDataRetentionTime":"15"
    },
    "MiataruLocation": [{
        "Device": "7b8e6e0ee5296db345162dc2ef652c1350761823",
        "Timestamp": "1376735651302",
        "Longitude": "10.837502",
        "Latitude": "49.828925",
        "HorizontalAccuracy": "50.00"
    }]
}
```


This simple data structure is posted onto the Miataru server service URL.

For example: http://service.miataru.com/UpdateLocation

It contains the Device Identifier. This device identifier is calculated from the name of the device entered by the user and a device unique identifier. On the iPhone the UDID is used on any other platform it could be anything that identifies that device.

These two values are then hashed using the SHA-1 algorithm:

DanielsIphone4s-7b8e6e0ee5296db345162dc2ef652c1350761823

e64a3682ce5a44cff5d9aeaf4c4697c26fa4f977

This SHA-1 hash is then used as the Device value.

The Timestamp value is a plain javascript timestamp when that location update was recorded. You can generate it yourself in javascript: `new Date().getTime();`

The LocationDataRetentionTime is the amount of minutes the server will keep that Location Data before it is removed/deleted automatically.

Longitude and Latitude are the location values themselves and with the HorizontalAccuracy field the precision of the location measurement is described in meters.

When locations where stored locally and need to be posted to the Miataru server according to the client strategies at once you just need to comma-separate the json datasets, starting with the oldest one.

Multiple Location Updates in one call:

```
{
    "MiataruConfig": [{
            "EnableLocationHistory":"True",
            "LocationUpdateRetentionTime":"15"
    },
    "MiataruLocation": [{
            "Device": "7b8e6e0ee5296db345162dc2ef652c1350761823",
            "Timestamp": "1376735651302",
            "Longitude": "10.837502",
            "Latitude": "49.828925",
            "HorizontalAccuracy": "50.00"
    },
    {
            "Device": "7b8e6e0ee5296db345162dc2ef652c1350761823",
            "Timestamp": "1376737022853",
            "Longitude": "10.807502",
            "Latitude": "49.818925",
            "HorizontalAccuracy": "75.00"
    }]
}
```

If the client tells the server to store a location history the server automatically disables the pre-configured data time-out behavior that removes location data after a given amount of time. Nevertheless the user is limited to the server-side pre-configured amount of location history entries (default: 1024). A client shall only post and record multiple locations if the location history is enabled client side. One client location update only corresponds with one device update. If you need to update multiple devices do that in separate calls.

In any case the server will answer back telling the client the result of the request:

```
{
    "MiataruResponse":"ACK",
    "MiataruVerboseResponse":"><)))x>"

}
```

The MiataruResponse field will be in all responses from the server stating if there have been any problems with the request or if everything went as planned. It can have the following values:

MiataruResponse	Description
ACK	Everything went well, the VerboseResponse contains an ASCII fish.
NACK	Something is wrong, find details in the MiataruVerboseResponse

Retrieving Location Information

To retrieve the previously posted device information the client posts the list of requested devices on the Miataru GetLocation URL. For example http://service.miataru.com/GetLocation

The content of that HTTP or HTTPS POST request contains the following JSON formatted data structure:

This example asks for the last known locations of two devices (7b8… and b0d…). The Miataru server answers by sending back the requested locations:

This answer represents two found device location and one unknown device location. It also contains the timestamp of the Location updates and the accuracy of that measurement.

Retrieving Location History

Location History is stored on the server only if the client told the server to do so using the *"EnableLocationHistory"* setting in the Location Update requests.

For transitions of enabling/disabling that functionality: Everytime a Location Update is received by the server with *"EnableLocationHistory=false"* the server removes all stored Location History till that point.

There is a server-side setting that controls up to how many Location Updates the server is storing in the Location History before it removes the oldest one.

To request the Location History of a particular device the client sends the following POST request to the GetLocationHistory service URL.

For example: http://service.miataru.com/GetLocationHistory


```
{
    "MiataruGetLocationHistory": {
        "Device":"7b8e6e0ee5296db345162dc2ef652c1350761823",
        "Amount":"25"
    }
}
```


This request contains the Device ID and the number of Location Updates the client wants to retrieve from the server starting from new to old.

The server will answer back with an Array of LocationUpdates if there are Location Updates, or a List of NoLocations when there are none. If the client requested more Location Updates than there are available the server will return those available. The answer from the server will look like this:


```
{
    "MiataruServerConfig": {
        "MaximumNumberOfLocationUpdates":"1000"
    },
    "MiataruLocation": [{
        "Device": "7b8e6e0ee5296db345162dc2ef652c1350761823",
        "Timestamp": "1376735651302",
        "Longitude": "10.837502",
        "Latitude": "49.828925",
        "HorizontalAccuracy": "50.00"
    },
    {
        "Device": "7b8e6e0ee5296db345162dc2ef652c1350761823",
        "Timestamp": "1376737022853",
        "Longitude": "10.937502",
        "Latitude": "49.858925",
        "HorizontalAccuracy": "5.00"
    }]
}
```

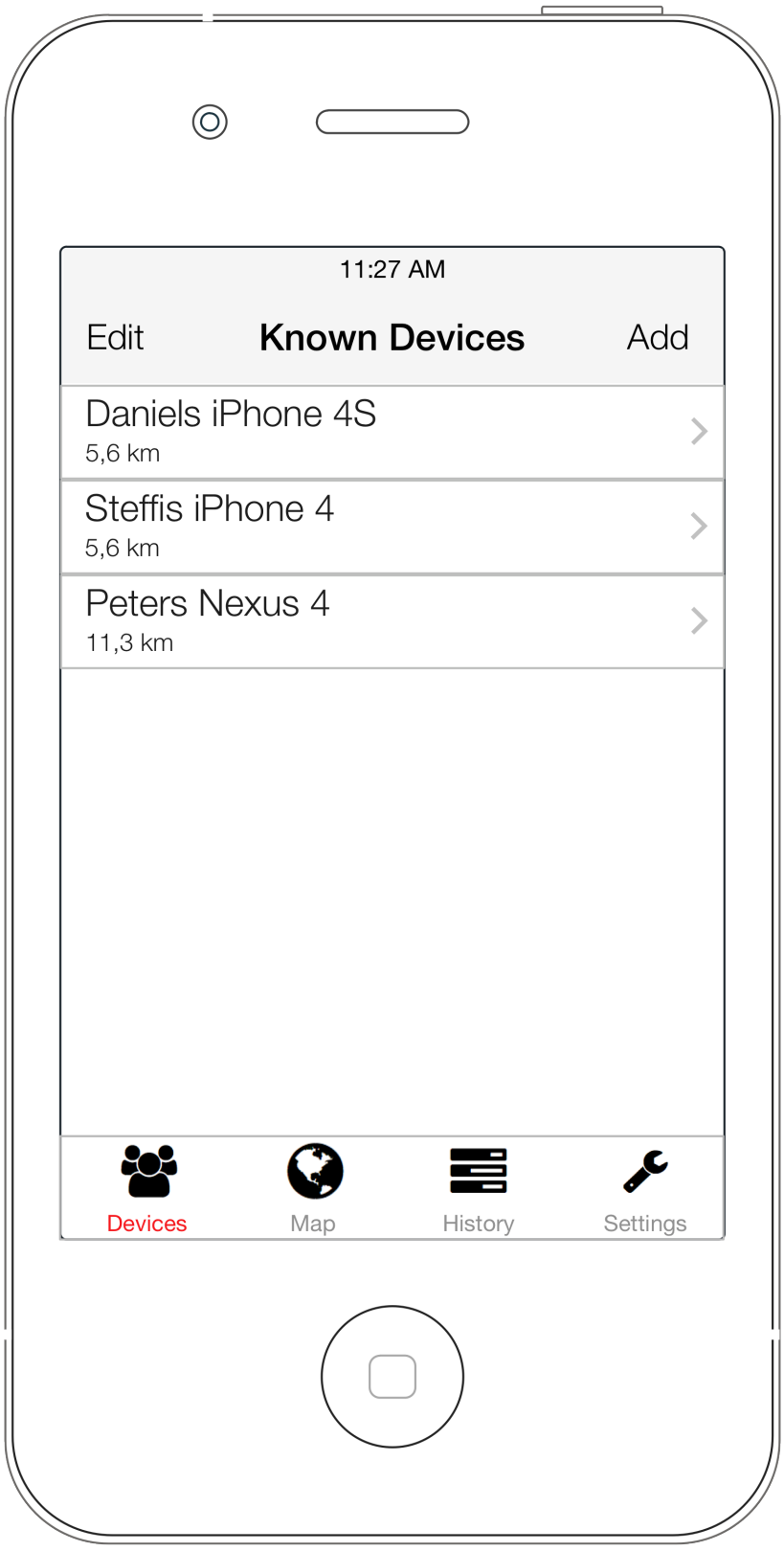

If the server was unable to find a Location History for this device (maybe there is none, or it was not enabled by the client) the result will look like this:


```
{
    "MiataruServerConfig": {
        "MaximumNumberOfLocationUpdates":"1000"
    },
    "MiataruNoLocation": [{
        "Device": " 7b8e6e0ee5296db345162dc2ef652c1350761823"
    }]
}
```

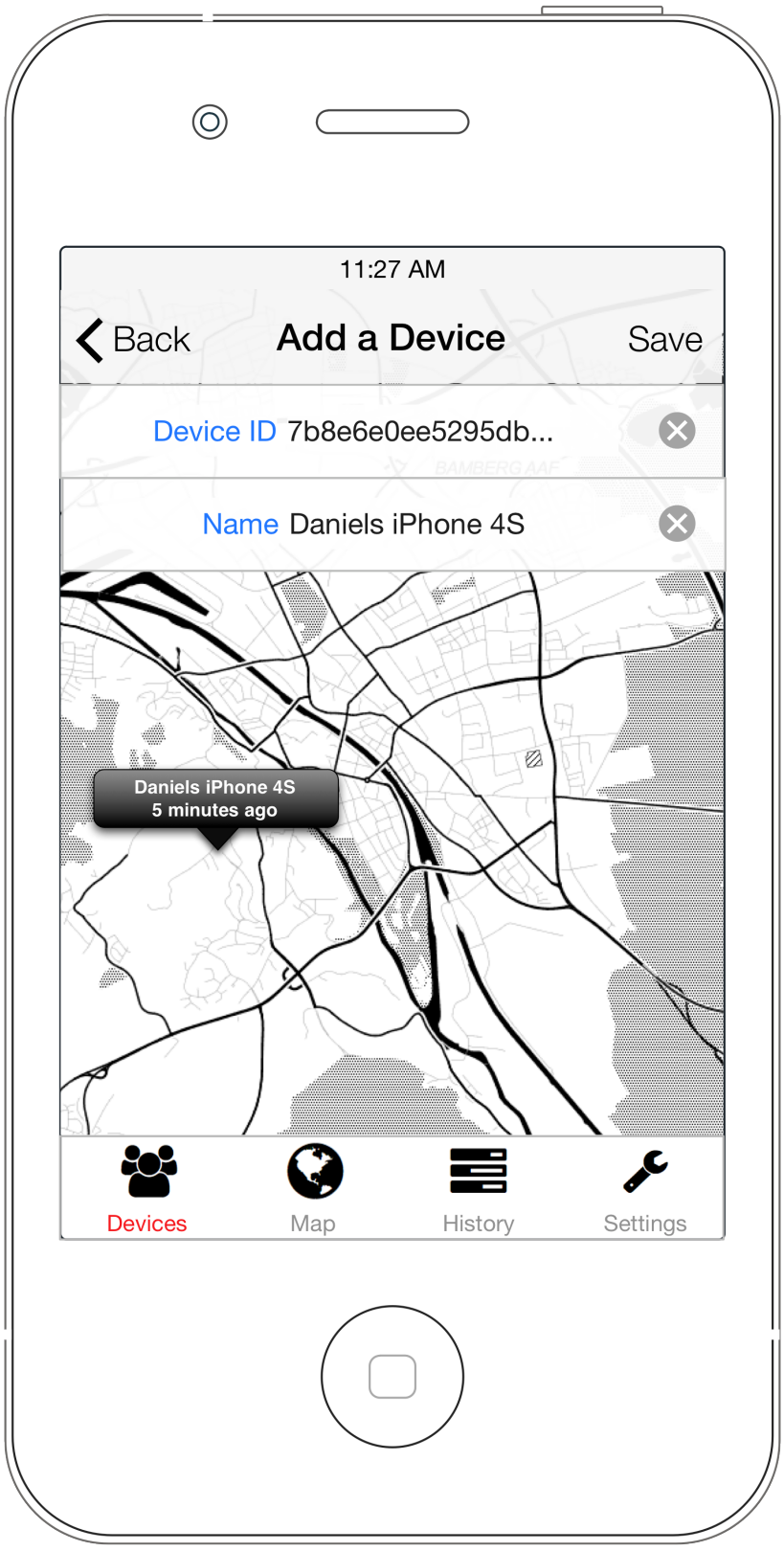
Basic Client UI Mock-Ups / Wireframes

App Launch

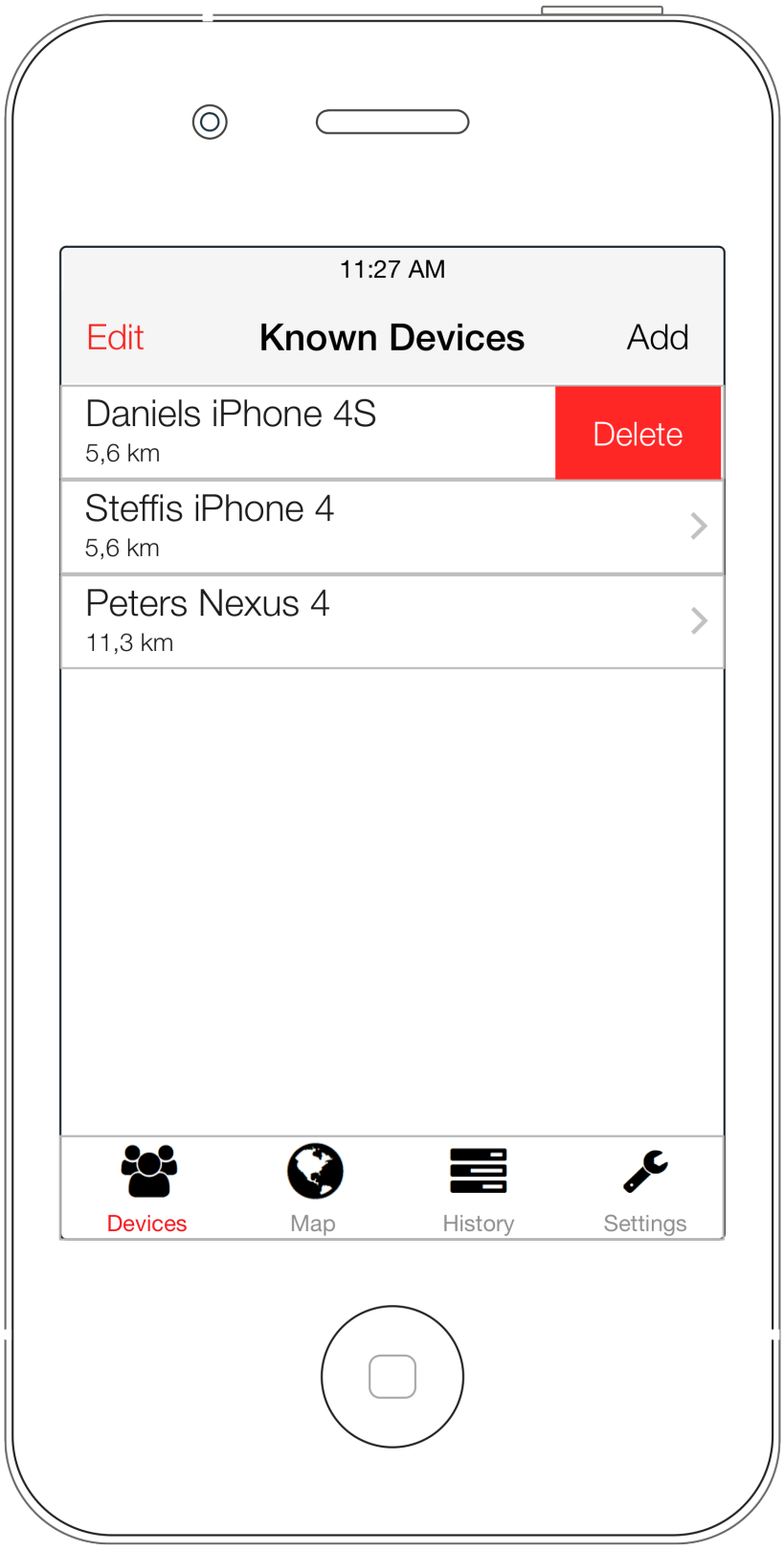
Devices Tab


11:27 AM
Edit
Known Devices
Add
Daniels iPhone 4S
5,6 km
Steffis iPhone 4
5,6 km
Peters Nexus 4
11,3 km
Devices
Map
History
Settings

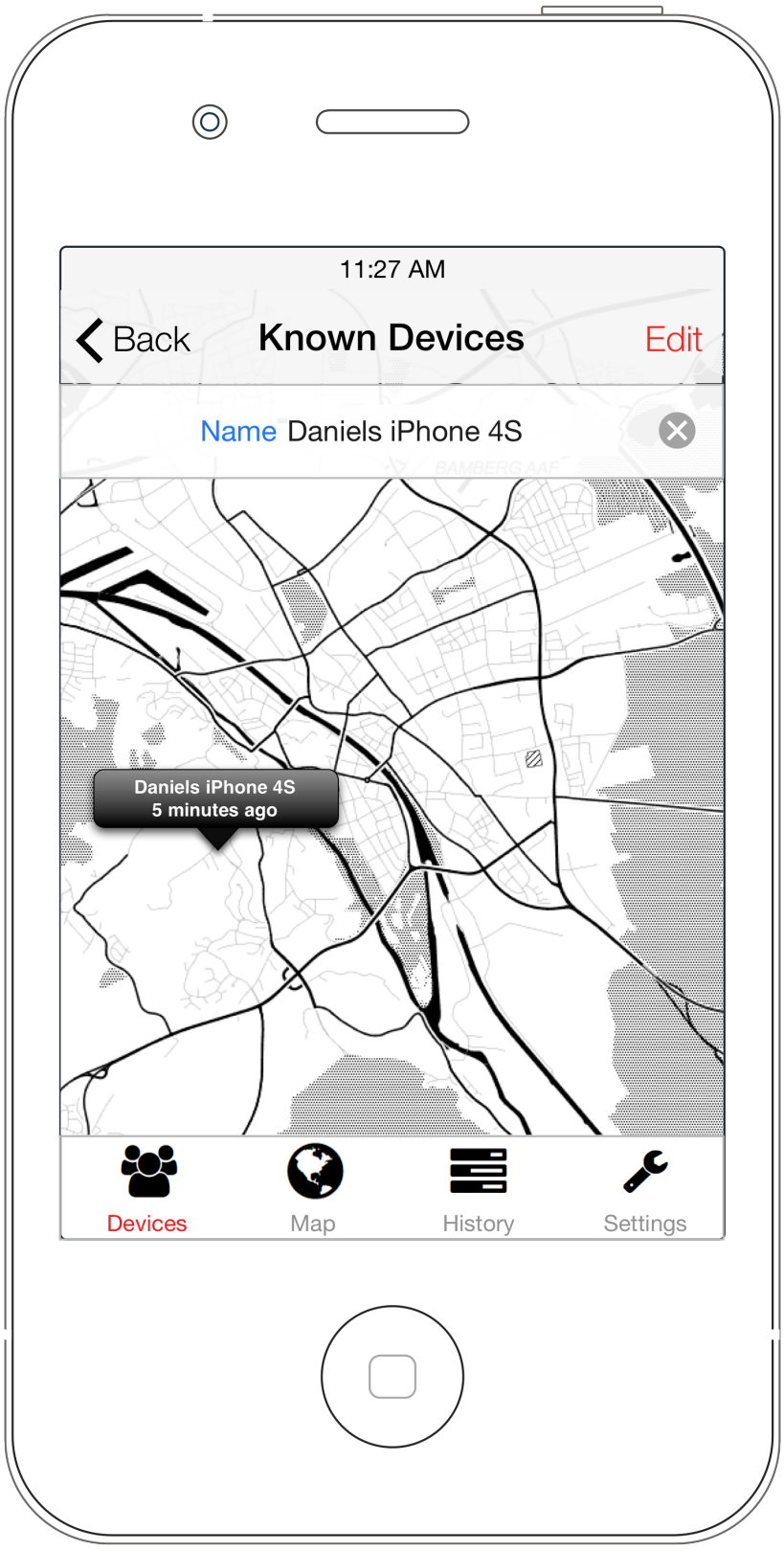
Devices Add


11:27 AM
Back
Add a Device
Save
Device ID 7b8e6e0ee5295db...
Name Daniels iPhone 4S
Daniels iPhone 4S
5 minutes ago
Devices
Map
History
Settings

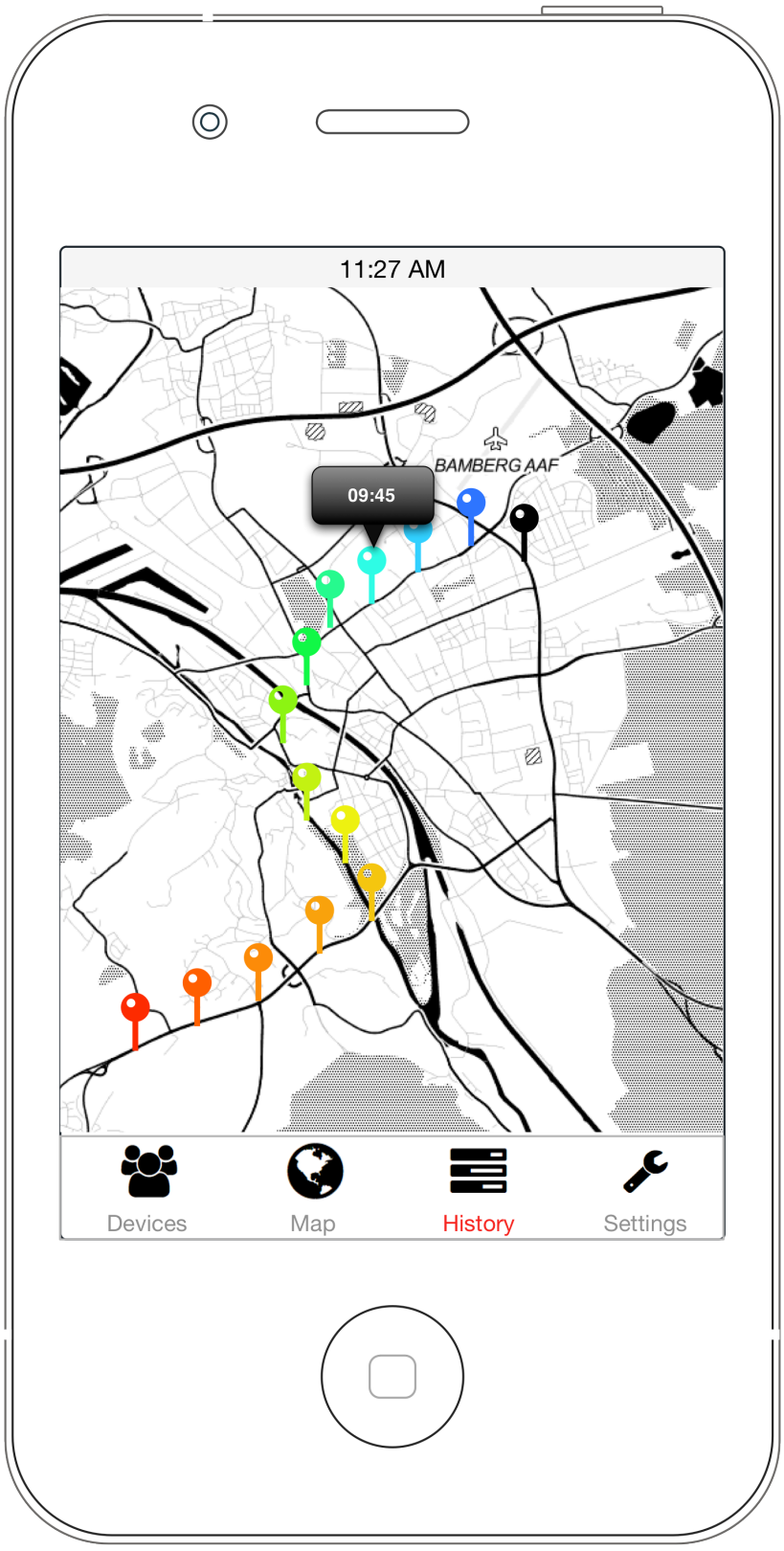
Devices Edit



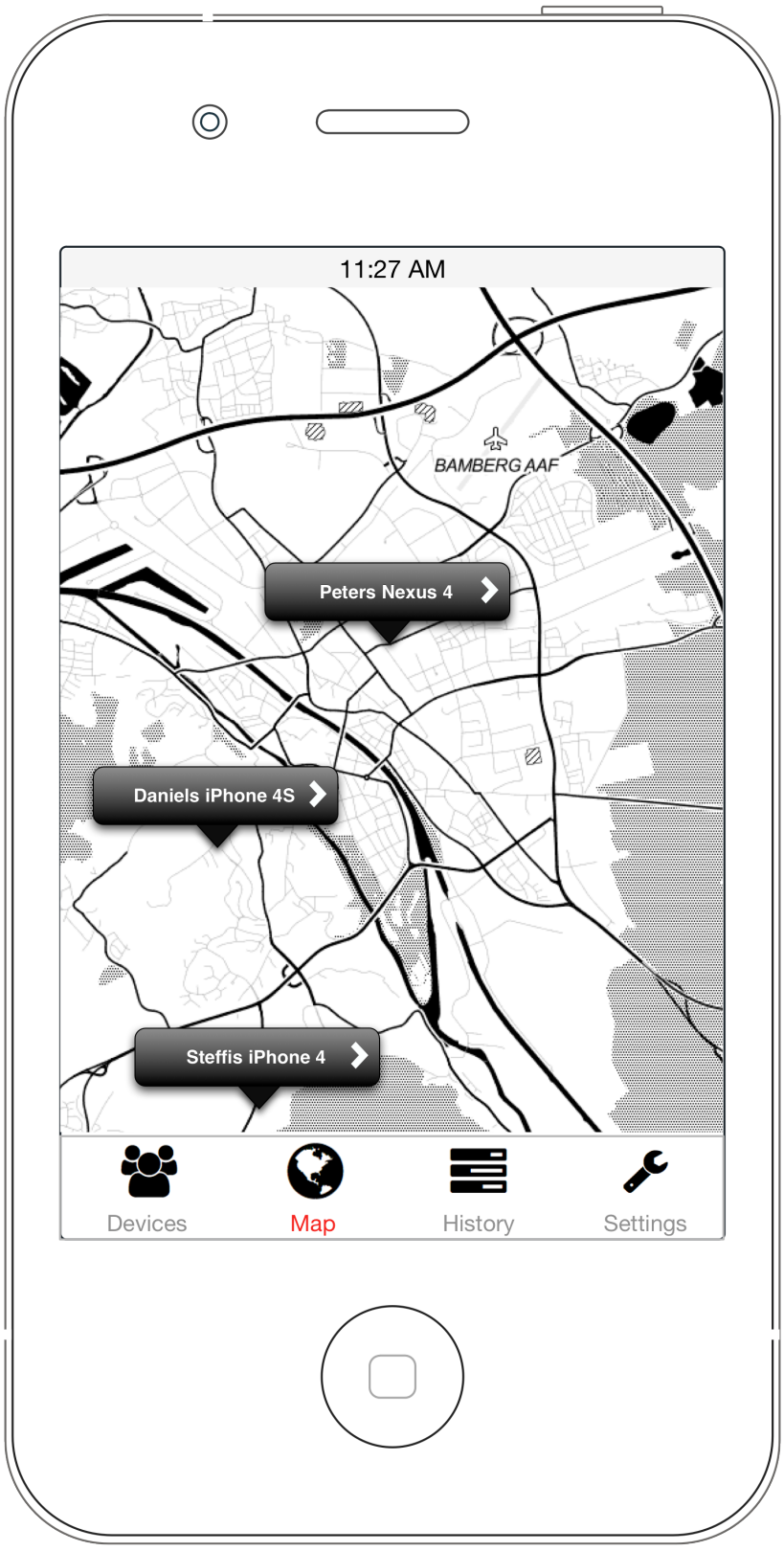
Devices Detail



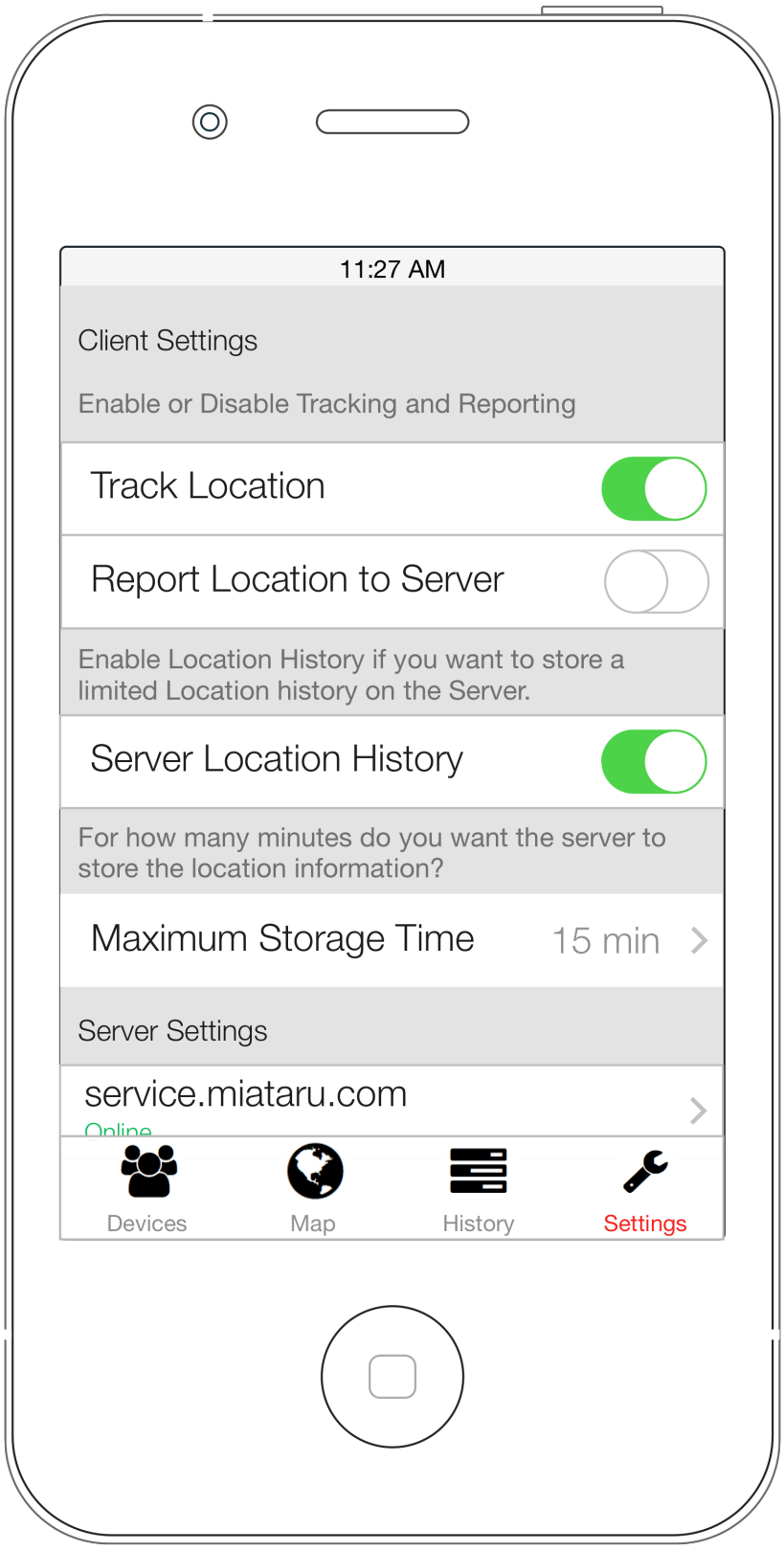
History



Map


11:27 AM
BAMBERG AAF
Peters Nexus 4
Daniels iPhone 4S
Steffis iPhone 4
Devices
Map
History
Settings

Settings


11:27 AM
Client Settings
Enable or Disable Tracking and Reporting
Track Location
Report Location to Server
Enable Location History if you want to store a limited Location history on the Server.
Server Location History
For how many minutes do you want the server to store the location information?
Maximum Storage Time
15 min
Server Settings
service.miataru.com
Online
Devices
Map
History
Settings