Smartphone App

Created: December 29th, 2017

Content:

* Requirements
* Screen descriptions
* APIs
* Data structure
* Test environment

Update 11.01.2018:

* Fixes in section Encrypting data
* Update Test Environment: add payload and new qr code
* Requirement 3a: configurable location update interval removed
* User screen: removed drop-down for location update interval

Update: 18.01.2018

* Update link in Login Screen to sign-up for new data vault

Update: 03.02.2018

* Support creating of multiple items at once
* Data-vault API and frontend is available at single URL

Requirements

**Nonfunctional requirements**

1. Create / support in setting up a Google Play Store and Apple App Store account for OwnYourData
2. Upload Android version of the OwnYourData Smartphone App to Google Play Store and make it available world-wide
3. Upload iOS version of the OwnYourData Smartphone App to Apple App Store, get it accepted and make it available world-wide
4. The following additional functionality is planned for future versions of the app (in case of different options when designing/implementing the software keep those aspects in mind):
   * Authentication and authorization of access to functionality and data through a hardware token
   * Collect additional data from the smartphone
     + Any other sensor data (e.g., accelerometer, gyroscope, temperature)
     + Communication data (incoming and outgoing calls, texts, messages)
     + Other data stores (Google Fit, Apple Health, Samsung Health)

**Functional requirements**

1. Multi-language support and English and German versions available
2. The following screens shall be available – for details see section “Scree Description”
   1. Login screen (including QR scan mode)
   2. Module list (including in-app browser to display web apps)
   3. User screen
3. The app shall read location information provided by the smartphone and write the data to the data vault
   1. A sensible time interval for reading location information shall be chosen and described for documentation purposes
   2. If the approval process in the Apple App Store requires the user to directly access any collected data, provide a function to email the user the collected data in the User Screen
4. The app shall write collected location information to the data vault
   1. Location information shall be cached on the smartphone and written to the data vault when connected to a WLAN; if possible, a manual trigger for uploading the location data regardless of the available internet connection shall be available on the user screen
   2. The user screen shall allow to configure the repo to be used for storing the location information in the data vault; default value: eu.ownyourdata.location
   3. If the repo has a public\_key defined (APIs > Public Key for Repo) the data shall be stored encrypted (see APIs > Write Data)
   4. The format of the location data is defined in the section “Data Structure”

Screen Description

**General**

Please understand the screen designs as proposals and I’m happy to discuss suggestions. The UI design follows the Google Material Design guidelines (<https://material.io)>. Graphics were created using PowerPoint and source file is available as screens\_171229.pptx

**App Icon**

Use the icon below as app icon – source is available as Adobe Illustrator file in OYD\_Logo\_final.ai



**Login screen**

English version:



German version:



Behavior of Login screen

1. Help link below logo displays help text (shown in middle screen)
2. [Scan QR Code] button accesses camera and reads qr information
   1. if qr code is scanned and contains valid information (JSON with PIA\_URL, APP\_KEY and APP\_SECRET keys) the app automatically logs in and shows module list
   2. if qr code is scanned and contains valid information but credentials don’t work: show scanned information in login form fields and error message
   3. if qr code contains invalid / incomplete information show read information in login form fields where possible
   4. if possible show [Cancel] / [Abbrechen] in scan screen to get back to login screen
3. [Enter login data] shows login form; “address of data vault” has default value <https://data-vault.eu>  
   note: for testing purposes use the address <https://mobile.data-vault.eu>
4. [Login] button performs login request – see section APIs > Login
   1. if request is successful (response = status code 200) show module list
   2. if data vault is not reachable (no response or response time out) display:
      * + Error: can’t access data vault
        + Fehler: Datentresor nicht erreichbar
   3. If response != 200 display:
      * Error: invalid username and/or password
      * Fehler: ungültiger Benutzername   
        und/oder Passwort
5. [Sign up for new data vault] opens the address   
   <https://data-vault.eu/en/new>   
   [neuen Datentresor anlegen] opens the address   
   <https://data-vault.eu/de/new>   
   both links are opened in the default web browser of   
   the smartphone
6. [OwnYourData.eu] link opens   
   <https://www.ownyourdata.eu> in the default web   
   browser of the smartphone
7. The login screen shall only support portrait   
   orientation and on displaying the onscreen keyboard   
   the input fields and login button shall not be covered   
   by the keyboard

**Module list**



Behavior of the Module list

1. The module list renders a list of all items returned from API > List of Modules and displays the image, title and description.
2. The module list shall only support portrait orientation.
3. Title is a single line and truncated with an ellipsis (…) if longer than the available space; description is 3 lines and truncated with an ellipsis (…) if longer than the available space.
4. Tapping on an item shall open the address in the in-app browser using the following URL built of components provided by APIs > List of Modules:

* MODULE\_URL: provided as field “url” in response
* PIA\_URL: provided when logged in to the smartphone app
* APP\_KEY: provided as field “uid” in response
* APP\_SECRET: provided as field “secret” rin response

Concatenate those pieces in the following way:

*[MODULE\_URL]*?PIA\_URL=*[PIA\_URL]*&APP\_KEY=*[APP\_KEY]*&APP\_SECRET=*[APP\_SECRET]*

Example: <https://kontoentwicklung.oydapp.eu/?PIA_URL=https%3A%2F%2Fdemo.datentresor.org&APP_KEY=eu.ownyourdata.bank&APP_SECRET=fZTpuPtu6XEAVlJCcI5y>

1. In case an empty list is returned from APIs > List of Modules the following text shall be rendered:

* no modules available  
  open your data vault and connect to data sources
* keine Module verfügbar  
  öffnen deinen Datentresor und erschließe Datenquellen

„open your data vault“ / “öffne deinen Datentresor” shall open the data vault URL provided on login in the default web browser of the smartphone.

1. Opening an URL in the in-app browser shall display the module title in the header and tapping on the title shall bring back the module list. If possible the screen with the in-app browser shall support portrait and landscape view as depicted below.  
     
   

**User screen**



Behavior of the User Screen

1. The user screen shall render available actions and options as depicted above. The username in the header shall be used from the response from APIs > Login. The user name shall be truncated with an ellipsis (…) if it is wider than the available space
2. The user screen shall only support portrait orientation.
3. Information Section:
   1. Display number of location records currently stored
   2. Display date and time of last successful upload
4. Actions
   1. [Upload location data now]: upload all location records regardless of WLAN connection; show error message in case data vault cannot be contacted; clear internal location cache; update information in information section
   2. [read location]: create a new location record with currently available location information; show error message in case the OS does not provide location information; refresh number of available location records in information section
   3. [clear cache]: delete all location data currently stored on the smartphone; refresh number of available location records in information section
   4. [Open Web Data Vault]: open the data vault URL provided on login in the default web browser of the smartphone
   5. [Log out]: display login screen
5. Options
   1. Textfield [Repo for location data]
6. Footer
   1. Display version string
   2. Display copyright notice and link to <https://www.ownyourdata.eu>

APIs

**Login: POST /oauth/token**

Summary

request token to access resources associated to a user

Body Parameters

* email – string, email address of the user
* password – string, password of the user
* grant\_type – string, “password”

Response (status code, description, schema)

* 200, OK,   
  {"access\_token":"*[user token for subsequent requests]*",  
   "token\_type":"bearer",  
   "expires\_in":7200,  
   "created\_at":*[unix timestamp] ]*",  
   "username":"*[user name]*"}
* 401, Unauthorized,  
  {"error":"*[title]*",  
   "error\_description":"*[description]*"}

Example

$ curl -d grant\_type=password \

-d email=user\_a@ownyourdata.eu \

-d password=user\_a \

-X POST https://mobile.data-vault.eu/oauth/token

Result:  
{"access\_token":"197a8130efe8933c8a99683c02f6bb46aa53146d18528350dfd03f6df6fd3e12","token\_type":"bearer","expires\_in":7200,"created\_at":1514391263,"username":"User A"}

**List of Modules: GET /api/modules/index**

Summary

get list of mobile applications to be displayed

Header Parameters

* Content-Type – string, “application/json”
* Authorization – string, “Bearer “ + token from login

Response (status code, description, schema)

* 200, OK,  
  {"id":*[unique id for each item]*,  
   "name":"*[title]*",  
   "description":"*[short description]*",  
   "url":"*[url of web site to be opened within app]*",  
   "uid":"*[key to access data vault]*",  
   "secret":"*[secret to access data vault]*",  
   "picture":"*[base64 encoded image]*"}
* 401, Unauthorized,  
  {"error":"*[title]*",  
   "error\_description":"*[description]*"}

Example[[1]](#footnote-1)

$ export TOKEN=`curl -s -d grant\_type=password \  
-d email=user\_a@ownyourdata.eu -d password=user\_a \  
-X POST https://mobile.data-vault.eu/oauth/token | \  
jq -r '.access\_token'`

$ curl -H "Content-Type: application/json" \  
-H "Authorization: Bearer $TOKEN" \  
-X GET https://mobile.data-vault.eu/api/modules/index

Result:

[{"id":1,"name":"Annotate","url":"https://location-annotate.oydapp.eu","description":"add notes to places you visit","uid":"8ac6377e5697123aa18f8bca860f3bd2d8db6ec54002cc3becbdce9b1ac97943","secret":"95f1ea7336476d2d6c9b352c75d7bad1d2de0d3c456b973d3e9ffae4e00a1694","picture":"..."}]

**Public Key for Repo: GET /api/repos/{:identifier}/pub\_key**

Summary

get the public key for a given repo (to be used when writing data into a repo)

Header Parameters

* Content-Type – string, “application/json”
* Authorization – string, “Bearer “ + token from login

Response (status code, description, schema)

* 200, OK,  
  {"id":*[unique id for repo]*,  
   "identifier":"*[fully qualified name for repo]*",  
   "pub\_key":"*[hexadecimal 64-character string]*"}
* 401, Unauthorized,  
  {"error":"*[title]*",  
   "error\_description":"*[description]*"}

Example

$ export TOKEN=`curl -s -d grant\_type=password \  
-d email=user\_a@ownyourdata.eu -d password=user\_a \  
-X POST https://mobile.data-vault.eu/oauth/token | \  
jq -r '.access\_token'`

$ curl -H "Content-Type: application/json" \  
-H "Authorization: Bearer $TOKEN" \  
-X GET https://mobile.data-vault.eu/api/repos/oyd.location/pub\_key

Result:

{"id":1,"identifier":"oyd.location","public\_key":"c2c45a740316583c8b1af013e24c66711590f1f1921cbc8ee2c8be9188b13731"}

**Write Data: POST /api/repos/{:identifier}/items**

Summary

Write a record into a given repo

Header Parameters

* Content-Type – string, “application/json”
* Authorization – string, “Bearer “ + token from login

Body Parameters

* JSON encoded record or array of JSON encoded records

Response (status code, description, schema)

* 200, OK,  
  {"id":*[unique id for new item]*}
* 400, Bad Request  
  {"processed":*n*,  
   "responses":[{"error":"*[description]*"},{"id": *n*},...]}
* 401, Unauthorized,  
  {"error":"*[title]*",  
   "error\_description":"*[description]*"}
* 403, Forbidden,  
  {"error":"*[title]*",  
   "error\_description":"*[description]*"}

Example

$ export TOKEN=`curl -s -d grant\_type=password \  
-d email=user\_a@ownyourdata.eu -d password=user\_a \  
-X POST https://mobile.data-vault.eu/oauth/token | \  
jq -r '.access\_token'`

$ curl -H "Content-Type: application/json" \  
-H "Authorization: Bearer $TOKEN" \  
-d "{\"value\":\"2322818113e73144fab77ce62916ec8860568ea5e9fbe3867c862056bd4c1d8e10349cbb82c52e82e0b55764805a3e923bd784820c5922ba0f079f10eeae4ef0b5155561d57986185708919d5318257d811beb9510697e307bd5d9fe27326f4e82af87f860eaa77a7eaf12079048735afad76e1396101a4576c9aa8d2778f5a8b842117d\",\"nonce\":\"59b142f13c9520cd203effa2a96dbe589defb3c1e021876e\",\"version\":\"0.4\"}" \  
-X POST https://mobile.data-vault.eu/api/repos/oyd.location/items

Result:

{"id":1}

Example for storing multiple items

$ export TOKEN=`curl -s -d grant\_type=password \  
-d email=user\_a@ownyourdata.eu -d password=user\_a \  
-X POST https://mobile.data-vault.eu/oauth/token | \  
jq -r '.access\_token'`

$ curl -H "Content-Type: application/json" \  
-H "Authorization: Bearer $TOKEN" \  
-d "[{\"value\":\"2322818113e73144fab77ce62916ec8860568ea5e9fbe3867c862056bd4c1d8e10349cbb82c52e82e0b55764805a3e923bd784820c5922ba0f079f10eeae4ef0b5155561d57986185708919d5318257d811beb9510697e307bd5d9fe27326f4e82af87f860eaa77a7eaf12079048735afad76e1396101a4576c9aa8d2778f5a8b842117d\",\"nonce\":\"59b142f13c9520cd203effa2a96dbe589defb3c1e021876e\",\"version\":\"0.4\"},{\"value\":\"2322818113e73144fab77ce62916ec8860568ea5e9fbe3867c862056bd4c1d8e10349cbb82c52e82e0b55764805a3e923bd784820c5922ba0f079f10eeae4ef0b5155561d57986185708919d5318257d811beb9510697e307bd5d9fe27326f4e82af87f860eaa77a7eaf12079048735afad76e1396101a4576c9aa8d2778f5a8b842117d\",\"nonce\":\"59b142f13c9520cd203effa2a96dbe589defb3c1e021876e\",\"version\":\"0.4\"}]" \  
-X POST https://mobile.data-vault.eu/api/repos/oyd.location/items

Result:

{"processed":2,"responses":[{"id":2,"status":200},{"id":3,"status":200}]}

Comment:

Make sure to check if all items were written successfully. The response status code is 200 if all items were written successfully and 400 otherwise. Delete only those location items in the local cache that were actually written into the data vault and keep the others.

Encrypting data

The Sodium crypto library (<https://libsodium.org>) is used to encrypt and decrypt data for OwnYourData.

The following R script illustrates the encryption process:

# public\_key is return value from GET /api/repos/eu.ownyourdata.location/pub\_key

public\_key <- '3c140482018abdc7e2a8d70f1f797e83b46c77218a9bc56ce1ae35acd2b53920'

# public\_key\_raw are 32 pairs of hex digits representing the key for encryption

public\_key\_raw <- as.raw(strtoi(sapply(seq(1, nchar(public\_key), by=2),

function(x) substr(public\_key, x, x+1)), 16L))

# authentication\_key is only used because of the required signature key when

# using the function auth\_encrypt

authentication\_key <- sodium::sha256(charToRaw('auth'))

# nonce is non-secret unique data to randomize the cipher

nonce <- sodium::random(24)

# JSON message to be encrypted

message <- '{"latitude":40.75846,"longitude":73.92248,"elevation":160.34, "timestamp":"2017-12-28T16:49:22Z","datum":"EPSG:4326"}'

# libsodium requires a sequence of raw bytes

message\_raw <- charToRaw(message)

# encrypting and signing the message

cipher <- sodium::auth\_encrypt(message\_raw,

authentication\_key,

public\_key\_raw,

nonce)

# converting value and nonce into a character string for JSON encoding

value <- paste0(as.hexmode(as.integer(cipher)), collapse = '')

nonce <- paste0(as.hexmode(as.integer(nonce)), collapse = '')

# creating JSON record to be stored in data vault

record <- jsonlite::toJSON(list(value = value,

nonce = nonce,

version = "0.4"),

auto\_unbox = TRUE)

Result (content of variable record):

{"value":"2322818113e73144fab77ce62916ec8860568ea5e9fbe3867c862056bd4c1d8e10349cbb82c52e82e0b55764805a3e923bd784820c5922ba0f079f10eeae4ef0b5155561d57986185708919d5318257d811beb9510697e307bd5d9fe27326f4e82af87f860eaa77a7eaf12079048735afad76e1396101a4576c9aa8d2778f5a8b842117d","nonce":"59b142f13c9520cd203effa2a96dbe589defb3c1e021876e","version":"0.4"}

Use the following link to open the OwnYourData Location application and display the stored data: <https://location.oydapp.eu/?PIA_URL=https%3A%2F%2Fmobile.data-vault.eu&APP_KEY=8ac6377e5697123aa18f8bca860f3bd2d8db6ec54002cc3becbdce9b1ac97943&APP_SECRET=95f1ea7336476d2d6c9b352c75d7bad1d2de0d3c456b973d3e9ffae4e00a1694>

The password for decrypting the stored data is: user\_a

(this is now the same as the password for login)

Data Structure

Proposed JSON format for location data:

{

    "timestamp":"2017-12-28T00:49:22Z",  
    "speed":0,  
    "accuracy":20,  
    "heading":0,  
    "longitude":47.0700467,  
    "altitude":1.34,  
    "latitude":15.4288967,  
    "datum":"EPSG:4326"

}

JSON format for encrypted data:

{

"value":"*[encrypted record]*",

"nonce":"*[unique data for randomizing cipher]*",

"version":"0.4"

}

Test Environment

To implement and test the app the current state of the OwnYourData data vault is deployed at the following URL: <https://mobile.data-vault.eu>

The following data is available:

* 2 users (use for login to <https://mobile.data-vault.eu>)
  + Name: User A  
    Email: [user\_a@ownyourdata.eu](mailto:user_a@ownyourdata.eu)  
    Password: user\_a
  + Name: Very long User Name with many words  
    Email: [user\_b@ownyourdata.eu](mailto:user_b@ownyourdata.eu)  
    Password: Q/wdfbv67

Note: New users (with working email address) can be created at this URL:  
<https://mobile.data-vault.eu/en/new>

* Desktop View “Location” installed for user\_a
* Mobile View “Location” installed for user\_a
* Repo oyd.location with Public Key available and permissions set to allow writing
* QR code for loging in as [user\_a@ownyourdata.eu](mailto:user_a@ownyourdata.eu)

payload:

{

"PIA\_URL":"https://mobile.data-vault.eu",

"email":"user\_a@ownyourdata.eu",

"password":"user\_a"

}  
  


1. requires jq (command line JSON processor: <https://stedolan.github.io/jq/>) [↑](#footnote-ref-1)