BONTRAX

Hiking/Treking App Technical Specification

Version 0.03

July 15, 2016

# Change History

|  |  |  |
| --- | --- | --- |
| Version | Changes | Author |
| 0.01 | Define all API / user story used in mobile app and web app | Hoang.Phan |
| 0.02 | Add pois, routes, track, me api | Hoang.Phan |
| 0.03 | Encode all in params: no more page, perPage, and search. | Hoang.Phan |

# Overall Architecture

The diagram below shows a high level architecture. Each of the components below will be described in a separate section. App will be architect with Restful Backand Webservice and 3 client: Mobile App (iOS), Admin WebApp (for backoffice work) and Website (for user who can manage track, route and read news)

Figure 1 - Hiking/Treking Mobile App Architecture

Figure 2 - Admin WebApp Architecture

Figure 3 - Website Architecture

# Webservice

Backend REST services support only 2 types of formats: JSON (maybe XML in the future). Clients that prefer using JSON format for communication must send the HTTP accept header and specify JSON as the first accepted MIME format.

Additionally, the backend services respond with different HTTP status codes depending on the outcome of processing requests. The following table lists the error HTTP status codes that could be returned.

|  |  |
| --- | --- |
| **HTTP Status Code** | **When** |
| 400 (Bad request) | If the request has an invalid syntax. |
| 401 (Unauthorized) | Authentication fail. |
| 403 (Forbiden) | If the request requires the user to have a specific role that she doesn’t have |
| 404 (Not found) | If the request is requesting a resource/entity that cannot be found |
| 405 (Mehod not allowed) | If the request’s HTTP method is not supported. |
| 406 (Not Acceptable) | If the requested response format based on the accept header is missing or not supported by this operation. For example, the operation only supports JSON but the client is requesting XML or vice versa |
| 408 (Request Timeout) | The service requires communication with an external service but that communication has timed out |
| 411 (Length Required) | If the length is not specified in the HTTP request. Should be returned automatically by the HTTP server |
| 414 (URI too long) | If the request URI is too long. Should be returned automatically by the HTTP server |
| 419 (Token Expires) | If the authentication token provided in the request has expired |
| 422 (Semantic error) | If the request cannot be processed due to semantic errors. For example, validation errors while creating or updating a resource/entity |
| 428 (Missing token) | If the request requires the user to be authenticated but the request doesn’t contain the authentication header either is the session cookie or in the HTTP header |
| 429 (Blacklist token) | Token should be used only once, so blacklist is to prevent |

**Endpoint For Mobileapp / Website**

Testing endpoin: https://sandbox.niteco.se/bontrax/api/v1/

Staging endpoint: https://api.dev.bontrax.com/v1/

Default protocol should be https for client to connect, but webservice support both protocol: http, https.

**Note for online use:**

Webservice will support both offline and online use. For online use, no need to store data in local storage and all data will come from online use. For example, if you want to get all pois of 1 user, for example, user\_id=5 and for zoom\_level=1 (all Sweden), call

... api/v1/pois/5?params=123dsfasl45gfdg4566lkgjfg

Long1

Long2

Lat1

Lat2

Where params = encodebase64(jsonencode(array("area"=>{"north":lat1,"south":lat2,"east":long1,"west":long2})))

If need for zoom\_level = 2, call another request

... api/v1/pois/5?param=123dsfasl45gfdg4566lkgjfg

Where param = encodebase64(jsonencode(array("area"=>{"north":lat1’,"south":lat2’,"east":long1’,"west":long2’})))

**Note for offline use:**

For offline use, first should sync at splash screen / or when user push a button “sync now”. Client should call all pois of 1 user that’s not caches the last time

... api/v1/pois/5?param=3424234

Where params=encodebase64(jsonencode(array("sync\_at"))) ->will return all pois of 1 users that create after the last cached time.

Then Client will call database to get pois that is corresponding to zoom\_level.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **URI** | **Method** | **Parameters** | **Required Roles** | **JSON Response** |
| /register | POST | **name, email, password**  {  "name": "Dung Vuong"  "email":"dung.vuong@niteco.se",  "password":"123123123"  } | VISITOR | **All info of new user, default status=1 (activated).**  {  "code": 200,  "message": "OK",  "result": {  "id": 14,  "name": "Hoang P",  "avatarFile": null,  "email": "hoang78@gmail.com",  "tel": null,  "role\_id": 3,  "active": 0,  "created\_at": "07/15/2016",  "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOjE0LCJpc3MiOiJodHRwczpcL1wvYXBpLmRldi5ib250cmF4LmNvbVwvdjFcL3JlZ2lzdGVyIiwiaWF0IjoxNDY4NTUzMjY5LCJleHAiOjE0Njg1OTY0NjksIm5iZiI6MTQ2ODU1MzI2OSwianRpIjoiMTQ0ZTA4MTFkNWVhNThhYTZiMGEyOGFhYmE4ZjAzMDUifQ.gs14grsvvDTJONSsSS2RB19xtKRYrHqc9ACDuY4cPDA"  }  } |
| /register/{hash\_link} | GET | **hash\_link: for verified email**  /register/123bcd456edf789qwe3df4dsg | VISITOR | **OK or Timeout (hash link will be timeout in 1 week)**  {"code":"200",  "message":"ok",  "result":{  "email":"dung.vuong@niteco.se",  "toast":"Your email is verified",  }  } |
| /login | POST | **email, password**  {  "email":"dung.vuong@niteco.se",  "password":"123123123"  } | VISITOR | **Return JWT token and expired time, default is 5 minutes. If for mobile, please send to refresh token, so that mobile app only login 1 times.**  {  "code": 200,  "message": "OK",  "result": {  "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1..."  }  } |
| /loginext | POST | **email, access\_token, provider**  {"email":"dung.vuong@niteco.se",  "provider":"facebook"  "access\_token":"abcabcabc"} | VISITOR | **JWT token, expected time (like login) now move to JWT (see JWT specification)** |
| /refresh | GET | None | VISITOR | **JWT token, same as /login** |
| /forgot | POST | **email**  {"email":"dung.vuong@niteco.se"} | VISITOR | Send mail include link to reset password.  Link forgot will be like that  https://bontrax.se/forgot/123fadsf434dsgfasd |
| /users/me | GET | None | USER | **All info of 1 users, based on token (Token also contain some short info like user\_id, avatar, roleid, ...)**  {  "code": 200,  "message": "OK",  "result": {  "id": 4,  "name": "Hoang.Phan",  "avatarFile": {  "id": 8,  "origin": .../123\_52140.png",  "crop": .../123\_52140\_90x90.png",  "small": .../123\_52140\_300x300.png",  "medium":.../123\_52140\_600x600.png",  "large": .../123\_52140\_900x900.png"  },  "email": "hoang.phan@niteco.se",  "tel": "0912106550",  "role\_id": 1,  "active": 1,  "created\_at": "06/27/2016"  }  } |
| /news?params= | GET | **Params(page=,perPage=,search=,status\_id=1,is\_homepage=1)**  Params will be encode for filter/search pager. The algorithm is encodeBase64(json\_string)  Ex (see more at param specification) params=**eyJwYWdlIjoxLCJwZXJQYWdlIjo1fQ==**, so **eyJwYWdlIjoxLCJwZXJQYWdlIjo1fQ==** is encodeBase64(‘{"page":1,"perPage":5}’) | VISITOR | **List all news that has been published. Note: params=enbase64(json\_endcode(array("status"=>1)))**  {"code":"200",  "message":"ok",  "result":[  {  "id":"1",  "title":"abc song",  "desc":"abc song created by .. xxx",  "image": "https://images.bontrax.se/1.jpg",  "publish\_date": "25-09-2016T12:12:12+7",  "type":"news",  "url":"https://www.bontrax.se/a.html",  "created\_by":{  "user\_id":"1",  "user\_name":"Hoang.Phan",  "role\_id":"1"  }  }  ]  } |
| /news/{news\_id} | GET | **news\_id** | VISITOR | **All content of 1 news, including content, some meta data: geolocation, byline, ....**  **Note: api check status\_id first.** |
| /{user\_id}/pois?params= | GET | **params (search=,perPage=,page=, sync\_at)**  **user\_id=0 if system pois and #0 if user.**  search="Niteco"  params=4234dfgsdfg564654fdhtghgfh5  Params will be encode like new.  **For online use,** params=en64{"area":{"north":lat1,"south":lat2,"east":long1,"west":long2}}  **For offline use,**  params=en64{"sync\_at":"25-09-2016T12:12:12+2"} | USER | **List all POIs of registered user. Should from access\_token->user\_id->api.**  **+If use for offline,** params = en64{"sync\_at":?},  **+ if for online,** params=en64["area"=>{"north":?,"south":?,"east":?,"west":?}]  {"code":"200",  "message":"ok",  "result":[  {  "id":"100",  "title":"My pee place",  "note":"abc song created by .. xxx",  "icon":"http://static/icon/pois/tent.jpg",  "coordinates":"112.3,1123.4,23.2",  "created\_by":{  "user\_id":"1",  "user\_name":"Hoang.Phan",  "role\_id":"1"  }  }  ]  } |
| /{user\_id}/pois/{poi\_id} | GET | **user\_id, poi\_id**  user\_id="1"  poi\_id="5"  In webservice, we will check wether poi\_id is belong to user\_id and user\_id is belong to current access token. | USER | Detail of 1 POIs  {"code":"200",  "message":"ok",  "result":  {  "id":"100",  "title":"My pee place",  "note":"abc song created by .. xxx",  "icon":"http://static/icon/pois/tent.jpg",  "coordinates":"112.3,1123.4,23.2",  "created\_by":{  "user\_id":"1",  "user\_name":"Hoang.Phan",  "role\_id":"1"  }  }  } |
| /{user\_id}/pois//{poi\_id} | POST | All field of 1 poi (new) | USER | Same as GET |
| {user\_id}//pois/ | PUT | Same as POST (edit) | USER | Same as POST |
| /{user\_id}/routes?params= | GET | **params, user\_id**  user\_id=5  params=4234dfgsdfg564654fdhtghgfh5  Params will be encode like “news”.  **For online use,** params={}  **For offline use,**  params={"sync\_at":"25-09-2016T12:12:12+2"} | USER | **List all Routes of registered user. Should from access\_token->user\_id->api**  {"code":"200",  "message":"ok",  "result":[  {  "id":"100",  "title":"Home to Work",  "note":"abc song created by .. xxx",  "gpx":"http://static.bontrax.se/work.gpx",  "created\_by":{  "user\_id":"1",  "user\_name":"Hoang.Phan",  "role\_id":"1"  }  }  ]  } |
| /{user\_id}/routes/{route\_id} | GET | **user\_id,poi\_id**  user\_id="1"  poi\_id="5"  In webservice, we will check wether poi\_id is belong to user\_id and user\_id is belong to current access token. | USER | **Detail of 1 Routes**  {"code":"200",  "message":"ok",  "result":  {  "id":"100",  "title":"Home to Work",  "note":"abc song created by .. xxx",  "gpx":"http://static.bontrax.se/work.gpx",  "created\_by":{  "user\_id":"1",  "user\_name":"Hoang.Phan",  "role\_id":"1"  }  }  } |
| /{user\_id}/routes/{poi\_id} | POST | All field of 1 poi (new) | USER | Same as GET |
| /{user\_id}/routes/ | PUT | Same as POST (edit) | USER | Same as POST |
| /upload | POST (multi-part) | Binary file | USER  ADMIN | ID of media file uploaded, and 5 path if images  {"code":"200",  "message":"ok",  "result":[  {  "id":"100",  "filename":"Home to Work",  "url":[  "original":"http://images.bontrax.se/1",  "filename":"Home to Work",  ]  }  ]  } |
| /files/{file\_id} | GET | All information of file.  If file is images-> include all sizes of images.  If file is gpx-> only original file. | USER  ADMIN |  |
| {user\_id}/tracks?params= | GET | Params: encode all filter|searh|pager | USER |  |
| {user\_id}/tracks/{track\_id} | GET | Same as routes. | USER |  |
| {user\_id}/tracks/{track\_id} | POST | Same as routes. | USERS |  |
| {user\_id}/tracks/ | PUT | Same as route | USERS |  |
| {object\_id}/comments/ | GET | All comments of 1 objects: news, profile, route, track, poi,.... |  |  |
| {object\_id}/comments/{comment\_id} | POST |  |  | Edit 1 comment. |
| {object\_id}/comments/ | PUT |  |  | New comment |

**Endpoint For Admin Webapp**

Testing endpoind: https://sandbox.niteco.se/bontrax/api/v1/

Staging endpoint: https://api.dev.bontrax.com/v1/

Default protocol should be https for client to connect, but webservice support both protocol: http, https.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **URI** | **Method** | **Parameters** | **Required Roles** | **JSON Response** |
| /users?params= | GET | None | ADMIN | All users filter by param  {"code":"200",  "message":"ok",  "result":[  {  "id":"1",  "name":"Hoang Phan",  "email":"hoang@gmail.com",  "title": "Worker",  "role\_id":"1",  "avatar\_id":"23",  "status\_id":"1",  }  ]  } |
| /users/{user\_id} | GET | None | ADMIN | All user information.  {"code":"200",  "message":"ok",  "result":[  {  "id":"1",  "name":"Hoang Phan",  "email":"hoang@gmail.com",  "tel":"1234567890",  "address":"abc str 123, HaNoi city",  "title": "Worker",  "role\_id":"1",  "avatar\_id":"23",  "status\_id":"1"  }  ]  } |
| /users/{user\_id} | POST | All field of user. If select avatar, should update directly (auto save) -> this will be describe in /upload | ADMIN | Same as GET user detail. |
| /users/ | PUT | Same as POST, bt now create new user | ADMIN | Same as GET user detail. |
| /articles?param= | GET | None | ADMIN | List all articles.  {"code":"200",  "message":"ok",  "result":[  {  "id":"1",  "title":"abc song",  "desc":"abc song created by .. xxx",  "image": "https://images.bontrax.se/1.jpg",  "publish\_date": "25-09-2016",  "type":"normal",  "url":"https://www.bontrax.se/a.html",  }  ]  } |
| /articles/{id} | GET | None |  |  |
| /pois?params= | GET | params | ADMIN | For manage all pois (user\_id=0 is system\_pois, #0 is user\_pois) |
| /pois/{poi\_id} | GET | None | ADMIN | For get specific poi. |
| /pois/{poi\_id} | PUT | All info of poi | ADMIN | For new poi |
| /pois/ | POST | All info of poi | ADMIN | For edit 1 poi |
| /routes?params= | GET |  | ADMIN |  |
| /routes/{route\_id} | GET |  | ADMIN |  |
| /routes/{route\_id} | PUT |  | ADMIN |  |
| /routes/ | POST |  | ADMIN |  |
| /tracks?params= | GET |  | ADMIN |  |
| /tracks/{track\_id} | GET |  | ADMIN |  |
| /tracks/{track\_id} | PUT |  | ADMIN |  |
| /tracks/ | POST |  | ADMIN |  |
| /{object\_id}/tags/ | GET |  | ADMIN | All tags of 1 objects |
| /{object\_id}/tags/{tag\_id} | POST |  | ADMIN | Edit tag. |
| /{object\_id}/tags/ | PUT |  | ADMIN | New tag. |
| /maps?params | GET |  | ADMIN | All map zone (offline map) |
| /maps/{map\_id} | GET |  | ADMIN | Info of 1 map zone (including kmz list | zip package) |
| /maps/{map\_id} | POST |  | ADMIN |  |
| /maps | PUT |  | ADMIN | Add new map zone. |

# Hiking/Treking Mobile App

This is the mobile application that treking/hiking guy use. Specification of user stories and GUI are described as cards in the project’s Jiraboard (and describe here)

The following are general requirements for the app iOS (Android will be specified later)

1. The App should support iOS 8,9.
2. The App should support screen sizes with resolutions between 640 X 1136 (5s) and 1080 X 1920 (6s).
3. Crash reporting will be done with CrashLytic.
4. SQLite database will be used for storing information.
5. Upon logging in for the first time, the app retrieves all user data from the server and stores it locally in SQLite database. The app also stores the authentication token in a SharedPreferences so it doesn’t need to prompt the user to login again unless the server returned an authentication or authorization error as a response for one of the requests.
6. In case of conflicts between local updates and server updates, the newest change overrides the older one. All records in the app will have a "Last Modified" timestamp to indicate when the update occurred.
7. All names used in the application should follow Naming Conventions.
8. Mobile app should have a test suite that tests the app automatically either in isolation or with the service layer.
9. Follow the quality of code checklist.

The authentication token is generated by JWT library (https://jwt.io/). For iOS parser, we can use this lib (https://github.com/auth0/JWTDecode.swift).

**Describe this token as below (note: we also tracking token base on AppID, now AppID for iOS = 1)**

|  |  |
| --- | --- |
| **Encode** | **Decode** |
| eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJuYW1lIjoiSG9hbmcuUGhhbiIsInJvbGVfaWQiOjEsImF2YXRhciI6Imh0dHBzOlwvXC9hc3NldHMuZGV2LmJvbnRyYXguY29tXC9hdmF0YXJzXC80XC8yMDE2XzA3XzA2XC8xMjNfNTIxNDBfMzAweDMwMC5wbmciLCJjcmVhdGVkX2F0IjoiMjAxNi0wNi0yN1QxMjowNTo0MiswNzowMCIsImNyZWF0ZWRfYXRfY21zIjoiMDZcLzI3XC8yMDE2Iiwic3ViIjo0LCJpc3MiOiJodHRwczpcL1wvYXBpLmRldi5ib250cmF4LmNvbVwvdjFcL2xvZ2luIiwiaWF0IjoxNDY4NTU0MzIxLCJleHAiOjE0Njg1OTc1MjEsIm5iZiI6MTQ2ODU1NDMyMSwianRpIjoiM2NiMjQyNjQxODIwYTYzY2I5YTJjNThiNzRmNzYzMzcifQ.Xi4owIVmHeBimlK8U4T8y4YHDp\_4tu\_yD1mZ5AWN5dM | Header (ALGORITHM & TOKEN TYPE)  {  "alg": "HS256",  "typ": "JWT",  } |
| Payload(DATA)  {  "name": "Hoang.Phan",  "role\_id": 1,  "avatar": .../123\_52140\_300x300.png",  "created\_at": "2016-06-27T12:05:42+07:00",  "created\_at\_cms": "06/27/2016",  "sub": 4,  "iss": "https://api.dev.bontrax.com/v1/login",  "iat": 1468554321,  "exp": 1468597521,  "nbf": 1468554321,  "jti": "3cb242641820a63cb9a2c58b74f76337"  } |
| Signature (VERIFY)  HMACSHA2HMACSHA256(  base64UrlEncode(header) + "." +  base64UrlEncode(payload),    KnCJwVrx0HY3UxmI62GdJpOknobqkWdf  ) |

**Describe about param encoding**

|  |  |
| --- | --- |
| **Encode (encodeBase64)** | **Decode (decodeBase64)** |
| eyJwYWdlIjoxLCJwZXJQYWdlIjo1fQ== | {"page":1,"perPage":5} |
| eyJwYWdlIjoxLCJwZXJQYWdlIjo1LCJzZWFyY2giOiJob2FpIn0= | {"page":1,"perPage":5,"search":"hoai"} |
| eyJwYWdlIjoxLCJwZXJQYWdlIjo1LCAicm9sZV9pZCI6M30= | {"page":1,"perPage":5, "role\_id":3} |
| eyJwYWdlIjoxLCJwZXJQYWdlIjo1LCAicm9sZV9pZCI6MywgInN0YXR1c19pZCI6MX0= | {"page":1,"perPage":5, "role\_id":3, "status\_id":1} |
| eyJwYWdlIjoxLCJwZXJQYWdlIjo1LCAicm9sZV9pZCI6MywgInN0YXR1c19pZCI6MSwgInN5bmNfYXQiOiIyMDE1LTA5LTA5IDEyOjEyOjM0KzcifQ== | {"page":1,"perPage":5, "role\_id":3, "status\_id":1, "sync\_at":"2015-09-09 12:12:34+7"} |

# Admin Webapp

This is the webapp for admin/webmaster/editor/partner\_admin can use (for management like: user, map, route, notification, ...). Specification of user stories and GUI are described as cards in the project’s Jiraboard (and describe here).

**Note:** token same as mobile app (but do not allow for refresh)

The following are general requirements:

1. Webapp will use Angular2 as client business logic.
2. Webapp should support responsive design (use Angular and Bootstrap)
3. Follow coding convention.

# Website

This is the webapp for hikor / trekor ise (for manage personal information: profile, map, friend, route, ...). Specification of user stories and GUI are described as cards in the project’s Jiraboard (and describe here).

**Note:** token same as mobile app (but do not allow for refresh)

The following are general requirements:

1. Webapp will use Angular2Universal as server rendering (for fastboot and also no binding)
2. Webapp should support responsive design (use Angular and Bootstrap)
3. Follow coding convention.

# Detail user story

Ref: admin\* here mean: sysadmin/editor/partner\_admin.

**Admin webapp**

|  |  |
| --- | --- |
| Selection_002 | **US01: As an admin\*, I can login to webapp.** |
| Input: username and correct password. |
| Output:  - Admin can login to dashboard and see all the menu/dashboard.  - The session time: will be 5 minutes. |
| Testcase:  - Correct username and password -> ok with session lifetime of 5 minutes.  - Wrong username and password->warning (after 5 fail? Wait? Capcha) |
| QA Ref: |
|  | **US02: As an admin\*, I can reset password** |
| Input: Click on forgot password, enter email. Open mail and click link provided in mail to reset password. |
| Output:  - Admin can click on reset password (like normal user)->display page to enter: email.  - Mail should be in inbox and click link provided to reset pass.  - Display page to enter new password |
| Testcase:  - Email correct.  - Not exist email / email format not correct.  - Link is timeout (should resend again or click forgot again) |
| QA Ref: |
| Selection_002  Selection_001 | **US03: As an admin, I can manages users in system (2 UI)** |
| Input: Admin can use menu “Users/Add” to CRUD all users in the system. Add/Edit support upload avatar (Change status/role will be decribe in another stories, but this also should support later) |
| Output:  - Admin can CRUD all users registers in our system.  - Support upload avatar.  (- May support cardview if have time) |
| Testcase:  - Can CRUD all users.  - Can filter/search/paging in listing page.  - Should validate some info needed (email, password, fullname) |
| QA Ref: |
| Selection_003 | **US04: As an admin, I can see all my menus for management** |
| Input: After login, admin/partner can see menus in system |
| Output: Menu at the left side |
| Test:  - Display menus in all browsers.  - Support responsive |
| QA Ref: |
| Selection_002  Selection_003 | **US05: As an admin, I can manage roles** |
| Input: same as manage users. |
| Output: Same as manage users (Later, maybe should support permission schema in our system) |
| Test case:  - Same as manage users.  (Support permission later) |
| QA Ref: |
|  | **US06: As an admin, I can manage article in web** |
| Input: same as manage users |
| Output: same as manage user. Later, article should support meta-data and can be pushed as notification, can add some resource like POI, EVENT, ROUTES,... |
| Test case:  - Same as manage users  (- Another will be describe in another US) |
| QA Ref: |
|  | **US12: As an admin, I can add images/video/youtube link to article** |
|  |
|  |
|  |
|  |

**Website**

|  |  |
| --- | --- |
|  | **US07: As an user, I can register an account** |
|  |
|  |
|  |
|  |
| Selection_003 | **US08: As an user, I can login to website** |
| Input: User click on login, and can enter user/password or click on facebook/google+ to login. |
| Output:  - Login ok -> link become logout and menu breadcrum appear.  - User click menu: can manage profile, track, route, map, ... |
| Testcase:  - Correct username and password -> ok with session lifetime of 5 minutes.  - Click on facebook button.  - Click on google+ button.  - Wrong username and password->warning (after 5 fail? Wait? Capcha) |
| QA Ref: |
|  | **US09: As an user, I can forgot/reset password** |
|  |
|  |
|  |
|  |
|  | **US10: As an user, I can see my menu/profile** |
|  |
|  |
|  |
|  |
|  | **US11: As an user, I can change my profile setting (including password)** |
|  |
|  |
|  |
|  |
|  | US13: |
|  |
|  |
|  |
|  |

**Mobile app**

|  |  |
| --- | --- |
| Selection_004 | **PS01: As an user, I can login to mobile.** |
| Input: For first time user, login screen will display after wizard/splash screen. User can use email/pass or click on Google+/Facebook to login. If they do not have acc, they click on button Register. |
| Output:  - Login ok -> user go to landing screen (map with no layer)  - User can see menu at bottom right and can click to manage profile, track, route, map, ... |
| Testcase:  - Correct username and password.  - Click on facebook button.  - Click on google+ button.  - Wrong username and password->warning (after 5 fail? Wait? Capcha) |
| QA Ref: |
| Selection_001 | **PS02: As an user, I can register an account.** |
| Input: User should enter name, email, password and re-password. Also should agree with Privacy Policy. |
| Output:  - If everything ok -> toaster “Welcome” and goto Home Screen.  - If password is not match -> toaster  - If email is existed -> toaster and back to login.  - If not agree with Privacy Policy -> Toaster |
| Testcase: Same as output |
| QA Ref: |
| Selection_002 | **PS03: As an user, I can use forgot password** |
| Input: User should enter email |
| Output:  - If email is found, toaster “Please check email and click link.”  - If not found, toaster “Email is not registered” |
| Testcase:  - Email correct.  - Not exist email / email format not correct  - Link is timeout (should resend again or click forgot again) |
| QA Ref: |
| Selection_003 | **PS04: As an user, I can see my favorite map when login successfully.** |
| Input: User will be redirected to HomeMap Screen after login ok. If for second use, user should be bringed to this screen. |
| Output:  - If first time on this screen, default is terran map near your location without any icons.  - If second time, map type, layer, ... restore from preferences. |
| Testcase: Following with the output and some actions  - Click on icon -> more info.  - Click on my location -> center to screen.  - Click on layer -> open dialog to choose layer  - Click on menu -> show menu  - Click on search -> show search dialog |
| QA Ref: |