

Project Deliverable

Project Number:	Project Acronym:	Project Title:
287901	BUTLER	uBiquitous, secUre inTernet-of-things with Location and contExt-awaReness

Instrument:	Thematic Priority
Integrated Project	Internet of things

Title
BUTLER SmartMobile platform and enabling technologies

Contractual Delivery Date:	Actual Delivery Date:
September, 30 th , 2013	October, 31 st , 2013

Start date of project:	Duration:
October, 1 st 2011	36 months

Organization name of lead contractor for this deliverable:	Document version:
INNO	V1.1

Dissemination level (Project co-funded by the European Commission within the Seventh Framework Programme)		
PU	Public	X
PP	Restricted to other programme participants (including the Commission)	
RE	Restricted to a group defined by the consortium (including the Commission)	
CO	Confidential, only for members of the consortium (including the Commission)	

Authors (organizations) :

Fabrice Clari (inno), Bertrand Copigneaux (inno),

Reviewers (organizations) :

Cristina Frà (TIL), Miguel-Angel Monjas (ERC), Massimo Valla (TIL)

Abstract :

BUTLER SmartMobile is a framework that allows application developers to make their apps available to end-users and end-users to browse, install and use available applications. Said applications can be either hybrid or web thanks to the use of HTML5.

The BUTLER SmartMobile framework provides a framework to develop, deploy and deliver applications through various components: a mean to deploy apps that can be used either as native, a responsive HTML5 UI framework, a client-side BUTLER library and a backend service acting as the BUTLER Applications Repository.

Through BUTLER SmartMobile applications, end-users can take advantage of BUTLER functionalities.

This deliverable covers components provided by the BUTLER SmartMobile. This document might be read along with D4.1 as BUTLER SmartMobile elements that have server-side components have been described in that document.

Keywords :

Mobile, GUI, End User Application, Android, iOS, Human Machine Interface, API, App, PhoneGap, Bootstrap

Disclaimer

THIS DOCUMENT IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE. Any liability, including liability for infringement of any proprietary rights, relating to use of information in this document is disclaimed. No license, express or implied, by estoppels or otherwise, to any intellectual property rights are granted herein. The members of the project BUTLER do not accept any liability for actions or omissions of BUTLER members or third parties and disclaims any obligation to enforce the use of this document. This document is subject to change without notice.

Revision History

The following table describes the main changes done in the document since it was created.

Revision	Date	Description	Author (Organisation)
V0.1	August 2013	Creation	F. Clari (inno)
V0.5	September 30, 2013	Major Updates	F. Clari (inno)
V0.8	October, 2013	Major updates	F. Clari (inno)
V0.9	October, 2013	Review	Cristina Frà (TIL)
V1.0	October, 2013	Final Review and updates	B. Copigneaux (inno)
V1.1	November, 2013	Update after project Review	F. Clari, B. Copigneaux (inno), Miguel-Angel Monjas (Ericsson Spain), Cristina Frà (TIL), Massimo Vala (TIL)

Table of Content

1. ACRONYMS AND DEFINITIONS	6
2. EXECUTIVE SUMMARY	7
3. HIGH-LEVEL ARCHITECTURE	8
3.1. BUTLER Entities	8
3.2. The BUTLER SmartMobile Framework.....	8
4. BUTLER SMARTMOBILE FRAMEWORK DETAILED ARCHITECTURE	10
4.1. Introduction	10
4.2. Hybrid BUTLER Applications	10
4.3. Mobile Web BUTLER Applications.....	11
4.4. BUTLER.js	12
4.4.1. Authentication and User Profile	12
4.4.1. Application-specific functions.....	16
4.4.2. BUTLER.js settings.....	22
4.5. BUTLER User Interface	22
4.6. How to use UI elements	25
4.7. Single Page Application (SPA).....	25
4.8. Integration of native apps within the BUTLER SmartMobile framework.....	26
5. SERVER-SIDE COMPONENTS	27
5.1. BUTLER applications repository.....	27
6. DEVELOPING BUTLER SMARTMOBILE APPS	28
6.1. Development framework overview.....	28
6.2. Developer mode	28
6.3. Developing a BUTLER application	29
6.3.1. Creating a simple applicaton	29
6.3.2. Starting the application on a device	30
6.4. Deployment of BUTLER apps	32
7. CONCLUSIONS AND FURTHER DEVELOPMENTS	34
8. REFERENCES	35

List of Figures

Figure 2 - BUTLER SmartMobile, hybrid version.....	11
Figure 3 - BUTLER SmartMobile, mobile web version.....	11
Figure 4 - UI: typography	23
Figure 5 - UI: navigation	23
Figure 6 – UI: Alert boxes	23
Figure 7 - UI: buttons.....	24
Figure 8 - UI: miscellaneous.....	24
Figure 9 - Example of BUTLER native application.....	25
Figure 10 - Add a native application	26
Figure 11 - Skeleton application	30
Figure 12 - Starting the skeleton application	31
Figure 14 – BUTLER Application life-cycle	32

1. ACRONYMS AND DEFINITIONS

Acronyms	Definitions
NFC	Near Field Communication
SPA	Single Page Applications
AJAX	Asynchronous JavaScript and XML
URL	Uniform resource locator
HTTP	Hypertext Transfer Protocol
DOM	Document Object Model
CSS	Cascading Stylesheets
SPI	Single Page interfaces
IP	Internet Protocol
USB	Universal Serial Bus
GPG	GNU Privacy Guard

2. EXECUTIVE SUMMARY

End-users are intended to access context-aware BUTLER applications by means of mobile terminal (smartphones, tablets and the like). BUTLER SmartMobile is the framework that allows, on one hand, **BUTLER application developers to make their apps available to end-users** and, on the other, **end-users to browse, install and use available BUTLER applications**. As mobile terminals are the main target of the BUTLER SmartMobile framework, we use the widely used term ‘app’ to name BUTLER applications.

The BUTLER SmartMobile framework targets mobile platforms and thus, the following terminal and technologies are supported:

- HTML5 mobile web browsers;
- Android devices;
- iOS devices (at the time of writing, this version is not yet available, Android being the main OS targeted);

In order to access BUTLER applications, end-users download a single BUTLER SmartMobile ‘app’ (the BUTLER SmartMobile framework) on their terminal. This generic app will give access to the actual BUTLER applications. The BUTLER SmartMobile framework provides:

- A **mechanism to develop, deploy and deliver BUTLER end-user applications**: e.g. a set of tools have been developed and made available to BUTLER application developers for them to easily build applications on top of the BUTLER SmartMobile framework. It provides a skeleton application that can be used as a template, an app launcher that is useful when external applications need to be started, and a repository and related scripts to securely deploy applications.
- The **means to deploy apps that can be used either as native** (more precisely hybrid through PhoneGap) **or as mobile web apps**. As previously explained, application can be available from the BUTLER SmartMobile ‘app’ downloaded to the end-user terminal or through a web browser.
- A **responsive HTML5 UI framework**: such a UI framework allows any application built on top of it to be made available on various types of mobile devices, e.g. desktop, phones, tablets, TVs, etc.
- A **client-side BUTLER library (BUTLER.js)**. This JavaScript library provides a set of core features required to develop BUTLER applications on the end-user terminals, such as the authentication. While those features are not provided by BUTLER SmartMobile, the BUTLER.js library wraps external libraries in order to provide a set of consistent API to developers. This library provides as well methods to access low-level components on the mobile device in a common way (e.g. same method calls on any targeted mobile platform).
- A **backend service acting as the BUTLER Applications Repository**: this service is the central place where developed applications are deployed. Indeed, it allows developers to deploy and publish their apps. Then, this service makes applications available either to BUTLER SmartMobile clients or web browsers. Finally, it provides a set of API used by the developers’ tools to deploy apps and by BUTLER SmartMobile clients to get the lists of available applications (e.g. applications that have been developed and published).

However, it is worth noting that the BUTLER SmartMobile framework does not provide server-side apps features (e.g. no hosting facilities for server-side scripting).

3. HIGH-LEVEL ARCHITECTURE

3.1. BUTLER Entities

This document will describe component that have been developed in the task 4.2, related to SmartMobile. In that context, two entities exist and need to be clearly defined:

- **BUTLER SmartMobile client**, often simply mentioned as SmartMobile, is the client side of BUTLER that is meant to be used on mobile devices. It is basically an application that runs on the client side;
- **BUTLER SmartMobile framework** is the full set of components developed in order to build and distribute BUTLER application: it includes the SmartMobile client, a server-side architecture acting as an application store and a set of tools that are used by developers to deploy their BUTLER applications.

3.2. The BUTLER SmartMobile Framework

The overall architecture of the BUTLER SmartMobile framework was introduced previously in D2.4 [1] and is also described from another point of view in D4.1 [3] as it provides server-side components. Basically, initial statements remain identical: BUTLER SmartMobile is a framework that makes it possible the development and the deployment of BUTLER mobile applications as either hybrid apps (e.g. mobile but embedded with PhoneGap [4]) or mobile web apps (e.g. as HTML5 apps).

The figure below shows a high-level view of the overall BUTLER SmartMobile framework, covering both the client side (BUTLER SmartMobile app, web browser) and the BUTLER Applications Repository.

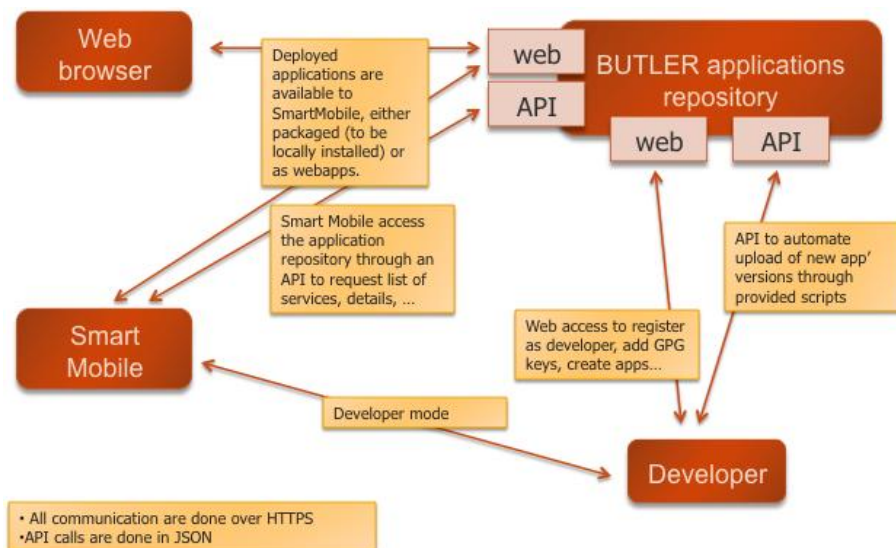


Figure 1 - BUTLER SmartMobile framework overview

In order to clarify the functionality of every component within the BUTLER SmartMobile framework, we describe below the way a BUTLER application is deployed and user by end-user:

1. The developer builds his/her application using the BUTLER SmartMobile development tools

2. When the application is finished and ready to be deployed, the developer uses a common script to send the application to the BUTLER Applications Repository . This script signs the code and sends it to the BUTLER Applications Repository
3. Once on the BUTLER Applications Repository, the server-side component checks the signature, and if the file is validated, makes the application available to deployment to BUTLER SmartMobile clients
4. Then, clients (e.g. BUTLER SmartMobile client or web browsers) can access the application: while the BUTLER SmartMobile client will use the BUTLER Applications Repository API to get the list of files to be downloaded, the web browser will request files as done in regular web applications.

4. BUTLER SMARTMOBILE FRAMEWORK DETAILED ARCHITECTURE

4.1. Introduction

As already defined in D2.4 [1], the BUTLER SmartMobile framework allows developers to make applications available both as native applications and as web applications. In the context of the BUTLER SmartMobile framework, native applications will be in fact hybrid; indeed they will embed web resources (such as HTML pages, images...) and will be developed as HTML5 apps. Then, they will be packaged as hybrid application through the PhoneGap platform. On the other side, each application will also be available as a regular web application that end-users will be able to access from HTML5 mobile web devices. Here, it is worth noting the webapp version will provide fewer features than hybrid application, as they will get access to less low-level resources.

Both types of applications will take advantages of BUTLER components, such as user interface elements (see section 4.4.), or BUTLER SmartServers features made available through the BUTLER.js library (such as for example authentication or user profile management).

4.2. Hybrid BUTLER Applications

The figure bellows illustrates how hybrid BUTLER Applications take advantage of the BUTLER SmartMobile framework. In this version, the BUTLER SmartMobile runtime deployed on the end-user mobile terminal gives access to various low-level components and moreover gives developers the opportunity to extend the PhoneGap framework to add new functionality, such as for example access to an NFC reader (that is available on various Android-powered phones). In such a context, with regard to the BUTLER SmartMobile framework, a developer willing to write a new component accessing low-level features will have to first develop the low-level code, and then wrap this code within JavaScript code that will be integrated within the BUTLER.js library.

Moreover, this version gives also access to operating systems features such a notification engine.

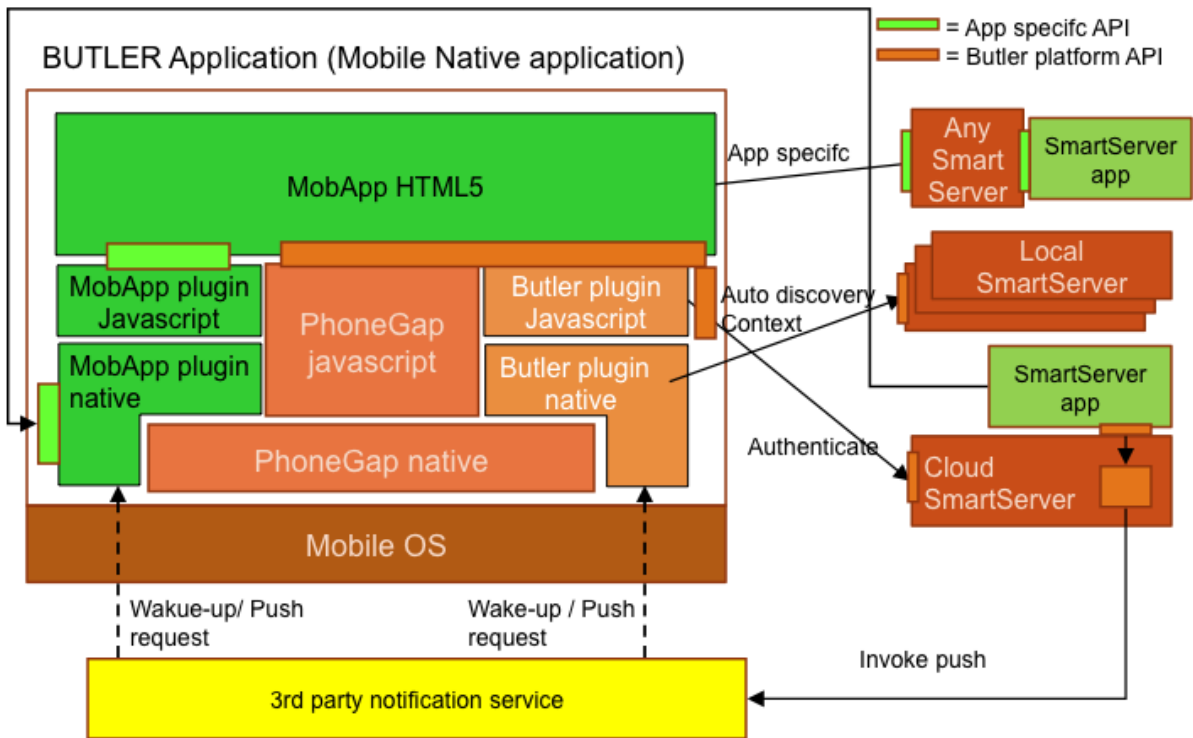


Figure 2 - BUTLER SmartMobile, hybrid version

4.3. Mobile Web BUTLER Applications

As described earlier, the BUTLER SmartMobile framework gives also developers the opportunity to package their applications as HTML5 web applications, as highlighted in the figure below. The overall architecture is similar to the hybrid version except that such applications do not have access to low-level features, unless provided through HTML5, nor operating systems features such as notifications.

On Figure 3, the orange box "BUTLER library Javascript" shows the BUTLER.js library whereas the MobApp library is the custom application Javascript code.

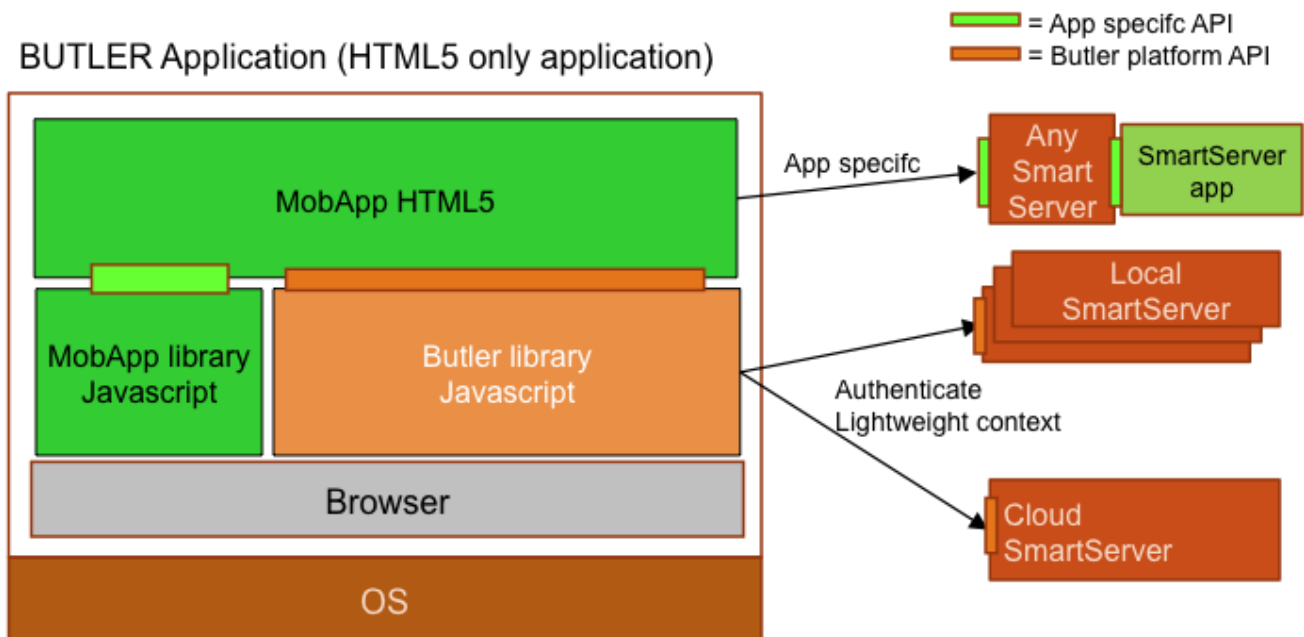


Figure 3 - BUTLER SmartMobile, mobile web version

4.4. BUTLER.js

The BUTLER.js library is a software component that gives BUTLER SmartMobile developers' common BUTLER features, such as authentication, SPA (see 4.7.) or user profile management. In that context, BUTLER features can be either available on the device itself or on remote SmartServer.

It is worth noting that BUTLER.js tries to provide a consistent API so developers can use same methods while working on the hybrid version or the mobile web app version.

4.4.1. Authentication and User Profile

This section describes the BUTLER JavaScript library that handles authentication and user profile management in the BUTLER SmartMobile framework.

The library provides methods for user registration, reading/updating user profile data, and user authentication. The library abstracts and simplifies for the application programmer interaction with the User Profile SmartServer and the BUTLER security framework implemented by the Authorization SmartServer. Programmers using this library, part of the BUTLER SmartMobile framework, will just need to use simple JavaScript methods to perform these functionalities.

The JQuery library is used for synchronous and asynchronous calling and is required for AJAX requests to SmartServers.

Available JavaScript functions in this library are:

- RegisterAccount
- LoginAccount
- GetAccount
- UpdateAccount
- DeleteAccount
- RefreshTokenThread

4.4.1.1. RegisterAccount

This method allows new user registration in the system. User is identified with unique login that each new user has to provide in registration phase. Authorization is checked according to this field. Password must be provided with client security enforcement: it's up to the client to decide about its encryption.

Optional resultFunction can be provided for asynchronous response, otherwise response is returned as exit parameter of registerAccount function.

4.4.1.1.1. Signature

```
registerAccount(login, password, resultFunction)
```

4.4.1.1.2. Parameters

PARAMETERS	VALUE	DESCRIPTION	TYPE
login	donald	Unique string for user login	string
password	password	Clear/encrypted string containing user password	string
resultFunction	See Example	Callback function for asynchronous result	function

4.4.1.1.3. Example

```
function resultCallback(result){
    if(result.error)
```

```

        alert('Error: '+result.message);
    else
        alert('Register OK, user id: '+result.id); }
registerAccount('donald','password',resultCallback);

```

4.4.1.1.4. Error codes

The following table shows the most common errors.

CODE	DESCRIPTION
400	The request is not valid
401	User already exists
500	Database error / Internal Server error

4.4.1.2. LoginAccount

This method allows user to login into the system. This is the authorization phase, where user is recognized as a previously registered user. User is identified with unique login provided during the registration step. Password must match the one already provided.

Optional resultFunction can be provided for asynchronous response. Otherwise response is returned as exit parameter of loginAccount function.

4.4.1.2.1. Signature

```
loginAccount(login, password, resultFunction)
```

4.4.1.2.2. Parameters

PARAMETERS	VALUE	DESCRIPTION	TYPE
login	donald	Unique string for user login provided in registration phase	string
password	password	Clear/encrypted string containing user password, same as registration phase	string
resultFunction	See Example	Callback function for asynchronous result	function

4.4.1.2.3. Example

```

function resultCallback(result){
    if(result.error)
        alert('Error: '+result.message);
    else
        alert('Login OK, user auth data: '+result.access_token); }
loginAccount('donald','password',resultCallback);

```

4.4.1.2.4. Error codes

The following table shows the most common errors.

CODE	DESCRIPTION
400	The request is not valid
401	Unauthorized / User not found
500	Database error / Internal Server error

4.4.1.3. GetAccount

This method allows user to retrieve profile information, providing an authorization token was obtained after logged in successfully. User profile is returned as an object compliant with OpenSocial specification. This is a protected method: authorization mechanism provides the authorization after being authenticated through login method.

Optional resultFunction can be provided for asynchronous response. Otherwise response is returned as exit parameter of getAccount function.

4.4.1.3.1. Signature

```
getAccount(token, resultFunction)
```

4.4.1.3.2. Parameters

PARAMETERS	VALUE	DESCRIPTION	TYPE
token	"45gst5768ej"	token returned from loginAccount operation	string
resultFunction	See Example	Callback function for asynchronous result	function

4.4.1.3.3. Example

```
function resultCallback(result){
    if(result.error)
        alert('Error: '+result.message);
    else
        alert('Get OK, user profile: '+result); }

getAccount('45gst5768ej',resultCallback);
```

4.4.1.3.4. Error codes

The following table shows the most common errors.

CODE	DESCRIPTION
401	User not authorized
404	User not found
500	Database error / Internal Server error

4.4.1.4. UpdateAccount

This method allows a client to update profile information, providing data object with only fields to be updated and the token obtained after logging in. Data object must contain the mandatory id field. Updated user profile is returned as structured object compliant with OpenSocial specification. This is a protected method: authorization mechanism provides the authorization after being authenticated through login method. Optional resultFunction can be provided for asynchronous response. Otherwise response is returned as exit parameter of updateAccount function.

4.4.1.4.1. Signature

```
updateAccount(personJsonObj, token, resultFunction)
```

4.4.1.4.2. Parameters

PARAMETERS	VALUE	DESCRIPTION	TYPE
personJsonObj	{ "id": "38f1d453003e", "birthday": "1980-02-	JSON string with user profile data containing information to be updated	string

	19T00:00:00+0100"}}		
token	"45gst5768ej"	token returned from loginAccount operation	string
resultFunction	See Example	Callback function for asynchronous result	function

4.4.1.4.3. Example

```
function resultCallback(result){
    if(result.error)
        alert('Error: '+result.message);
    else
        alert('Update OK, user profile updated: '+result); }
updateAccount({'id': "38f1d453003e","birthday": "1980-02-19T00:00:00+0100"}',resultCallback);
```

4.4.1.5. Error codes

The following table shows the most common errors.

CODE	DESCRIPTION
401	User not authorized
404	User not found
500	Database error / Internal Server error

4.4.1.6. DeleteAccount

This method allows the client to delete profile information, providing token obtained after logged in successfully. This is a protected method: authorization mechanism provides the authorization after being authenticated through login method.

Optional resultFunction can be provided for asynchronous response. Otherwise response is returned as exit parameter of deleteAccount function.

4.4.1.6.1. Signature

```
deleteAccount(token, resultFunction)
```

4.4.1.6.2. Parameters

PARAMETERS	VALUE	DESCRIPTION	TYPE
token	"45gst5768ej"	token returned from loginAccount operation	string
resultFunction	See Example	Callback function for asynchronous result	function

4.4.1.6.3. Example

```
function resultCallback(result){
    if(result.error)
        alert('Error: '+result.message);
    else
        alert('Delete OK'); }
registerAccount('45gst5768ej',resultCallback);
```

4.4.1.6.4. Error codes

The following table shows the most common errors.

CODE	DESCRIPTION
401	User not authorized

404	User not found
500	Database error / Internal Server error

4.4.1.7. RefreshTokenThread

This is helper method used to manage OAuth2 authentication and authorization. It's based on OAuth2 API and its purpose is to issue token requests in order to keep user logged in and avoid token expiration. Requests are sent periodically according to token duration. This function requires a valid access token structure to be present i.e. user successfully logged into platform.

4.4.1.7.1. Signature

```
refreshTokenThread();
```

4.4.1.7.2. Parameters

This method does not require any parameters.

4.4.1.7.3. Example

```
refreshTokenThread();
```

4.4.1.7.4. Error codes

No error codes are currently returned by this method.

CODE	DESCRIPTION
EMPTY	

4.4.1. Application-specific functions

4.4.1.1. butler.page.reload

When called from the hybrid framework (on PhoneGap), this method reloads the page. This method is mainly used when developing application and when used jointly with the SmartMobile developer mode as it permits the reloading of a page and its associated resources (e.g. Javascripts and stylesheet files) without having to redeploy the full application.

Signature

```
butler.page.reload();
```

Parameters

This method does not require any parameters.

Example

```
butler.page.reload();
```

4.4.1.2. butler.page.showScreen

Shows the SmartMobile splash screen.

Signature

```
butler.page.showScreen();
```


Parameters

This method does not require any parameters.

Example

```
butler.page.showScreen();
```

Error codes

No error codes are currently returned by this method.

4.4.1.3. butler.page.hideScreen

Hides the SmartMobile splash screen.

Signature

```
butler.page.hideScreen();
```

Parameters

This method does not require any parameters.

Example

```
butler.page.hideScreen ();
```

4.4.1.4. butler.page.close

If opened, close the application and shows the splash screen

Signature

```
butler.page.close();
```

Parameters

This method does not require any parameters.

Example

```
butler.page.close();
```

4.4.1.5. log

Simple log message. Mainly used for development purposes.

Signature

```
butler.log(message);
```

Parameters

PARAMETERS	VALUE	DESCRIPTION	TYPE
message	"this is a test"	Message to be displayed	string

Example

```
butler.log("this is a test");
```

4.4.1.6. butler.ajax

This helper method sends an AJAX request. It is useful when jQuery (or similar libraries) is not available.

Signature

```
butler.ajax(method, url, callback, bind, async)
```

Parameters

PARAMETERS	VALUE	DESCRIPTION	TYPE
method	GET, POST, PUT (any HTTP verb)	Name fo the HTTP method	string
url	http://example.com	Name of the endpoint (URL)	string
callback	function(){...}	Callback function called once a response (or an error) is received	Function
Bind	DOM element	Optional parameter used to set the context (e.g. it defines the value for <i>this</i>).	DOM element
async	True	Set the request to be sent asynchronously	boolean

Example

```
butler.ajax('GET', url, function(error, response){
  if( error ){
    return callback.call(bind, error);
  }
  try{
    response = JSON.parse(response);
  }
  catch(e){
    callback.call(bind, {message: 'invalid json', error: e});
  }
  finally{
    callback.call(bind, null, response);
  }
});
```

4.4.1.7. butler.geolocation.getPosition

This helper method returns the actual position of the device. It acts as a proxy with underlying libraries. For instance, when used from a web application, it will return values provided by the HTML5 GeoLocation API¹ whereas when called from a hybrid app, it will rely on values returned by PhoneGap.

Signature

```
butler.geolocation.getPosition(callback, bind)
```

Parameters

PARAMETERS	VALUE	DESCRIPTION	TYPE
callback	function() { ... }	Callback method	Function
Bind	DOM element	Optional parameter used to set the context	DOM element

¹ <http://dev.w3.org/geo/api/spec-source.html>

		(e.g. it defines the value for <i>this</i>).	
--	--	---	--

Example

```
butler.ajax('GET', url, function(error, response){
  if( error ){
    return callback.call(bind, error);
  }
  try{
    response = JSON.parse(response);
  }
  catch(e){
    callback.call(bind, {message: 'invalid json', error: e});
  }
  finally{
    callback.call(bind, null, response);
  }
});
```

4.4.1.8. butler.list.isInstalled

Checks if an application is installed within the SmartMobile client application.

The table bellows shows how an application is internally represented:

	DESCRIPTION
Hybrid application	{ "name": "Temperature", "description": "", "token": "6jBiFJFXTF", "last_update": "2013-11-12T20:19:46.000Z", "download_url": "http://repo.iot-butler.eu/apps/sq2SQ3dTF/download" }
Native application	{ "name": "BUTLER SmartApp", "description": "", "token": "DSKL3qqsf2", "last_update": null, "package_name": "com.example.smartapp ", "package_activity": "com.example.smartapp.SmartHome" }

Signature

```
butler.list.isInstalled(app)
```

Parameters

PARAMETERS	VALUE	DESCRIPTION	TYPE
App	An application	An JSON array holding an application details	JSON object

Example

```
boolean butler.list.isInstalled(app);
```

4.4.1.9. butler.list.renderAppList

Takes a list of applications and renders the corresponding HTML fragment listing applications.

Signature

```
butler.list.renderAppList(apps);
```

Parameters

PARAMETERS	VALUE	DESCRIPTION	TYPE
Apps	List of applications	List of available applications to be shown to the end-user	JSON Array of array (e.g. array of apps)

Example

```
butler.list.renderAppList(apps);
```

4.4.1.10. butler.list.saveState

Save the list of applications (installed and available) for later use (persistent storage).

Signature

```
butler.list.saveState();
```

Parameters

PARAMETERS	VALUE	DESCRIPTION	TYPE

Example

```
butler.list.saveState();
```

4.4.1.11. butler.list.getAppByToken

Returns an application definition based on the application token (unique identified).

Signature

```
app butler.list.getAppByToken(token);
```

Parameters

PARAMETERS	VALUE	DESCRIPTION	TYPE
Token	SDKQL3DS	A token, being an application unique identifier.	string

Example

```
my_app = butler.list.getAppByToken("JEKAXDQS");
```

4.4.1.12. butler.list.installApp

Installs an application.

Signature

```
butler.list.installApp(app, callback, onprogress);
```

Parameters

PARAMETERS	VALUE	DESCRIPTION	TYPE
App	An application	A JSON object describing an application	JSON array
Callback	A callback method	A callback method that is called once the installation process is over	Function
onprogress	A callback method	A callback method that is called during the installation process with the percentage of the installation done as parameter. Useful for displaying installation progress bar.	Function

Example

```
butler.list.installApp(my_app,
    function(error){
        target.classList.remove('progress');
        target.lastElementChild.style.width = '0';
        if( error ){
            alert('error installing app' + error);
        }
    },
    function(percent){
        target.lastElementChild.style.width = percent + '%';
    }
}
```

4.4.1.13. butler.list.uninstallApp

Uninstalls an application and removes it from the file system.

Signature

```
butler.list.uninstallApp(app);
```

Parameters

PARAMETERS	VALUE	DESCRIPTION	TYPE
App	An application	A JSON object describing an application	JSON array

Example

```
butler.list.uninstallApp(my_app);
```

4.4.1.14. butler.list.openApp

Opens an already installed application.

Signature

```
butler.list.openApp(app);
```

Parameters

PARAMETERS	VALUE	DESCRIPTION	TYPE
App	An application	A JSON object describing an application	JSON array

Example

```
butler.list.openApp(my_app);
```

4.4.2. BUTLER.js settings

Various elements of the SmartMobile client can be configured from the BUTLER.js library. The following list describes each configuration setting available.

4.4.2.1. Settings definition

Setting	Type	Description
devmode	Boolean	Enable or disable the development mode. Enabled when set to true.
showreload	Boolean	When sets to true, shows a reload button on each page (when used from the hybrid version).
showterminal	Boolean	When sets to true, shows a very minimal console that is useful for debugging purposes.
server	String	Set the URL of the server from where SmartMobile will download file. When not in development mode, this value must be set to '.' (dot) as files are stored locally (once downloaded from the BUTLER application repository). In development mode, it must contain the full URL of SmartMobile files, for example: <code>http://192.168.21.118/butler</code> .

4.4.2.1. Example

```
var butler = {
  devmode: true,
  showreload: true,
  showterminal: true,
  server: 'http://192.168.21.118/butler'
}
```

4.5. BUTLER User Interface

The User Interface (UI) is a very important component as it is what is presented to the end-user. BUTLER apps share a common pre-requisite: all apps should behave the same way and should present a common user interface, regardless of their version (mobile web or hybrid).

Being in an HTML5 framework, it was decided to build the BUTLER UI on top of an existing CSS framework: Bootstrap (v2.3) [5]. The main advantage of this framework is that it is fully responsive (that is, it is possible to resize your browser window and the website/app resizes with it).

While this UI framework is mainly done for BUTLER SmartMobile apps (either hybrid or webapps), there are cases where fully native applications need to be integrated within BUTLER (and started through the launcher – see 4.8.). In such cases, native apps developers need to make their application looks like BUTLER apps: colour scheme, typography, etc. In order to do so, a Photoshop PSD file with all graphical elements is made available.

The following screenshots show various UI elements from the BUTLER UI.



Figure 4 - UI: typography

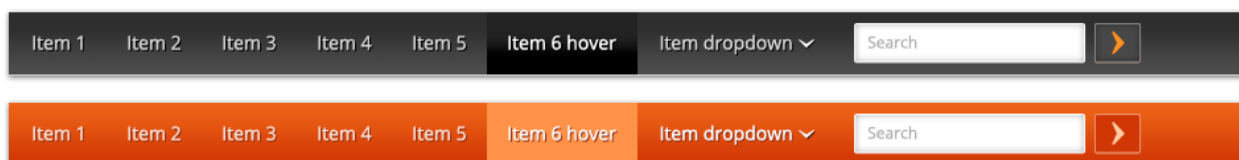


Figure 5 - UI: navigation

Alerts

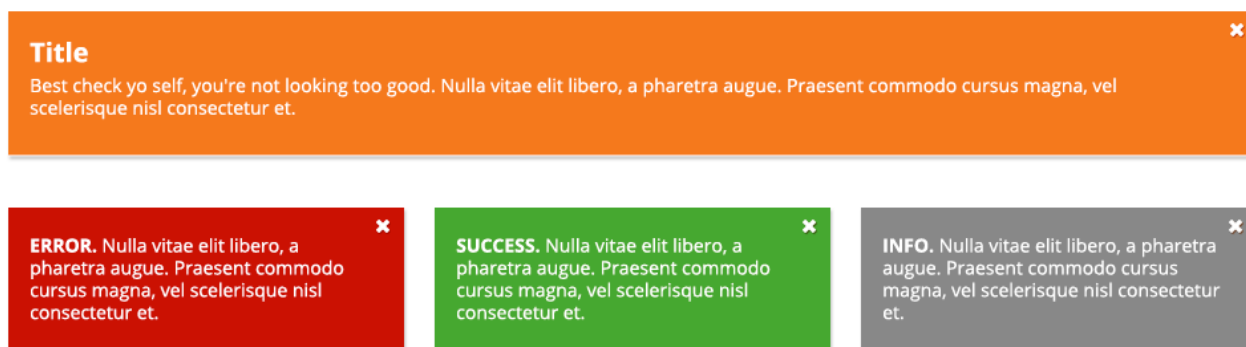


Figure 6 – UI: Alert boxes

Button	Large Button	Small Button	Disabled Button	Button with icon	Split Button
<div>Default</div>	<div>Default</div>	<div>Default</div>	<div>Default</div>	<div>➤ Default</div>	<div>Default ▼</div>
<div>Primary</div>	<div>Primary</div>	<div>Primary</div>	<div>Primary</div>	<div>👤 Primary</div>	<div>Primary ▼</div>
<div>Info</div>	<div>Info</div>	<div>Info</div>	<div>Info</div>	<div>💬 Info</div>	<div>Info ▼</div>
<div>Success</div>	<div>Success</div>	<div>Success</div>	<div>Success</div>	<div>✓ Success</div>	<div>Success ▼</div>
<div>Warning</div>	<div>Warning</div>	<div>Warning</div>	<div>Warning</div>	<div>⚠ Warning</div>	<div>Warning ▼</div>
<div>Danger</div>	<div>Danger</div>	<div>Danger</div>	<div>Danger</div>	<div>⚠ Danger</div>	<div>Danger ▼</div>
<div>Inverse</div>	<div>Inverse</div>	<div>Inverse</div>	<div>Inverse</div>	<div>↺ Inverse</div>	<div>Inverse ▼</div>

Figure 7 - UI: buttons

Breadcrumbs

Home / Category / Subcategory

Pagination

1 2 3 4 < 2 3 >

Pagers

< Previous Next >

Tabs

Item 1Item 2Item 3Item 4Item 5Item 6 hoverItem dropdown ▼

Menu Item 1Menu Item 2Menu Item 3Menu Item 4Menu Item 5

Labels

DefaultPrimaryInfoSuccessWarningDangerInverse

Progress bar

Figure 8 - UI: miscellaneous

The screenshot inserted here () shows a mobile native application (Energy/Augmented Reality application from the BUTLER Smart Energy Proof of Concept). While this application does not rely on the BUTLER SmartMobile framework, it uses UI elements that have been made available.

Furthermore, this application is called from BUTLER SmartMobile and retrieves parameters sent by BUTLER SmartMobile (e.g. user's first name, last name, email and authentication token).

4.6. How to use UI elements

The BUTLER user interface relies of the Bootstrap CSS framework², developed and maintained by Twitter.

This framework provides the foundation for BUTLER UI and has been fully customized with the BUTLER look'n feel.

In order to use BUTLER User Interface, one could check the Bootstrap documentation, in its 2.3 version (the one integrated within SmartMobile): <http://getbootstrap.com/2.3.2>.

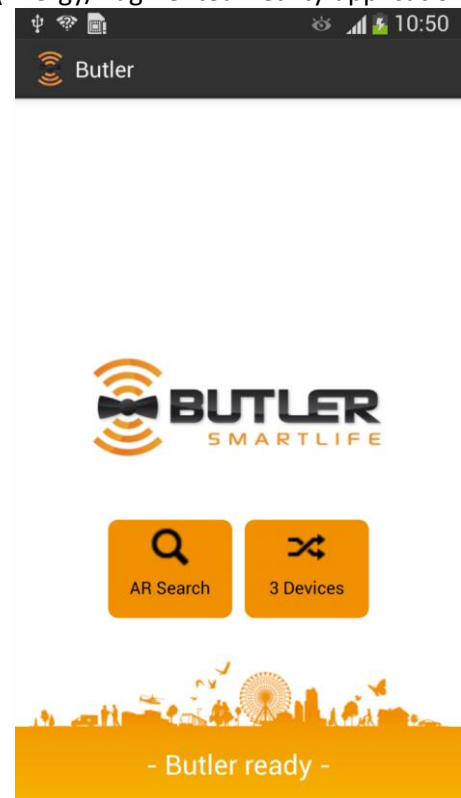


Figure 9 - Example of BUTLER native application

4.7. Single Page Application (SPA)

In order to provide a good end-user experience, various solutions have been envisaged on the front-end side. Indeed, as BUTLER applications would be made available either as web app or native apps on mobile devices, a good approach would have be to develop BUTLER applications as regular web applications, e.g. set of HTML pages and resources (images, style sheets, JavaScript file, etc.). Although this approach could have worked, it would have brought latency issues on mobile devices when used through BUTLER SmartMobile. Indeed, PhoneGap apps do not have access to various low-level components that make web apps fast, and reloading a full page each time the end-user does an action does not deliver a smooth rendering.

In that context, it was decided to use the Single Page Application (SPA) paradigm [6] to provide a good end-user experience. A single-page application, also known as single-page interface (SPI), is a web application or web site that fits on a single web page with the goal of providing a more fluid user experience akin to a desktop application.

The implemented solution is homemade, e.g. it does not rely on any existing SPA framework (AngularJS, BackboneJS to name a few) since BUTLER framework needed to remain simple with a limited footprint.

Therefore, this solution implements a single template solution and makes pages' transition smoother. The BUTLER wiki explains how to add pages to BUTLER SmartMobile applications, but basically the process is the following:

² <http://getbootstrap.com/>

1. The developer adds a page in the pages array on top of the butler.js file
2. Then s/he creates the page file in the partials folder: my_page.html
3. S/he adds a link to the page link

4.8. Integration of native apps within the BUTLER SmartMobile framework

In order to start from BUTLER SmartMobile apps that cannot be developed/ported to HTML5, an app launcher has been developed. This component allows developer to register any native application within the BUTLER Applications Repository and to start them from BUTLER SmartMobile.

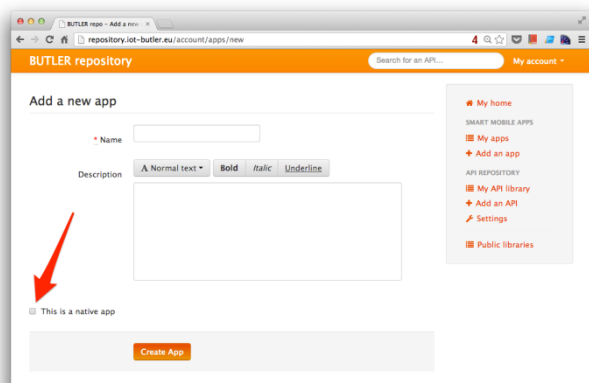


Figure 10 - Add a native application

Figure 10 shows how to add a native application to the set of available applications (see D4.1 [3] for the BUTLER Applications Repository webapps)

Furthermore, any app can be called from BUTLER SmartMobile through a Javascript call:

```
BUTLER.launch(app_package_name)
```

If the user is authenticated when the `BUTLER.launch()` call is done, his/her token will be passed as parameter to the called application.

5. Server-side components

The SmartMobile framework includes a set of server-side tools that are required to develop applications. The main element is the application repository that stores all deployed applications. It is used by SmartMobile at start-up time: when starting, a SmartMobile application (in its hybrid version) checks the list of available applications in order to show a list of applications to the user, along with the list of already installed applications.

5.1. BUTLER applications repository

This section describes how BUTLER applications are deployed on the BUTLER Applications Repository. As various processes involve server-side components, their descriptions have been moved to D4.1 [3].

6. DEVELOPING BUTLER SMARTMOBILE APPS

6.1. Development framework overview

A set of tools have been developed in order to cover all development steps:

- A script to create a BUTLER SmartMobile app skeleton, to be used as development starting point
- Scripts to package, sign and deploy applications to the BUTLER Applications Repository
- Server-side scripts (and related API) to validate new versions of apps
- Server-side tools to package applications in order to make them available to both BUTLER SmartMobile hybrid and web versions

```
~/Code/iot-butler/SmartMobile ./create_skeleton.sh my_app
~/Code/iot-butler/SmartMobile ls -l my_app
total 24
drwxr-xr-x 10 fabriceclari staff 340 1 aoù 23:38 app_root
-rwxr-xr-x 1 fabriceclari staff 2412 1 aoù 23:38 butler_package.sh
-rw-r--r-- 1 fabriceclari staff 128 1 aoù 23:38 config
drwxr-xr-x 3 fabriceclari staff 102 1 aoù 23:38 node_modules
-rwxr-xr-x 1 fabriceclari staff 2283 1 aoù 23:38 server.js
```

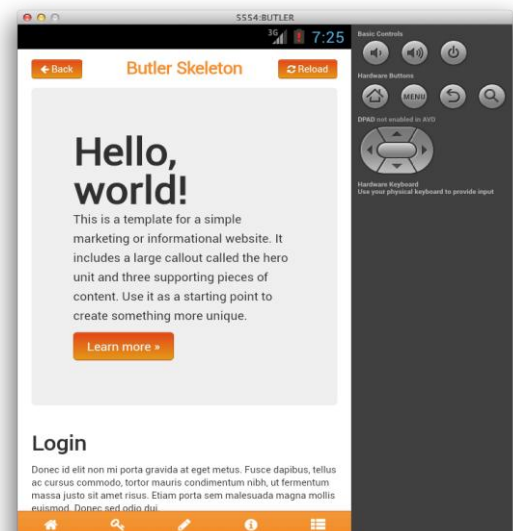
6.2. Developer mode

The developer mode allows deploying an application from the development environment and enabling the reloading of an app without restarting/redeploying the full app.

Indeed, when operating in the regular, non-developer mode, BUTLER SmartMobile client, developers would need first to deploy their application on the BUTLER Applications Repository before testing it, which would be time consuming. Another option would be for them to rebuild and re-deploy on the mobile device the full BUTLER SmartMobile app (e.g. the PhoneGap-based application) but this solution is also time consuming. In that context, a specific developer mode has been created: it allows developers to test in live their application, without any compilation.

Basically, it works as follow:

1. On his/her development environment, the developer runs a small web server, provided with the developers tools
2. On a BUTLER SmartMobile client, the developers set the developer mode on. When set, the BUTLER SmartMobile client prompts the user and requires configuration information about the development server, so that the user can introduce the IP address of the development server
3. Then the BUTLER SmartMobile client loads files from the development server instead of getting them from the BUTLER Applications Repository.



6.3. Developing a BUTLER application

6.3.1. Creating a simple applicaton

The following section will make use of the `$SMART_MOBILE_HOME` variable that will be considered as the location on the developer's computer of the SmartMobile framework.

At the time of writing, developer tools are not yet publicly available. In order to get the set of developer tools, a request needs to be sent to smart-mobile@iot-butler.eu.

Calling the `create_skeleton.sh` script can create a simple app.

```
> cd $SMART_MOBILE_HOME
> ./create_skeleton.sh my_app
> cd my_app
```

The newly created folder contains scripts to start locally the app and to deploy it. The `app_root` folder contains a simple but functional application.

```
> ls -l
total 24
drwxr-xr-x 10 fc  staff   340 26 nov 11:10 app_root
-rwxr-xr-x  1 fc  staff  2608 26 nov 11:10 butler_package.sh
-rw-r--r--  1 fc  staff   129 26 nov 11:10 config
drwxr-xr-x  3 fc  staff   102 26 nov 11:10 node_modules
-rwxr-xr-x  1 fc  staff  2327 26 nov 11:10 server.js
```

For there, the developer gets the possibility to start the application as a simple web app that he can test from his computer. In that context, he needs to start the `server.js` web server.

```
> node butler.js
```

Once started, the newly created `my_app` is available at http://localhost:8000/app_root/app.html.

From there, the developer can start by modifying pages that are in the `app_root/partials` folder. The table below shows the rendering of the application and the `home.html` page updated.

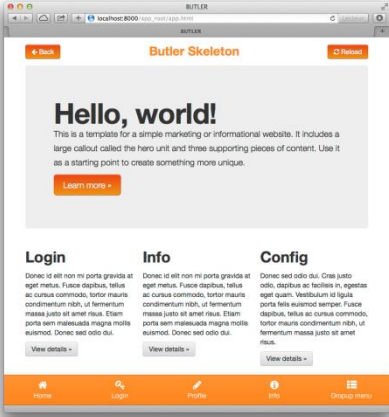
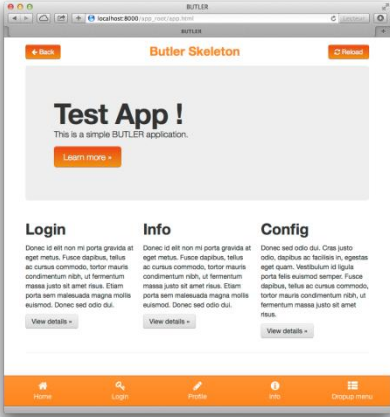
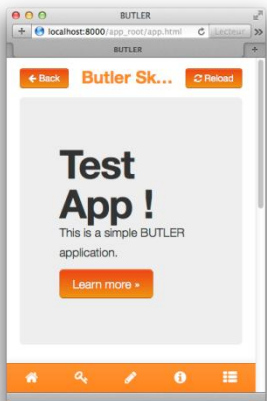
		
<p>This screenshot shows the application once created, with no modification done. It is a skeleton that can be customized to develop new applications.</p>	<p>The screenshot shows the same app with the home.html page updated. By using the local server, the developer can see how it renders without having to deploy the application on the device.</p>	<p>This screenshots the same application with the browser resized, in order to test the responsiveness of the layout (see for example legends in the footer that have been automatically hidden)</p>

Figure 11 - Skeleton application

6.3.2. Starting the application on a device

In order to start the application, the \$SMART_MOBILE_HOME/SmartMobile-Android/cordova/run needs to be called. This will either start an Android emulator or will deploy the application on a real Android device if a device is connected to an USB socket.

For the purpose of this documentation, let’s consider that no real device is plugged. The table below shows how the application renders within the SmartMobile hybrid version.


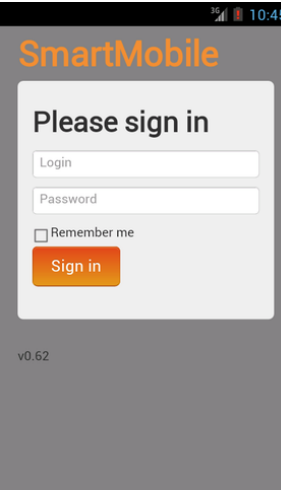
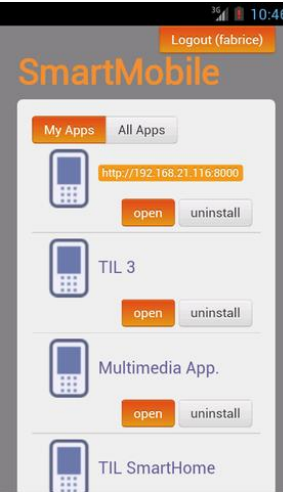
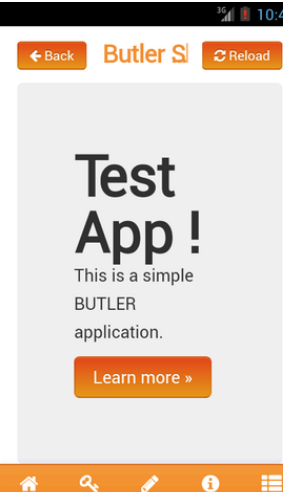
			
<p>Android applications screen. The SmartMobile application is shown on the bottom right corner.</p>	<p>When starting the application, the user is requested to log in. This login process aims at authenticating the user.</p>	<p>Once logged, a list of installed application is shown to the user. By clicking on the “All Apps” button, he can get the list of all available apps. Since we are in development mode, a special app is listed (first one). The URL is the one of the developer web server and can be updated by clicking on it.</p>	<p>When opening the first app, the newly created app is shown to the user. While in development mode, a “reload” button is displayed in the top right corner of the page, thus allowing the developer to update the application without having to restart the full framework.</p>

Figure 12 - Starting the skeleton application

6.4. Deployment of BUTLER apps

This section relies on the BUTLER Application Repository which is described in deliverable D4.1, section 12.

When a developer is ready to deploy a new version of his/her application, s/he calls (from the app folder) `./butter_package.sh`, which in turns:

1. Makes an archive of the app
2. Signs it with the developer's GPG key
3. Sends it to the BUTLER Applications Repository (through the BUTLER Applications Repository API)

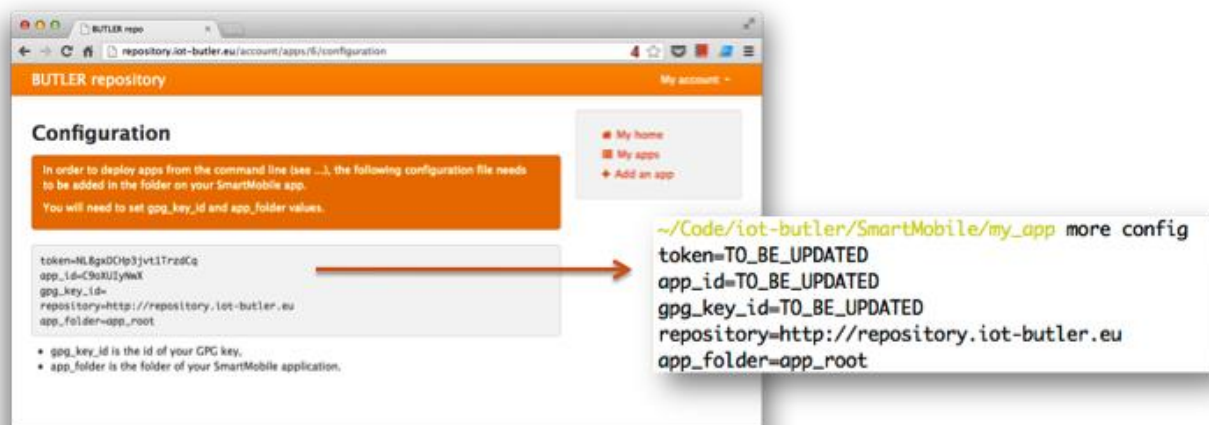


Figure 13 - BUTLER SmartMobile configuration file (development mode)

To run it on the device (real or simulator), the developer has first to update a configuration as shown on the figure above.

Once received by the BUTLER Applications Repository, the application is certified (e.g. the developer signature is validated), deployed and finally published.

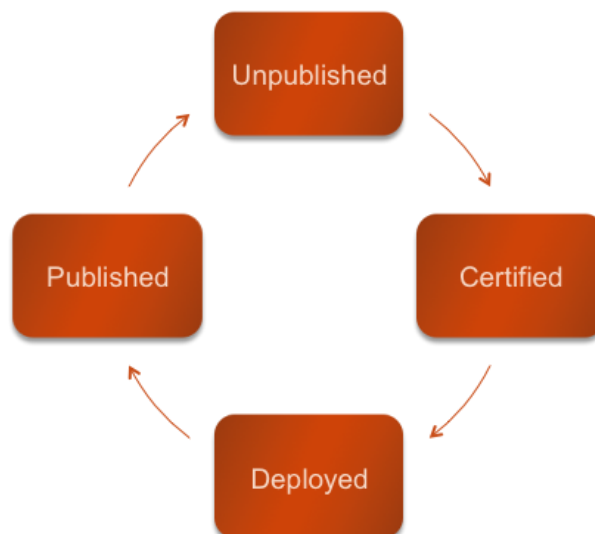


Figure 14 – BUTLER Application life-cycle

Once deployed, the app is listed within BUTLER SmartMobile client, ready to be installed and available as a responsive mobile web app.

7. CONCLUSIONS AND FURTHER DEVELOPMENTS

Even if it will be improved and completed during the third year of the BUTLER project, the BUTLER SmartMobile framework already provides necessary components to deliver functional BUTLER applications. Indeed, as described earlier in this document, it provides a framework to build hybrid and web applications, and tools to deploy and make them available to end-users. Thus, it covers the full production workflow of an application: development, deployment, and delivery. Moreover, it makes use of modern web technologies in order to give developers the power to build rich Internet of Things applications.

While it is a working framework, next steps will help in evaluating BUTLER SmartMobile framework performance in real conditions through trials in order to assess the potential of the framework and to update it where needed.

At this stage, multiple applications have been developed using BUTLER SmartMobile, either using the full framework or being native (and integrated within the BUTLER Applications Repository, using the BUTLER user interface and being launched from BUTLER SmartMobile):

- Hybrid and mobile web applications: diabetes app, security app (in order to demonstrate the end-to-end security paradigm), manual positioning control and SmartOffice (work in progress);
- Native applications: energy/augmented reality, follow me app, pick me app, energy visualisation

During next iterations, various components will also be improved. For example, the end-to-end security mechanism will be fully made available within the BUTLER library so developers will be able to use it in a simple way, thus authenticating the user and securing data exchanges between applications and sensors (or more generally data providers).

Also, access to low-level components, such as NFC readers will be added to the library. It is worth noting that access to other SmartMobile sensors will also be integrated, taking into account users' consents so applications will only be able to use data authorized by the user. On that topic, specific user interface screen will be added to the BUTLER SmartMobile client in order to allow users to authorize and deny access to their data in a simple and highly understandable way.

Additionally, as initially planned the support for iOS devices will be added. At the time of writing, only Android devices are supported. But since most of the BUTLER SmartMobile framework relies on HTML5, the support of iOS devices is expected to be straightforward, excepted for low-level components that relies on low-level code (such as for instance the component to access NFC reader).

Once this will have been achieved, BUTLER SmartMobile framework, and in consequence BUTLER applications, will be available on a wide range of devices, including mobile devices (phones and tablets), TVs, laptop and other emerging devices such as watches for instance.

8. REFERENCES

- [1] Butler Consortium, D2.4 – Selected technologies for BUTLER Platform. Deliverable of the Butler Project (April 2013)
- [2] Butler Consortium, D3.2 – Integrated System Architecture and Initial Pervasive BUTLER proof of concept (October 2013)
- [3] Butler Consortium, D4.1 – BUTLER SmartServer Platform and Enabling Technologies- Deliverable of the Butler Project (October 2013).
- [4] PhoneGap framework. Web page at <http://phonegap.com/>
- [5] Bootstrap 2.3.2 documentation. Web page at <http://getbootstrap.com/2.3.2/>
- [6] Single-page application. (2013, November 8). In *Wikipedia, The Free Encyclopedia*. Retrieved 15:44, November 18, 2013, from http://en.wikipedia.org/w/index.php?title=Single-page_application&oldid=580685962