MALTA INFORMATION TECHNOLOGY AGENCY
TRANSFORMATION • INNOVATION • EXCELLENCE

Government mServices

Technical Guidelines v2.0

Publication date: May 2019

CONTENTS

FIGURES AND TABLES

*depends on the technology/ frameworks used

1 EXECUTIVE SUMMARY

This guidelines document, complements the Mobile Government Strategy and provides a baseline set of

- directions on how to engineer native, hybrid and web mobile applications.
- governance measures with which these applications need to adhere to.

In summary, this document makes the following **key recommendations**:

1. The selection of which **deployment model** to consider (i.e. native, hybrid or responsive/form factor adapt) depends on a number of considerations – this document provides **guidelines** on when to adopt which.
2. The native build process will be **federated**[1] (in the sense there will be no central dependencies). P**ublishing** (in the respective stores) will be performed and governed centrally by MITA.
3. Specifically, for **web based**, mobile application development which interact with back office services, the primary technology to use needs to be **Javascript**, with focus on **Node.js** and **MongoDB**.
4. Both Web apps and Native apps will capitalise on the same solution architecture including the use of core building blocks such as web services to deliver the required services, irrespective of the medium or channel used.
5. Within MITA, **Xamarin and Ionic** are the preferred software development tools of choice– the reasons relate to:
    - Provides the capability to have a single code base for multiple platforms
    - Leverage's the platform strengths for optimal user experience
    - Integrates well with our mainstream engineering practices and processes
    - Can capitalize on current skill sets and development tools (IDE)

    MITA however will not mandate or restrict the use of tools/frameworks on 3rd party suppliers and/or contractors in the context of mobile app development.
6. App deployment to respective stores will be **governed centrally** by MITA

[1] This may require an upfront investment in a number of devices required for building and testing apps.

  - This does not inhibit the development (full lifecycle) of apps in a federation fashion
  - There will be a maintenance window for mainstream deployments – exceptions will be governed by a separate process
  - All published government mobile apps will be published under the Government of Malta

*depends on the technology/ frameworks used

2 BACKGROUND

Mobile application engineering is inherently different from traditional desktop applications. Designing and engineering for mobile needs consideration of how and whether to exploit features such as positioning systems, motion sensors, cameras and microphones. Considerations for battery use, computing resources and network disconnections are also challenges that need to be dealt with at the initial stages of the design. For the sake of this document and until the final strategy document is in place, the key considerations and approaches that need to be undertaken in this construct are grouped into 4 dimensions, namely Data, Security, User Experience and Platform *(refer to Chapter 3)*.

3 PROCESS GUIDELINES

The general engineering discipline is not yet geared fully and mature for mobile app development - this from a skills perspective, enabling platforms as well as internal/external processes perspective. Areas that require specific consideration are the SDLC, Security and general quality control, as well as standardization and streamlining of the development environments/processes to the maximum extent possible. Such challenges will become more amplified in the context where the mandatory federated approach for the engineering of native mobile apps is employed.

This section attempts to rationalize, a priori, a number of considerations as well as propose a baseline approach, from a **process perspective** (the software engineering part is discussed further on) to mitigate these challenges as we go along and mature in this construct.

3.1 Development approach

The following section provides an overview of how to identify those opportunities which can be exploited as apps as well as introduces a number of considerations from a process perspective in terms of the development tools to be considered, explains the basic deployment models that need to be considered as well as a discussion on how to best approach maintenance and updates.

3.2 Establish whether / which applications are good candidates

A simple decision framework is being described to help guide establish the appropriateness level of developing an app in the first place. These guidelines need to be applied in context to determine the relevance of app as well as identify a priori any potential risks that exist.

Perspective	Baseline
Is application graphics intensive?	As such these should be ‘avoided’ or heavily re-engineered, particularly when a native mobile app is being considered.

*depends on the technology/ frameworks used

Perspective	Baseline
Is the app core functionality computationally taxing?	Complex compute should not be processed on the end user terminal. If this precludes the app from having the required functionality, it should be either re-engineered or not considered.
What level of enterprise service integration does it need if at all?	If heavy integration is required, then the best candidate deployment model should be an enterprise class web application with the pre-requisite of being form-factor adapt/ responsive.
Connected vs. disconnected use of app features	Mobile applications might need to cache/ store data to provide the user with limited functionality of the mobile app even when the device is not connected to the internet. The adoption of such an approach should be given due consideration given that it has an impact on the engineering of the mobile app, and indirectly on the costs associated with its development.
What is the designated security profile of the application in the context of identity and data processing?	Complexities surrounding data security and privacy in the context of mobile app development is further amplified vis-a'-vis how data is accessed and how it is stored/ cached (retained on the mobile). Depending on the security profile and classification of the data being accessed, the application might require authentication/ authorization to access such data, and possibly need to encrypt any data/ tokens used to access such information[2]. Retention of such information is also tricky and might require further investigation as to how the app development should progress; a retention period for such data should be established and identified, and should be established on a case by case basis.

[2] IOS provides a local secure storage called KeyChain - https://developer.apple.com/library/ios/documentation/Security/Conceptual/keychainServConcepts/01introduction/introduction.html#//apple_ref/doc/uid/TP30000897-CH203-TP1

Perspective	Baseline
Payments and Authentication Functionality	Specifically, for mobile apps having Payments or Authentication functionality (citizens via eID and public officers via CORP), MITA shall look at each scenario of requirement on a case by case basis so as to suggest/ apply the applicable architecture design considerations.
What is the anticipated update/refresh lifecycle requirement of the app?	Identifying the refresh period of an app is crucial in determining whether it is viable to have it developed as a native/ cross platform mobile app or develop it as a web app (responsive web site). Such considerations should factor in the complete app deployment cycle which in some cases might take weeks to be officially published in a store. Periodically new updates to frameworks (used for the development and build of apps) should be considered to ensure currency of technology/ frameworks adopted by the published apps – deprecation and framework updates might have serious repercussions on the app features (particularly those dependent on plugins supplied by the framework usage).
App Support	The consideration to move to mobile app development/ production should also take into consideration the periodic review of the published apps themselves. Periodically a revision of the published apps should be undergone to ensure that they are still supported in terms of device support/ compatibility (new releases) as well as ensure that any APIs used by the app are still active (deprecated etc.).
What is the scope (in terms of extent and reach) of the app?	Apps should be developed to add (new) value added services or to extend existing functionality provisioned via traditional desktop/ web applications – it is important to note that this may imply a change in or addition to the existing business processes. Functionality which already exists and which is perhaps already available via a responsive web site should be carefully evaluated prior being 'replicated' onto a mobile platform as a native app. The scope, target audience and the frequency of use from the consumer's perspective is also something which needs careful consideration given the app development/ re-engineering costs incurred.

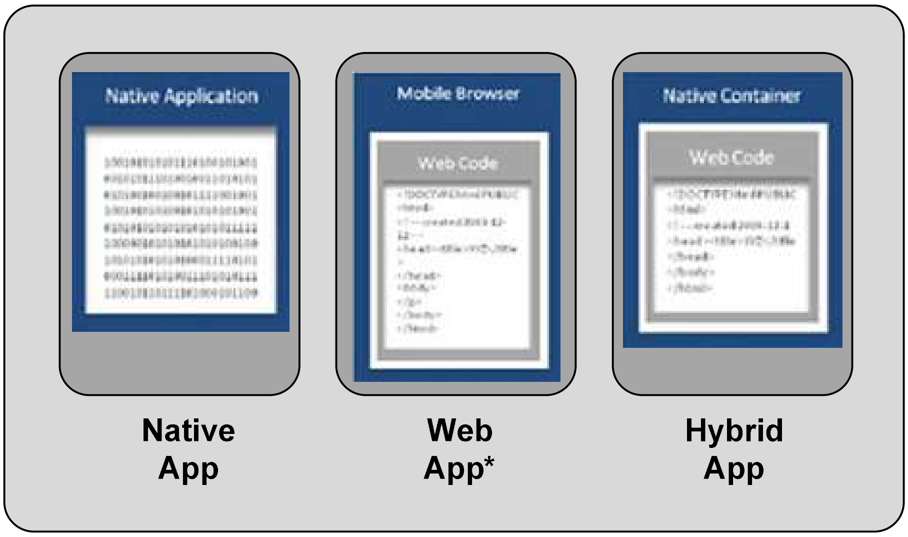
*depends on the technology/ frameworks used

Perspective	Baseline
The use of certain features in mobile app	Depending on the mobile app business nature, certain features might be required to enhance the UX/ UI. Due consideration of such features needs to be carefully evaluated and scrutinized due to 1. Complexities surrounding the mobile app engineering to expose such features (such as the use of GPS); 2. Notifications: Depending on the type of notification required (including whether it is a broadcast or personalized) certain considerations related to the approach of how to identify users or rather subscribers, is something which needs to be looked into on a case by case basis. From a business perspective it is important to ensure that the use of such features will not introduce unnecessary complexities when alternative mechanisms (such as the use of SMS) might be more practical from a consumer perspective (requires data connectivity vs. none) and involve additional costs.
Legal Obligations, Data Persistence and Privacy	**Depending on the business nature and target audience, the app might need to satisfy some legal obligations related to the retention of data. Why such data is being retained, its duration and the purpose should be clearly defined and agreed upon with the app owner.** **End users should always be made aware about such information and its purpose of use and retention.** **The app owner is responsible for the app and its legal aspects including any terms of use of the app. MITA on behalf of Government is responsible for publishing the respective app/s to stores.**

Table 1: Technical criteria

3.3 Development/Deployment models

A number of **development/ deployment models** are available when it comes to the final delivery and packaging (for distribution/consumption) of mobile applications. These are **Native App**, **Web App** and **Hybrid App**. A summary of the key features of each is provided below.



*Web Site / HTML5 responsive

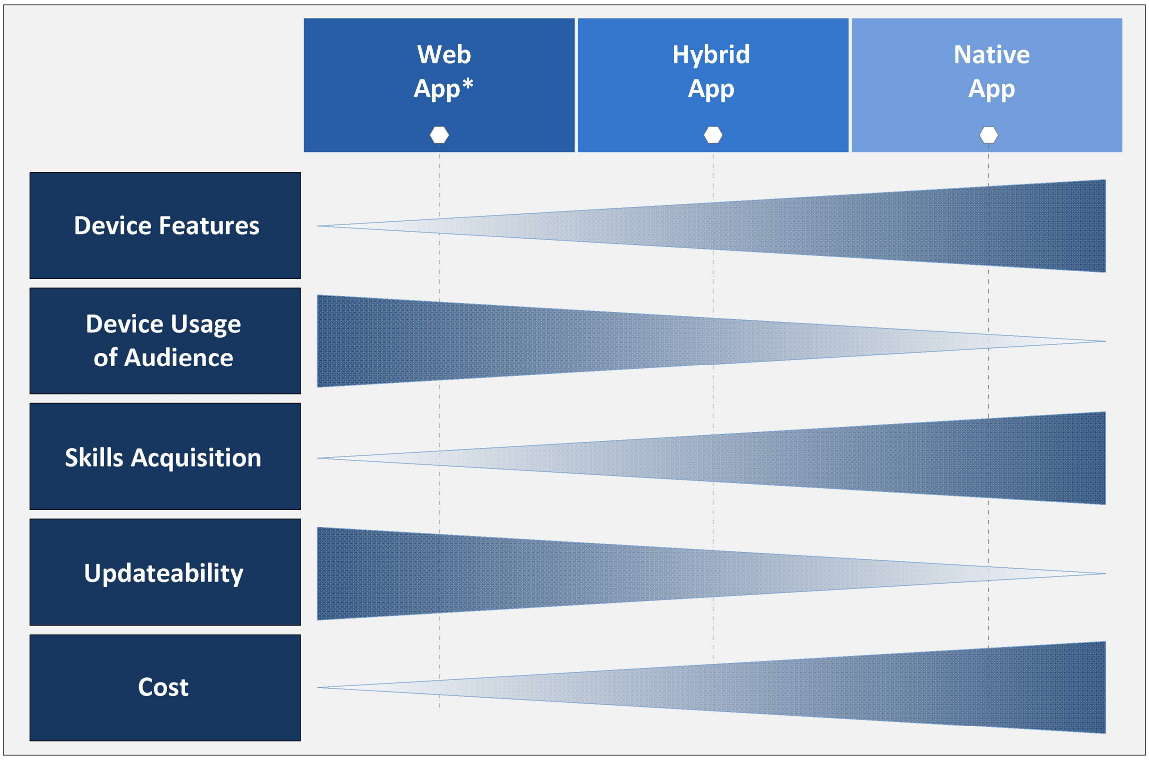
Figure 1: Development/Deployment models

- **Native** apps are specific to a given mobile platform[3] (iOS, Windows or Android) using the development tools and language that the respective platform supports (e.g., Xcode and Objective-C with iOS, Eclipse and Java with Android, Visual Studio and C# with Windows). Native apps look and perform the best.
- **Web** apps use standard web technologies—typically HTML5, JavaScript and CSS. This is primarily a write-once-run-anywhere approach to mobile development creates cross-platform mobile applications that work on multiple devices. While reasonably sophisticated apps can be created with HTML5 and JavaScript alone, some vital limitations remain at this point in time such as functionality related to session management, secure offline storage, and access to native device functionality (camera, calendar, geolocation, etc.)
- **Hybrid** apps enable the embedding of HTML5 apps inside a wrapper native container (such as Cordova and Ionic), combining the best (and worst) elements of native and HTML5 apps.

[3] Latest Blackberry phones are shipped with android.

*depends on the technology/ frameworks used

Figure 1 provides an overview of the pros and cons related to the challenges encountered depending on the type of development/ deployment model adopted.



*Web Site / HTML5 responsive

Figure 2: Pros and cons of different development/deployment models

In general, a key perspective that should be followed during the decision making process is the consideration of **native User Experience (UX) performance versus portability** because these have broader consequences on the application **manageability**, **time** to **market** and **cost**.

In general, during the early project stages, the following key principles need to be considered because they will influence the final selection of the deployment model, namely:

1. As the need for **application portability increases**, the ability to maximize on performance and richness of the native mobile platform and UX decreases.
2. **Code re-use capability decreases** with **maximization of native features** – this in turn will increase the cost of the respective maintenance lifecycle as well as increase the time to market, significantly.

The advantages and disadvantages of each **deployment** approach (*Native/Hybrid/Web*) are very well documented, and summarized in the following table. Mobile development is a constantly moving target. Every six months, there's a 'new' version / flavour mobile operating system, with unique features only accessible with native APIs. The containers bring those to hybrid apps soon thereafter, with the web making tremendous leaps every few years.

Based on current technology, one of the scenarios described here is bound to suit specific requirements.

App / Devices Features	Native	Web	Hybrid
Performance	Very Fast	Fast	Fast
Distribution	App Store	Web	App Store
Connectivity	Online and offline	Mostly online	Online and offline
Development skills	ObjectiveC, Swift, Java, Kotlin	HTML5, CSS, Javascript	HTML5, CSS, Javascript
Offline storage	Secure file storage	Shared SQL	Secure file system, shared SQL
Graphics	Native APIs	HTML, Canvas, SVG	HTML, Canvas, SVG
Native look and feel	Native	Emulated	Emulated
Camera/ Video	Yes	Yes*	Yes
Notifications	Yes	Yes	Yes
Contacts, calendar	Yes	Yes*	Yes
Geolocation	Yes	Yes	Yes
Swipe	Yes	Yes	Yes
Pinch, spread	Yes	Yes*	Yes
Multi touch	Yes	Yes*	Yes*

Table 2: Advantages and disadvantages of different deployment models

*depends on the technology/ frameworks used

3.4 Software Engineering tools/IDE's

There are various development environments / tool chains which can be considered for use. It is safe to state the primary focus (not necessarily sole) is environments which target Windows Operating environments and mostly follow the Fat Client Server or Web Application/Services pattern. Mobile development introduces another build context and whilst traditional Integrated Development Environments (IDE) compliers are being evolved with such functionality, the most prevalent and popular environments within the industry are specific tools or tool chains. These include **Xamarin and Ionic amongst others**. At this stage, in a MITA construct (based on market trends, current internal skill set and strategic direction) the development tool of choice should be **Xamarin and Ionic**. This beside the tight integration with the prevalent IDE/Tool Chains/Operating environments in use within MITA's software development arm.

For **web** or **hybrid** apps which are heavily baselined in **HTML5**, **CSS** and **JavaScript**, the consideration for technologies such as **Cordova, MongoDB** *(or platform specific alternatives)* and **Node.js**[4] is believed to be sensible. This approach allows developers to apply the same skill set in both front and back end constructs.

MITA however will not mandate or restrict the use of tools/ frameworks on 3rd party suppliers and/or contractors in the context of mobile app development.

3.5 Development/Testing deployment to respective app stores

In the context of MITA, a federated build process will be established. A federated Android/Windows/IOS development/testing environment (*based on the development environments described above*) will need to be set-up with any specific configuration requirements (including any investments required) established and made available a priori within each internal development team.

Publishing (in the respective stores) will be performed and governed centrally by MITA (Mobile Initiatives Team).

[4] Event-driven I/O server-side JavaScript environment.

3.6 Maintenance and updates

Updates cycles to mobile apps should be controlled. Whilst a degree of flexibility is to be in place for specific scenarios, a set of pre-defined update windows will be put into place and developers will be expected to schedule deployment to app stores within this schedule.

Updates must be concurrent with non-mobile app counterpart (such as web site and back office process) to ensure service delivery coherency across different consumption channels. This becomes a critical consideration in the context of the limited ability to influence the publishing timeframes of mobile apps to respective stores, especially when dealing with IOS variants.

3.7 Push Notifications

MITA, on behalf of the Government of Malta, does not provide the Firebase service. The current Government subscription with Android, Apple and Microsoft stores only caters for publishing of mobile apps.

Where applicable, the engineered solution (mobile app) should cater for the push notifications requirements including any subscriptions with third-party (for example Google Firebase) to be able to provision the required functionality. MITA, can provide the required APN key and Apple push notification certificate to be used in conjunction with third-party push notifications service – these shall be supplied to the CIO upon request.

*depends on the technology/ frameworks used

4 SOFTWARE ENGINEERING GUIDELINES

There are 4 key dimensions that need to be looked into considerably differently when developing mobile apps when compared to traditional client server approaches. These dimensions, Data, Security, UX and Platform, are discussed in the following sections with an associated set of guiding principles provided to ensure a coherent and sound approach and which should form a baseline pattern for the first wave of mobile apps – keeping in mind the opportunity for future evolution whilst still remaining relevant.

4.1 Data

Online/offline operation – The mobile experience would be badly affected if it depends on persistent internet connection. Mobile apps need to leverage local caching mechanism to deliver a smooth user experience while negotiating "dead-zones" and network signal variance.

Persistence and Privacy[5]

Data persistence or caching allows users to use the app (possibly with limited features) while their connection to the World Wide Web is down. Data persisted on the device should be considered and addressed on a need basis. Depending on the security and sensitivity of the data related to the particular scenario, the developed application should resort to encryption to store sensitive data, and decide which best storage mechanism to adopt (session or otherwise). Data persisted on the device should have a retention period set, to ensure that data is retained for the intended scope (for example for the duration of the active session or time period) and for a predefined period of time, thus reducing risks associated with data security.

Depending on the scenario and target audience of the mobile app there might be instances which mandate abidance to/ with a set of rules, legal acts and/or regulations.

[5] http://www.gsma.com/publicpolicy/wp-content/uploads/2012/03/gsmaprivacydesignguidelinesformobileapplicationdevelopmentv1.pdf

https://www.gsma.com/publicpolicy/wp-content/uploads/2012/03/GSMA2016_Guidelines_Mobile_Privacy_Principles.pdf

Latency-design for intermittent connectivity

Mobile applications should assume that the connection to any internet service is unknown, thus handle any failure to connect to an external service. In such cases the application should deny access to specific features which require such access. Depending on the case and the application features, the use of cached data might be considered to allow smooth running of the application in such

Push & Sync

Apps should refrain from being intrusive on any level. They should offer the user to block any notifications and updates, informing the user with the due implications. Apps should support automatic, manual and hybrid refresh functionalities as endorsed by the respective platform vendors, depending on the nature and features provided by the mobile app. Data synchronization depends on the nature of the application – data can be refreshed when the application is launched or through the provision of data refresh/ sync features, initialized directly by the user.

Web Service API's

Mobile apps should leverage the use of **REST** based web service APIs to exchange data from/ to the mobile device. This approach would allow users to multitask within the app itself, optimizing user experience while the data is being transferred. These web service APIs are utilized for all end-user applications, irrespective of the medium so as to ensure that there is seamless (data) conformance between the mobile app (native or hybrid) and the web app version (responsive web site).

4.2 Security

The general guidelines reflected in the document published by ENISA titled **Smartphone Secure Development Guidelines for App Developers**[6] should be followed.

[6] https://www.enisa.europa.eu/activities/Resilience-and-CIIP/critical-applications/smartphone-security-1/smartphone-secure-development-guidelines

*depends on the technology/ frameworks used

In the context of Security OWASP[7] provides various guides/ checklists *(some of which are still works in progress)* which outline several areas that need to be addressed. Furthermore, the following considerations also apply.

Encryption at rest and in transit

As outlined in the previous section (4.1), sensitive data persisted on the device's local storage and during transit should be encrypted. Ideally sensitive information (such as passwords) is not stored on the device. In case passwords need to be stored on the device, the encryption and key-store mechanisms provided by the mobile OS can be used to securely store passwords, password equivalents and authorization tokens. Instead of passwords consider using authorization tokens that can be securely stored on the device (as per the OAuth model). The tokens should be time bounded to the specific service as well as revocable (if possible server side), thereby minimizing the damage in loss scenarios.

Authentication

In the case of authentication various protocols like OAuth2 can be used to securely authenticate/ authorize the user to consume the web service APIs servicing the mobile app. Depending on the type of service/ data being consumed, appropriate authentication mechanisms should be used. Session management is another aspect which needs careful consideration to ensure that any requests from the device are seamless and legitimate to the authenticated user.

4.3 User Experience

In order to optimize and deliver an intuitive app, it is important to identify and limit the scope of functionality provided by the app – this means that the apps delivered should ideally focus on providing specific, focused services as opposed to mimic and/or replicate all the functionality provided by the entity's core business application/s.

User experience is heavily influenced and quality controlled by the respective mobile platform vendor. In this respect, first and foremost, the engineering roles must acquaint themselves with a number of guidelines presented by the respective platform[8] vendors. It is important to outline

[7] https://www.owasp.org/index.php/OWASP_Mobile_Security_Testing_Guide

[8] **Apple**: http://developer.apple.com/library/ios/#DOCUMENTATION/UserExperience/Conceptual/Mobile

that although there are commonalities between the respective stores, there might be some specific requirements pertaining to individual stores.

Initial design considerations should be performed via **wireframes** or **mockups** using tools such as **Balsamiq**, **Mockingbird**, **Visio** etc. These allow the designer to quickly design UX without having to worry about the actual UI design. Key considerations to be kept in mind are:

1. Design based on **device capability** and be able adapt the content to be delivered.
    a. Content scaling to optimize use of screen real estate and resolutions.
    b. Video formats need to be adapted to the device supported formats Interaction mechanisms need to be customized based on device capabilities.
2. **Simplicity** – An effective mobile layout needs to simple. The information presented needs to be structured into a layout that minimizes scrolling and the need for zooming. For example, a single column layout is generally preferred to avoid cluttering and horizontal scrolling.
3. **Navigation** – Mobile applications need to provide an easy way of navigating between 'pages/tabs' and make optimum use of the device capabilities.
    a. The navigation mechanisms in touch based smartphone are primarily icon and tab driven, while the navigation is menu driven for devices with keyboards. Reducing the number of touches to perform an action is necessary to enhance user experience.
    b. Minimize gestures/input– adopt OS specific conventions.
4. **Performance** – Mobile apps need to provide fast response times to the user. This can be achieved by keeping their size small, local caching and usage of background processing to provide a seamless experience.
5. **Accessibility** – Given that accessibility guidelines are platform specific it is imperative that the mobile app (native/ hybrid) is also available as a responsive web application which is designed with accessibility in kind. This approach would ensure that the mobile app (native/ hybrid) is also available as a web app (running on the mobile device web browser as a responsive web site) providing the necessary mechanisms to make it accessible to all intended users. Bundling separate modules, providing easier interaction with the application, could also be an option of consideration.

HIG/Introduction/Introduction.html; **Android**: http://developer.android.com/design/index.html; **Windows**: Design library for Windows Phone - http://msdn.microsoft.com/en-US/library/windowsphone/design/fa00461b-abe1-41d1-be87-0b0fe3d3389d(v=vs.105).aspx

*depends on the technology/ frameworks used

W3C WAI accessibility standards and guidelines provide insights on “Mobile Accessibility”, and although the standard is still evolving there are already a number of aspects related to mobile accessibility which the existing standard covers – for further information refer to https://www.w3.org/WAI/mobile/.

Reference Appendix D *FITA Web Accessibility Directive*

4.4 Platform

The development of mobile apps should target as a minimum the below platform version:

- Android 6 and above (however the minimum target version should be Android 8/ API level 26 - https://developer.android.com/distribute/best-practices/develop/target-sdk)
- iOS 10 and above
- Windows 8 and above (*optional*)

If the development tool/ framework/ component in use and/ or functionality mandates a more recent version to be used, then the oldest version possible should be used.

5 OTHER CONSIDERATIONS

5.1 UI Design – Branding

A degree of consistency in terms of look and feel is considered fundamental. A reference style guide has been published to aid developers achieve a coherent branding approach, specifically for the development of mobile apps. It is advisable to reference the targeted /leading platform vendors' UI design guidelines in order to ensure the mobile app is compliant with the respective store/s requirements.

Key areas that need a degree of standardization in this construct are:

1. **Splash Page** – A splash screen for all government published mobile apps shall be designed so as to ensure a common user experience when loading government mobile apps.
2. **Logos** – Mobile apps should refrain from displaying the logo on various screens. Users need to clearly distinguish the icon from other apps in the launch screen, possibly show it again on a splash screen and hide it throughout other screens.
3. **Icons and Images** – All images and icons should be tailored to mirror the native look and feel of the respective platform.
4. **Colour Scheme** – The main colour scheme should be defined per application. Colours should be used to enhance usability rather than define it. This makes it easier to cater for a number of borderline cases such as (but not limited to) colour-blindness and using the app in direct sunlight. The schemes used for the mobile app should be similar to the colour schemes used by the government/ department/ entity website.
5. **Font** – All government mobile apps should follow the standard font types and sizes identified and established in the mobile app branding guidelines. The latter guidelines will be based on the major targeted platforms' guidelines.

5.2 Testing

Because the deployment lifecycle is much more taxing than typical fat client server and even more web based application development, testing and quality control become a fundamental requirement. The development team shall design, develop, build and test the full functionality of the mobile app, prior submitting the application for publishing (to the respective app stores) which will be done by MITA.

Testing should cover the various aspects of mservice including but not limited to:

- Functionality – engagement of beta testers (also outside business domain)

*depends on the technology/ frameworks used

- Security – reference OWASP Mobile Security Testing Guides/ Checklist[9]
- User Experience/ User Interface – responsiveness etc.
- Conformance with published style guide
- Performance / Stress testing – on all components of the solution (including APIs, backend services etc.)

Various tools are available as subscription/ pay as you go services, which can aid during the testing of mservices.

5.3 Stores

There are various ways how government mobile apps can be distributed. Two popular methods are via app stores, either public or private. The decision of which store should be used depends on a number of considerations which typically include aspects surrounding security (for the type of service/ app being provisioned), consumer reach and the level of app management required. In the context of the current maturity landscape as well as objectives related to app publishing (at least for the immediate future), the public app stores are potentially the most viable solution – cheap to use, leverage a model that targeted consumers (citizen) are familiar with and is less intrusive (i.e. do not require consumers to connect to a different ‘corporate’ app store). MITA shall centrally govern the publishing of the respective apps in the public stores, in order to sign and certify the application as a legitimate government mobile app published by the Government of Malta.

It is important to note that the public app stores are not designed to provide device management required to deploy/ push apps on restricted devices – hence restrictions to use the app should be handled by the app UI. Such stores typically provide basic statistical features including but not limited to device types, number of downloads, ratings etc.

It is being envisaged that each developed app should ideally (and as a minimum) measure/ track user activity. Typical user activity logging might include login/ logout events, authentication issues (exceptions) and methods of authentication, method of access (WiFi/ 3G), and duration of activity within app. Retention of such activity logs, will differ according to the applied scenario and app in question.

[9] https://www.owasp.org/index.php/OWASP_Mobile_Security_Testing_Guide

Publishing the APP with a unique APP Package ID

The mobile apps that shall be published through MITA, shall follow the following naming convention

mt.gov.*<application name>*, thus ensuring that the package ID is unique and conforms with a naming convention which is associated with the Government of Malta.**Note:** should the respective App need to be included in the *Government Launcher* mobile app – refer to **Appendix A and B** for additional details.

5.4 Languages

The majority of applications, that will be developed showcasing a number of government mservices, will use English as their default/ preferred language of choice. Besides the default English language, the mobile app must also provide the functionality to switch to Maltese language – hence as a minimum the mobile app should have both English and Maltese language support. This rationale conditions the landscape of the mobile app development in that any engineered apps should take into consideration localization aspects form the very start of their development, thus providing the ability for the mobile app to change the apps base language with minimal or no effect to the source code. Typical engineering considerations and prerequisites supporting such rationale include:

- **Externalisation of resources** - one of the very first steps in the app localization process is to externalise resources (language files) – each language having a separate language resource file. Various leading platform vendors provide details and toolkits on how to approach this matter. *Note*: Details related to the app published in the app store should also be taken into consideration, however the preferred language for that should be English.
- **Layout** – the design should take into consideration the length of words and sentences in languages other than English. Such considerations might include spacing, left-to-right and right-to-left support, as well as other factors – hence a flexible layout that scales according to the language used is something which needs careful consideration. Other considerations linked to the design and layout of the app relate to the system-provided formatting methods for dates, times, number and perhaps currency.

The option to choose the language to use in the mobile app will be determined from within the app itself – i.e. it will not be determined or chosen through the device's language settings.

5.5 Contact/ Feedback/ Suggestion/ Comments Form

Each published mobile app should provide a mechanism of allowing citizens to provide feedback/ suggestions to the service owner. In this context each published app should feature a simple

*depends on the technology/ frameworks used

form/ screen allowing users to exchange such feedback/ comments related to the mobile app itself.

6 APPENDIX A: GOVMT LAUNCHER FUNCTIONALITY

The mServices on smartphones will be accompanied by an application acting as a central hub for all government applications - **maltapps**. The application will allow users (citizens) to outline the respective mServices listed under specific government sectors and provide the ability to launch the native app or the responsive website relating to the chosen service.

Specifically for applications (native and hybrid), the launcher will make use of deeplinking to launch the app installed on the device. Thus, each application submitted to the store will need:

1. **A unique URL Scheme**

   The convention being adopted will assimilate "**govmt*Name***", where *Name* is the name of the app.

   Note: MITA will only need the *App Name*[10] (supplied in the table below) to generate the URL Scheme. Once this has been generated the URL scheme shall be supplied back to the owner/ supplier to be configured in the respective app.

2. **Allowing remote intents**

   The application must request the appropriate permissions (from the OS) allowing the launcher to trigger the remote intent. This can be done in various ways, depending on the tools being used. In general, it consists of a simple checkbox for the appropriate permissions.

3. **Inclusion of parameters *[OPTIONAL]***

   The mobile app could allow external parameters to be passed upon invocation (via deeplink). Such parameters could include the language option for example.

[10] The *App Name* can be supplied in advance of the screen shots and icons required – i.e. prior the completion of development of the actual app.

*depends on the technology/ frameworks used

7 APPENDIX B: PUBLISHING DETAILS

MITA will be the sole publisher of all the applications on the public stores. Thus, the relevant details of each mobile application are to be sent to the Mobile Government Programme Team as early as possible. The following checklist outlines the required information:

Generic

- **Description:** A description of the app, detailing features and functionality. The description should be in English and Maltese.
    - Android, Apple and Windows - Maximum 4000 characters (English and Maltese combined).
- **Contact Information (Name, Surname, Phone Number, Email):**
    - Apple – The person in your organization who should be contacted if the App Review team has any questions or needs additional information.
    - Google and Windows – The email address will be publicly displayed within your app.
- **For mobile app updates ONLY**:
    - What's new in the release.
- **For Focus Groups BETA Testing:**
    - **(Apple) What to Test**: Include Information for your testers, such as instructions to test specific features.
    - **(Apple) Feedback Email**: The email address where TestFlight Beta Testers can send feedback to.
    - **User details who are going to install BETA test app**:
        - Name of individual,
        - Email account used on mobile device to install apps (the one used with iTunes, Playstore, Windows),
        - Platform OS.

Apple

- **Promotional Text:** lets you inform your App Store visitors of any current app features without requiring an updated submission. This text will appear above your description on the App Store for customers with devices running iOS 11 or later.
- **Keywords**: One or more keywords that describe the app in English and Maltese. Keywords make App Store search results more accurate. Maximum 100 characters (English and Maltese combined).
- **Category**: To select one from the following list - https://itunesconnect.apple.com/itc/static/category_definitions
- **Support URL**: A URL with support information for the app. This URL will be visible on the App Store.
- **Copyright**: The name of the person or entity that owns the exclusive rights to the app, preceded by the year the rights were obtained (for example, "2008 Acme Inc."). Do not provide a URL.

- **App Icon** (1024x1024): This icon will be used on the App Store and must be in the JPG or PNG format, with a minimum resolution of at least 72 DPI, and in the RGB colour space. It must not contain layers or rounded corners.
- **Screenshots** (72 dpi, RGB, flattened, no transparency, High-quality JPEG or PNG image file format) having the below resolutions:
  - 5.5-Inch Display (up to four screenshots): 1242 x 2208 pixels for hi-res portrait, 2208 x 1242 pixels for hi-res landscape
  - 12.9-Inch Display (up to four screenshots): 2048 x 2732 pixels for hi-res portrait, 2732 x 2048 pixels for hi-res landscape.

Android

- **Short description**: A short description of the app in English and Maltese - Maximum 80 characters (English and Maltese combined).
- **Category**: to choose one from the following - https://support.google.com/googleplay/android-developer/answer/113475?hl=en
- **Android content rating**: To choose one category of the following:
  - REFERENCE, NEWS, OR EDUCATIONAL
  - SOCIAL NETWORKING, FORUMS, BLOGS, AND UGC SHARING
  - CONTENT AGGREGATORS, CONSUMER STORES, OR COMMERCIAL STREAMING SERVICES
  - GAME
  - ENTERTAINMENT
  - UTILITY, PRODUCTIVITY, COMMUNICATION, OR OTHER

- **Website:** please provide website URL.
- **Screenshots**: JPEG or 24-bit PNG (no alpha). Min length for any side: 320px. Max length for any side: 3840px. At least 2 screenshots are required overall. Max 8 screenshots.
- **Hi-res icon**: 512 x 512. 32-bit PNG (with alpha).
- **Feature Graphic**: 1024 x 500. JPG or 24-bit PNG (no alpha).

Windows

- **Screenshots**:
  - 768 x 1280, 1280 x 768, 720 x 1280, 1280 x 720, 800 x 480 or 480 x 800 pixels
  - Max upload 8 images
  - Landscape or portrait
  - Accepted file types: .png
  - Less than 5 MB
- **App Icon**: 300 x 300 pixels. Accepted file types: .png
- **Website**: please provide website URL

Web Apps Details (responsive website)

- * Website name
- * URL

*depends on the technology/ frameworks used

- * Icon (200 x 200px)
- Website Details

* Required for maltapps

Kindly send the above information (and any queries) to mobile-app.mita@gov.mt at your earliest.

8 APPENDIX C: DEPLOYMENT CYCLES

In order to introduce a degree of governance as well as discipline in terms of the mobile apps publishing life-cycle, the following schedule has been established.

Type	Example	Lead time *(weeks)*	Schedule
New Application	*Unpublished apps*	*4*	**Monday*
Platform issue	*Specific functionality misbehaviour on specific platforms/versions/form factors*	*1*	**Monday*
Bug Fix	*General quality control issue (e.g. validations, crashes etc.)*	*2*	**Monday*
Store Detail Updates	*Change in image screen-shots, description, icons, app names etc.*	*2*	**Monday*
Feature Updates	*New functionality, UX changes*	*3*	**Monday*
Exceptional scenarios	*Political commitments etc.*	*N/A*	*Date to be agreed on a case by case basis.*

**Monday: This will allow ample time for the stores to review and approve and potentially publish by end of the same week.*

*depends on the technology/ frameworks used

9 APPENDIX D: FITA WEB ACCESIBILITY DIRECTIVE

Foundation for Information Technology Accessibility

The Web Accessibility Directive is a ‛minimum harmonisation' Directive – meaning that it only sets out the absolute minimum requirements that have to be met by public sector bodies for their websites and mobile applications. These accessibility requirements will be the same across all EU countries but Member States are striving to achieve them before the deadline or improve upon them.

Scope of the Directive

The Directive applies to all websites and mobile applications of public sector bodies; these are defined as follows:

1. the State, regional or local authorities;
2. bodies governed by public law;
3. associations formed by one (or more) such authorities or one (or more) such bodies governed by public law, if those associations are established for the specific purpose of meeting needs in the general interest, not having an industrial or commercial character;

Web Accessibility Directive implementation deadlines.

23/09/18 Deadline for transposing the Directive into national law

23/12/18 Deadline for European Commission to publish the implementing acts related to the Directive

A model accessibility statement

Technical specifications for the accessibility requirements

A methodology for monitoring for compliance

Arrangements for reporting by Member States to the

Commis
sion

23/09/19 All websites created after 23rd September 2018 have to be accessible

23/09/20 All websites have to be accessible

23/06/21 All mobile applications have to be accessible

For additional information regarding accessibility please visit - https://fitamalta.eu/fita-services/consultancy-services/

*depends on the technology/ frameworks used

Exemptions or extensions applicable to existing content

Exemptions under the Web Accessibility Directive do not imply compliance with the Equal Opportunities Act anti-discrimination law. For our purposes any form of ICT based discrimination against persons with disability must be avoided, as part of our goal of delivering a quality service to the public.

1. Office file formats included in web pages:

   these are documents such as PDFs, Microsoft Office documents or their open source equivalents. Documents published before 23rd September

   2018 are excluded unless they are needed for active administrative processes relating to the tasks performed by the public sector body concerned. For example a downloadable form required for administrative purposes on a university's website cannot be exempted.

2. Pre-recorded time-based media published before 23rd September 2020.

   This mean that pre-recorded time-based media published after 23rd September 2020 will have to be made accessible (meaning with text alternatives, captions and audio description or better solutions available at the time). Note that the sooner such a requirement is put in place, the faster it will become common place and drive associated costs

   down. Subject to the Equal Opportunities Act, a viable alternative is already required.

3. Live time-based media.

   This refers to video streaming which is not archived. Please note that if such media is re-published later or kept in the website, then it will be considered pre-recorded time-based media and this should be made accessible after a period of time, usually after 14 day. The Directive states that "when it is impossible to procure the relevant services in due time", the 14-day period might exceptionally be extended to "the shortest time necessary to make the content accessible".

4. Online maps and mapping services intended for navigational use as long as essential information is provided in an accessible digital manner, such as postal address and nearby public transport stops. This should be provided in a form that is simple and readable for most

   users;

5. Third-party content that is neither funded nor developed by, nor under the control of, the public sector body concerned.

   The Directive states that such content should not be used if it hinders or decreases the functionality of the public service offered on the website (or mobile application) concerned. For example, content used to organise consultations or forum discussions has to be accessible.

   However, user-contributed content which is not under the control of the public sector body concerned (for example: when a member of the public posts an inaccessible photo without alternative text or video without captions on a forum) is exempt;

6. Reproductions of items in heritage collections that cannot be made fully accessible for one of the following reasons:

   a. the incompatibility of accessibility requirements with the preservation of the item concerned or the authenticity of the reproduction; or

   b. the unavailability of automated and cost-efficient solutions to extract information content and make it accessibile;

*depends on the technology/ frameworks used

7. Content of extranets and intranets

For example websites that are only available for a closed group of people and not to the general public published before 23rd September

2019, until such websites undergo a substantial revision, like refurbishing or platform upgrade. Discriminating against employees with disability by enforcing a non accessible communication solution within an organization, is still violates the requirements of the Equal Opportunities Act, and unless the solution can be made accessible, alternatives must be provided.

8. Content of websites and mobile applications qualifying as archives

This refers to resources that only contain content that is neither needed for active administrative processes nor updated or edited after 23rd September 2019

NOTES

*depends on the technology/ frameworks used

NOTES

Back cover