**Nordic Folkecenter for Renewable Energies**

Folkecenter Mobile Application Report

The Information and Communications Technology (ICT) Programme

Winter Semester 2019

**Ionel-Cristinel Putinica**

**266123**

Table of Contents

**Abstract3**

**1 Introduction4**

**2 Requirements**5

2.1 Functional Requirements5

3 Analysis6

3.1 Use Case Diagram6

3.2 Use Case Descriptions6

**4 Design & Implementation12**

4.1 Architecture12

4.2 User Interface/Back-end Design and Implementation14

**5 Testing28**

**6 Results, Discussion, Conclusion and Project Future29**

**Sources of Information30**

**Abstract**

The importance of mobile phones in our everyday life and activities is undeniably unending. This is so because there is ongoing tremendous transformation in that mobile phones are no longer the ordinary communication device it used to be. It has become the colossal point of attention for individuals and businesses alike, courtesy of the various incredible features and opportunities that mobile phones offer. The cumulative progress of mobile technology, the availability and access to high speed internet and the remarkable communicative interface in these devices results into a whole level of new and innovative experience mobile computing. This is made possible through the development of mobile applications (mobile apps).

Thus, the result of these project is the final product of the Folkecenter’s need of a mobile application, which helps at promoting Folkecenter, its different events and renewable energy as a whole. This project report has the purpose of describing all the methods, stages and iterations that went into the development and implementation of this project. The app was developed by a single person, and one of the main requirements was that is it build in such a way that, in the future, if another person has to work on the application, it will be an easy task to do, such that this report has the purpose of simplifying this task.

The application has been fully designed with the end-user experience in mind, the app being developed to give the visitors of the Nordic Folkecenter a more enjoyable experience, including even a guided tour (in multiple languages). The user interface has been programmed using XML, and the back-end was implemented using Java. The app was envisioned to only run on Android mobile devices, but development for different platforms is possible in the future.

1. **Introduction**

The importance of mobile phones in our everyday life and activities is undeniably unending. This is so because there is ongoing tremendous transformation in that mobile phones are no longer the ordinary communication device it used to be. It has become the colossal point of attention for individuals and businesses alike, courtesy of the various incredible features and opportunities that mobile phones offer. The cumulative progress of mobile technology, the availability and access to high speed internet and the remarkable communicative interface in these devices results into a whole level of new and innovative experience mobile computing. This is made possible through the development of mobile applications (mobile apps).

Today, the availability of mobile apps is on the increase such that it is produce a noticeable change in the way humans feel and experience computing. Few years ago, in other to access the internet, check and read mails, one had to use the computer but today this has changed because computing is now carried everywhere in mobile phones. Imagine buying a train ticket on the go, this is something our ancestors never imagined or did. Imagine not going to the bank but still transfer money to family and friends. All thanks to app developers and **top app development companies.** No matter which, they have come to the rescue enabling easy life.

The developed product has the main purpose of giving the visitors of Nordic Folkecenter a more enjoyable experience, giving them an overall idea on what Folkecenter is all about, helping them keep up with the most recent news and events and even offering them the option to have a guided tour, in multiple languages, through the application.

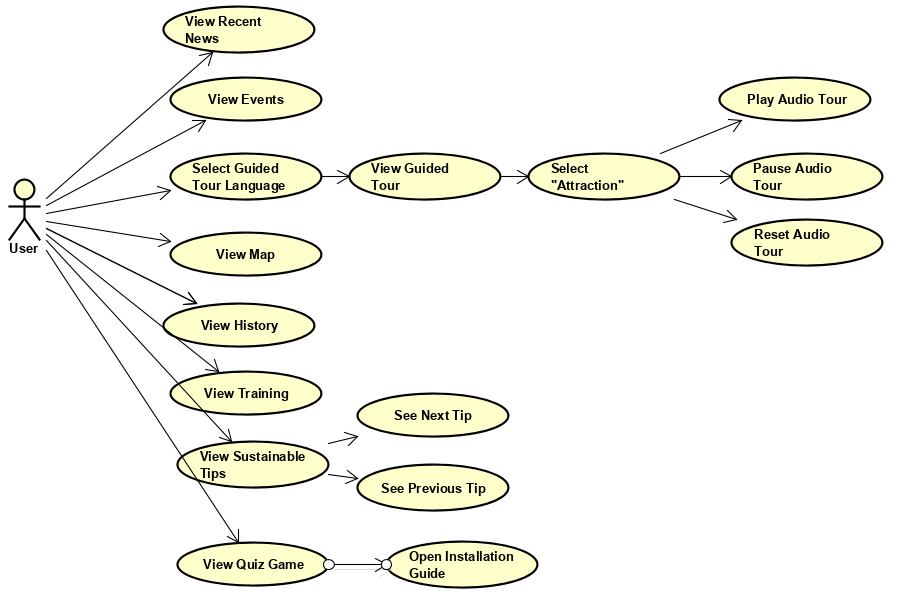
1. **Requirements**

**2.1 Functional Requirements**

|  |  |  |
| --- | --- | --- |
| ID | User Story Description | Priority |
| 1 | As a user, I want to be able to run the application on my Android Device | High |
| 2 | As a user, I want the application to be light-weight and well optimized | High |
| 3 | As a user, I want to be able to use most of the app’s features without a constant internet connection | High |
| 4 | As a user, I want the application to present me with relevant and well-organized information | Medium |
| 5 | As a user, I want it to be easy for me to access the Folkecenter website within the application | Medium |
| 6 | As a user, I want it to be easy for me to keep up with the latest news and events related to Folkecenter | Medium |
| 7 | As a user, I want to be able to have access to a guided tour, in my own language, within the application | Medium |
| 8 | As a user, I wish that, when using the application, as little data as possible is asked for me, and, if it is, this data is well-protected | Low |
| 9 | As a user, I wish for a crash-free experience | Low |
| 10 | As a user, I wish that the installation and deletion process is as easy as possible | Low |

*Table 1: Functional Requirements*

1. **Analysis**

****

*Figure 1: Use Case Diagram*

* 1. **Use Case Diagram**

The above Use Case diagram represents the various operations that the user can

perform in the application. Most of these operations are simple and straight-forward, since, to simplify the application, most of its functionality is viewing pre-defined data, rather than the user interacting with the app itself.

* 1. **Use Case Descriptions**

The following Use Case descriptions explain how the functionality of the system was envisioned to function described thoroughly.

|  |  |
| --- | --- |
| Use Case | View Recent News |
| Actor | User |
| Pre-Condition | The application needs to be on the front page layout  The device has an active internet connection |
| Post-Condition | The recent news are displayed |
| Base sequence | 1.User taps on the “Recent News” icon  2.The application loads in the recent news and displays them on-screen |
| Exception Sequence | 2A: The application cannot load in the news because the device is not connected to the internet |

*Table 2: View Recent News Use Case Description*

|  |  |
| --- | --- |
| Use Case | View Events |
| Actor | User |
| Pre-Condition | The application needs to be on the front page layout  The device has an active internet connection |
| Post-Condition | The events are displayed |
| Base sequence | 1.User taps on the “Events” icon  2.The application loads in the events page and displays them on-screen |
| Exception Sequence | 2A: The application cannot load in the events page because the device is not connected to the internet |

*Table 3: View Events Use Case Description*

|  |  |
| --- | --- |
| Use Case | Select guided tour language |
| Actor | User |
| Pre-Condition | The application needs to be on the front page layout |
| Post-Condition | The application displays the “guided tour” main page in the selected language |
| Base sequence | 1.User taps on the “Guided tour” icon from the front page  2.User taps on one of the 3 icons with the language: Danish, German or English |
| Exception Sequence |  |

*Table 4: Select Guided Tour Language Use Case Description*

|  |  |
| --- | --- |
| Use Case | Select Attraction |
| Actor | User |
| Pre-Condition | The application needs to be on the “guided tour” main page |
| Post-Condition | Application displays the selected attraction |
| Base sequence | 1.The user taps one of the icons, depending on the Folkecenter attraction he wants to learn about  2.Application displays the page for the selected attraction |
| Exception Sequence |  |

*Table 5: Select Attraction Use Case Description*

|  |  |
| --- | --- |
| Use Case | Play/Resume Audio Tour |
| Actor | User |
| Pre-Condition | The application needs to be set on one of the attraction pages |
| Post-Condition | The application plays the audio for the user |
| Base sequence | 1.User taps on the “play audio” button  2.The system receives the requests and starts playing the audio file |
| Exception Sequence |  |

*Table 6: Play/Resume Audio Tour Use Case Description*

|  |  |
| --- | --- |
| Use Case | Pause Audio Tour |
| Actor | User |
| Pre-Condition | The application needs to be set on one of the attraction pages  The audio needs to be in the “playing” state |
| Post-Condition | The sound is paused |
| Base sequence | 1.User taps on the “stop audio” button  2.The system pauses the audio file, which can be resumed using the play/resume audio function |
| Exception Sequence |  |

*Table 7: Pause Audio Tour Use Case Description*

|  |  |
| --- | --- |
| Use Case | Reset Audio Tour |
| Actor | User |
| Pre-Condition | The application needs to be set on one of the attraction pages  Audio needs to be in either the “paused” or “play/resume” state |
| Post-Condition | The audio file is reset, thus, if played again, it will start from the 00 time mark |
| Base sequence | 1.User taps on the “Reset Audio” button  2.The audio file is reset to the 00 time mark |
| Exception Sequence |  |

*Table 8: Reset Audio Tour Use Case Description*

|  |  |
| --- | --- |
| Use Case | View Map |
| Actor | User |
| Pre-Condition | The application needs to be on the front page layout  The device needs to be connected to the internet |
| Post-Condition | The map is displayed on the screen |
| Base sequence | 1.User taps the “Map” icon  2.System loads in the Map page |
| Exception Sequence | 2A:The application cannot load in the map page because the device is not connected to the internet |

*Table 9: View Map Use Case Description*

|  |  |
| --- | --- |
| Use Case | View History Page |
| Actor | User |
| Pre-Condition | The application needs to be on the front page layout  The device needs to be connected to the internet |
| Post-Condition | The history page is displayed on the screen |
| Base sequence | 1.User taps the “History” icon  2.System loads in the History page |
| Exception Sequence | 2A:The application cannot load in the history page because the device is not connected to the internet |

*Table 10: View History Page Use Case Description*

|  |  |
| --- | --- |
| Use Case | View Training Page |
| Actor | User |
| Pre-Condition | The application needs to be on the front page layout |
| Post-Condition | The training page is displayed on the screen |
| Base sequence | 1.User taps the “Training” icon  2.System loads in the Training page |
| Exception Sequence |  |

*Table 11: View Training Page Use Case Description*

|  |  |
| --- | --- |
| Use Case | View Sustainable Tips |
| Actor | User |
| Pre-Condition | The application needs to be on the front page layout |
| Post-Condition | The Sustainable Tips page is displayed on the screen by the application |
| Base sequence | 1.User taps the “Tips” icon  2.System loads in the Sustainable tips page |
| Exception Sequence |  |

*Table 12: View Sustainable Tips Use Case Description*

|  |  |
| --- | --- |
| Use Case | See Next Tip |
| Actor | User |
| Pre-Condition | The application needs to be set on one of the tips layouts |
| Post-Condition | The application displays the next tip |
| Base sequence | 1.User taps on the “Next tip” arrow  2.The application displays a new tip layout |
| Exception Sequence | 2A: An exception appears when the applications reaches the last tip, and here, when the “next tip” arrow is tapped by the user, he will be sent back to the main page |

*Table 13: See Next Tip Use Case Description*

|  |  |
| --- | --- |
| Use Case | See Previous Tip |
| Actor | User |
| Pre-Condition | The application needs to be set on one of the tips layouts |
| Post-Condition | The application displays the previous tip |
| Base sequence | 1.User taps on the “Previous tip” arrow  2.The application displays the previous tip layout |
| Exception Sequence | 2A. An exception appears when the application is set on the first tip, the back arrow sending the user back to the main tips page |

*Table 14: See Previous Tip Use Case Description*

|  |  |
| --- | --- |
| Use Case | View Quiz Game Page |
| Actor | User |
| Pre-Condition | The application needs to be on the front page layout |
| Post-Condition | Application displays the quiz game page |
| Base sequence | 1.User taps on the “Quiz Game” layout  2.The application displays the Quiz Game page |
| Exception Sequence |  |

*Table 15: View Quiz Game Page Use Case Description*

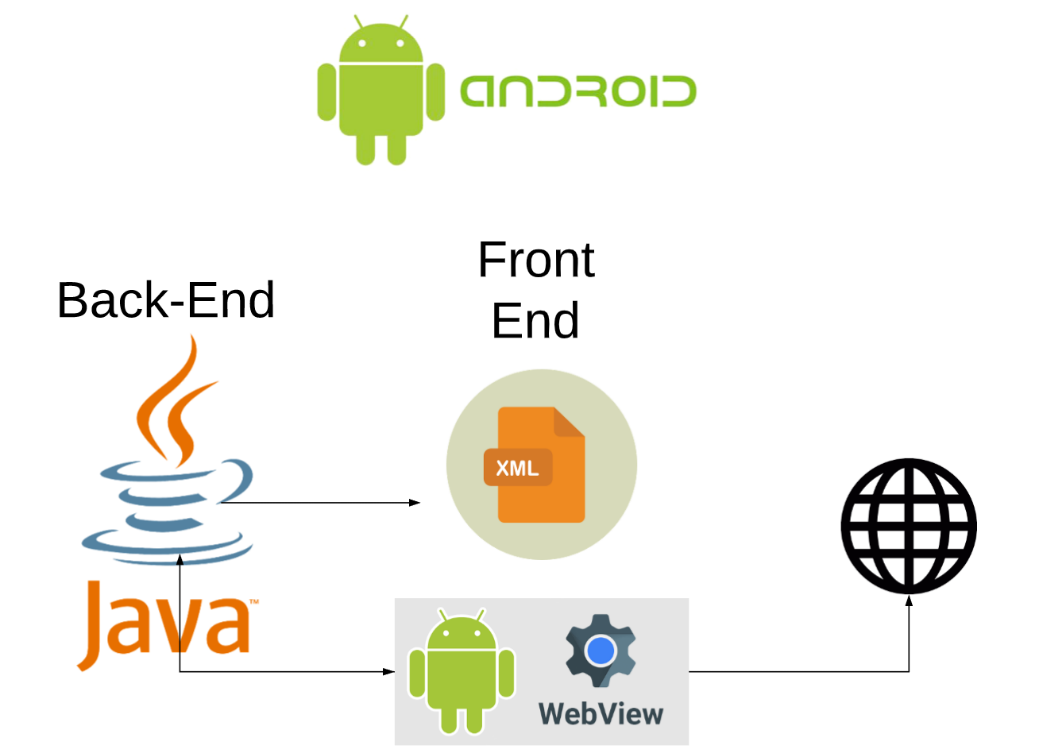
|  |  |
| --- | --- |
| Use Case | View Quiz Game Installation Guide |
| Actor | User |
| Pre-Condition | The application needs to be on the quiz game layout  There is an active internet connection on the device |
| Post-Condition | The installation guide page is displayed |
| Base sequence | 1.User taps on the “installation guide” button  2.Installation guide page is displayed |
| Exception Sequence | 2A. The page cannot be displayed due to the lack of an internet connection |

*Table 16: View Quiz Game Installation Guide Use Case Description*

1. **Design & Implementation**

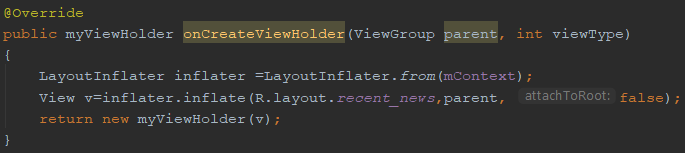
The purpose of this part is to define the architecture, technologies, design patterns and UI choosing of the system.

**4.1 Architecture**

****

*Figure 2: Android Architecture Diagram*

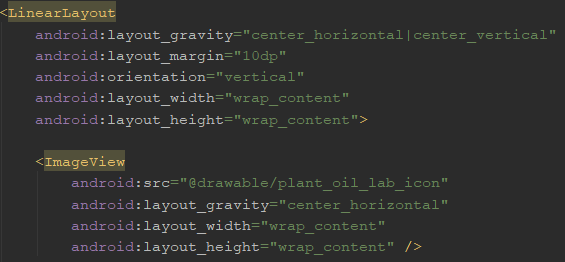
The architecture diagram has the purpose of showing the logic behind the Android application. The application was designed for Android devices, thus, as the big majority of Android applications, the whole back-end of the application was coded using the object-oriented programming language Java.

****

*Figure 3: Java Code Example inside the Application*

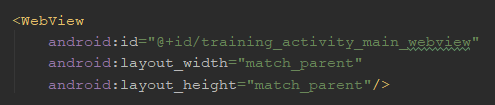
Java was needed to initialize all the layouts, but also to add different back-end logic for the application, such as adding functionality to different buttons and hyperlinks in the application.

As far as the front-end of the application goes, it has been coded using the XML language, which is a markup language (same as HTML).



*Figure 3: XML code example inside the Android Application*

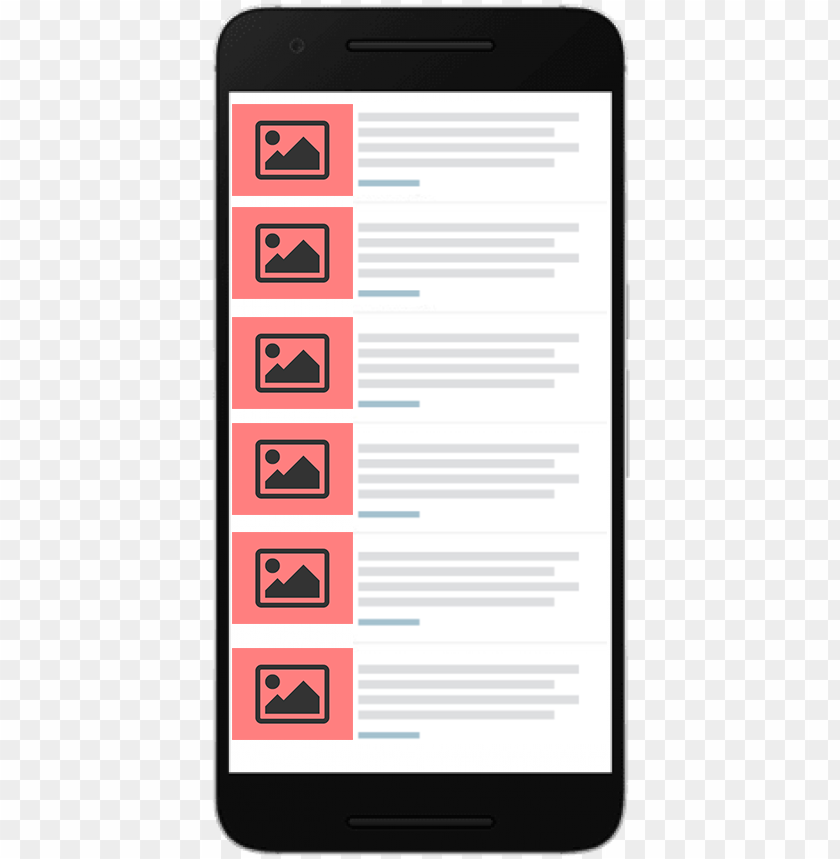
Apart from standard Android layouts, inside the app, WebViews are also used, which allow the Android Application to display a webpage inside the layout itself, which gives the developer some freedom, given that the linked web-page is responsive, but, as a downside, to access such a layout, a constant internet connection is required (the application itself won’t request for constant internet connection of the application contains a WebView).

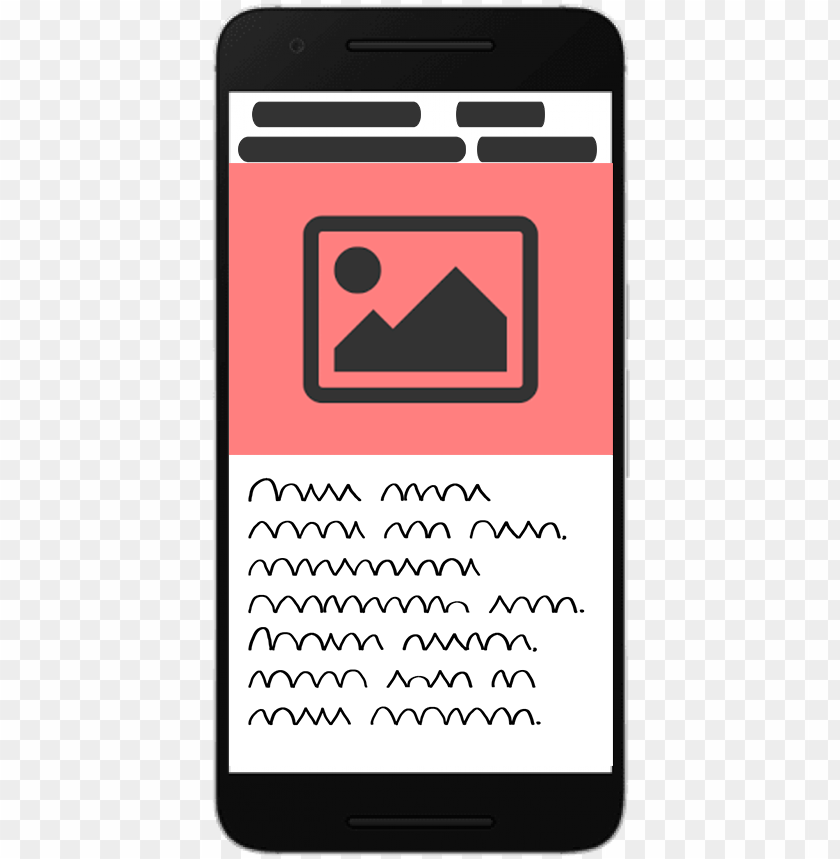


*Figure 4: WebView XML code example*

**4.2 User Interface/Back-end Design and Implementation**

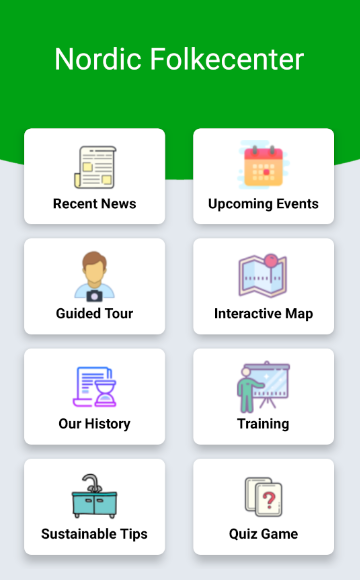
From the early stages of the project, the main focuses regarding the User Interface were to make every layout of every single activity in the Android Application simple, intuitive and eye-catching. Before starting the actual implementation of the application, a few small sketches of the layouts have been created.





*Figure 5/6/7: Small Sketches that were designed in preparation of layout implementation*

To stick with simplicity and ease of use, a clean and eye-catching front-page has been created – all the features of the application can be accessed from a grid layout, with appealing icons. Thus, the icons in the grid layout send the user to the following features: Recent News, Upcoming Events, Guided Tour, Interactive Map, Our History, Training, Sustainable Tips and Quiz Game.

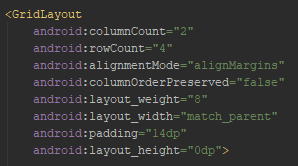


*Figure 8: Front page of the Android Folkecenter Application*

This Grid Layout makes use of Card Views, which, essentially can be thought as a FrameLayout with rounded corners and shadow based on the elevation of elements. CardView wraps a layout and is often the container used in a layout for each item within a ListView or a RecyclerView.

The main element of the front page is a grid layout, containing eight CardView elements/children. The GridLayout is a layout that places its children in a rectangular grid, which is composed of a set of infinitely thin lines that separate the viewing area into cells. Throughout the API, grid lines are referenced by grid indices. A grid with N columns has N + 1 grid indices that run from 0 through N inclusive. Regardless of how GridLayout is configured, grid index 0 is fixed to the leading edge of the container and grid index N is fixed to its trailing edge (after padding is taken into account).

Children occupy one or more contiguous cells, as defined by their GridLayout.LayoutParams#rowSpec and GridLayout.LayoutParams#columnSpec layout parameters. Each spec defines the set of rows or columns that are to be occupied; and how children should be aligned within the resulting group of cells. Although cells do not normally overlap in a GridLayout, GridLayout does not prevent children being defined to occupy the same cell or group of cells. In this case however, there is no guarantee that children will not themselves overlap after the layout operation completes.



*Figure 9: Grid Layout XML code example*

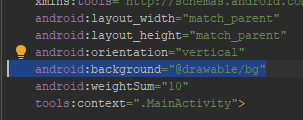
CardView provides a default elevation and corner radius so that the cards have a consistent appearance across the platforms. However, these values can be customized, a background color for the card can also be set.



*Figure 10: Card View XML code example*

The above code is an example of the implementation of a CardView inside the Folkecenter Mobile Application – proprieties such as Elevation, Corner Radius, margins etc. for the CardView are defined in the element itself, and then, the CardView contains an ImageView, which sets the icon and defines it’s proprieties, and a TextView – a small caption for the icon, these two views being wrapped in a Linear Layout. This implementation repeats 8 times, since the GridLayout is composed of eight CardView elements/children.

The main design of the front-page has been recycled for other layouts in the application. The green arch on the top side of the layout is not generated by any XML code – a background image is set, which is a grey wallpaper containing the green arch on the top side.



*Figure 11: Setting the background of an XML page*

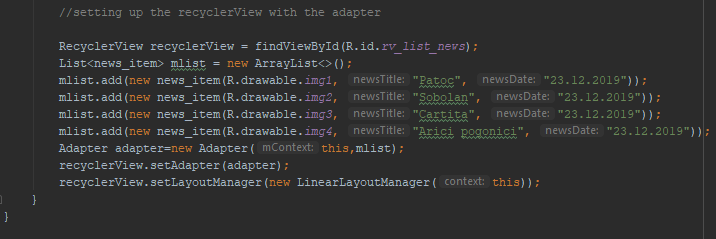
*Recent news*

The recent news section makes use of a RecyclerView. A RecyclerView is a more advanced version of ListView with improved performance and other benefits. The way a RecyclerView works is, by creating a template (framework) element, and, whenever a new element needs to be created, data is provided to the RecyclerView, and, based on the template, the new element is created.

There are three major components for any RecyclerView:

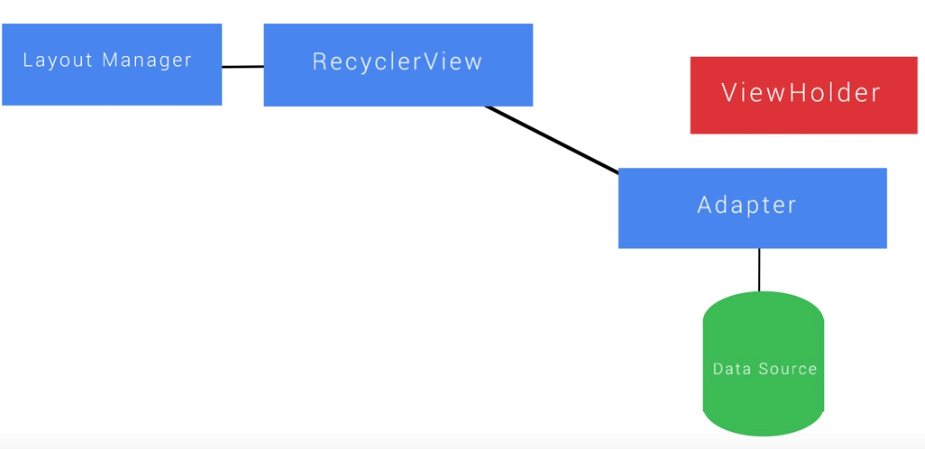
**Recycler.LayoutManager**- In the RecyclerView model, several different components work together to display your data. The overall container for your user interface is a RecyclerView object that you add to your layout. The RecyclerView fills itself with views provided by a layout manager that you provide. You can use one of our standard layout managers (such as LinearLayoutManager or GridLayoutManager), or implement your own.

**RecyclerView.ViewHolder** -The views in the list are represented by view holder objects. These objects are instances of a class you define by extending RecyclerView.ViewHolder. Each view holder is in charge of displaying a single item with a view. For example, if your list shows music collection, each view holder might represent a single album. The RecyclerView creates only as many view holders as are needed to display the on-screen portion of the dynamic content, plus a few extra. As the user scrolls through the list, the RecyclerView takes the off-screen views and rebinds them to the data which is scrolling onto the screen.



*Figure 12: RecyclerView Java Implementation*

**RecyclerView.Adapter -** The view holder objects are managed by an adapter, which you create by extending RecyclerView.Adapter. The adapter creates view holders as needed. The adapter also binds the view holders to their data. It does this by assigning the view holder to a position, and calling the adapter's onBindViewHolder() method. That method uses the view holder's position to determine what the contents should be, based on its list position.



*Figure 13: RecyclerView functional diagram*

*Upcoming Events, Interactive Map & Our History*

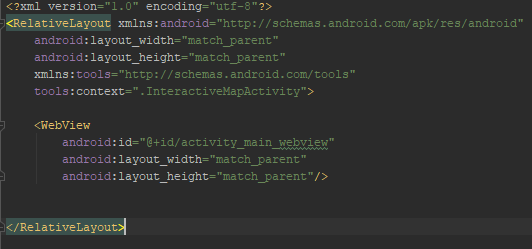
Inside the application, so simplify some implementation, and take advantage of already existing implementations, some of the layouts are implemented using WebViews. In Android, a WebView is a way to link an external website inside the application. The only disadvantage when using this technique is that, to access such a layout, the device needs a constant internet connection, otherwise the app is going to crash or display an error message.

The WebView in Android enables you to show the web browser in your Activity. It is very useful if our application needs to embed some web content, like maps from some other providers and so on.

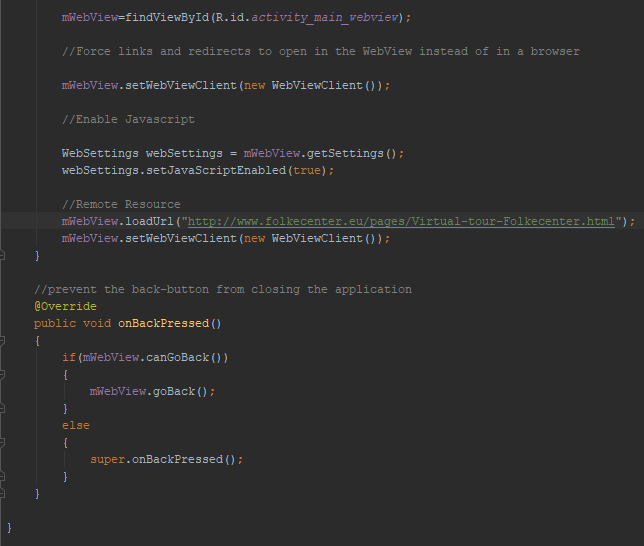
Here, the web page can be loaded inside the application with the help of URL. Mainly it is used to show the internet content inside the activity.

WebView in Android turns the application into a web application. It comes from android.webkit.WebView. Here, the WebView class is an extension of Android’s View class which is used to show the web pages.

WebView doesn’t include all the features of Web-browser-like navigation controls or an address bar etc. But we can navigate back or forward inside WebView. Because it automatically accumulates the history of web pages. So we can navigate backward and forward through the history with the help of goBack() and goForward() methods by using device back button in our activity.



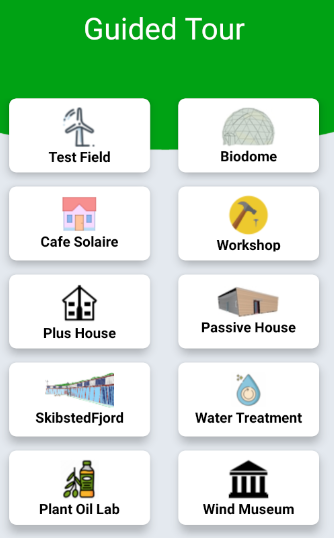
*Figure 14: WebView XML implementation for Interactive Map*



*Figure 15: WebView Java implementation for Interactive Map*

*Guided Tour*

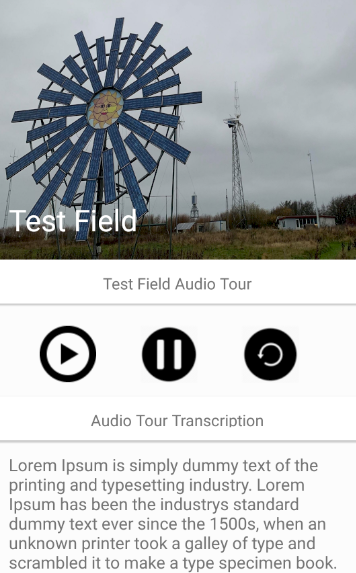
The guided tour feature is the most complex feature of the Folkecenter Mobile Application, as, during its development, the requirements for this feature expanded – the supervisor decided that he wants this feature to also be available in Danish and German, not only English, so a few changes had to be done. The guided tour page features a “front-page”, which uses the same template as the main layout of the application, a grid layout with card views and image icons – from here, the user is able to choose a location from Folkecenter that he is interested to learn about.



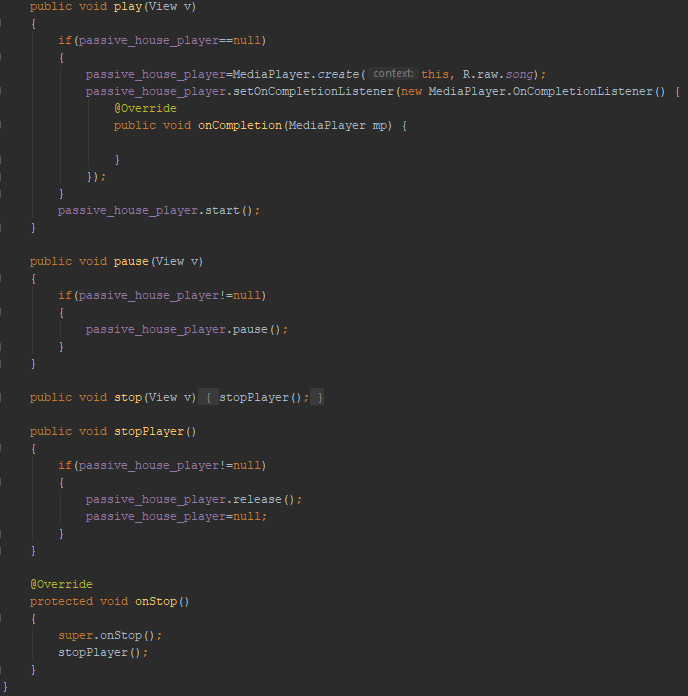
*Figure 16: Guided Tour Layout*

Once the user picks one of the locations, a new layout will open, which features a CollapsingToolbarLayout, a NestedScrollView, a written version of the tour, and also an audio version of the guided tour, where, with the help of three buttons, the user can play an audio file containing information about the chosen attraction, he can pause the audio, or reset it, so that it plays again from the beginning.

During the development of the feature, the application supervisor decided he wants this feature to be available in multiple languages – to implement this, a new layout has been created, where, by pressing one of the three language buttons, the user is able to select his preferred language, which will redirect him to the proper guided tour page with the updated text and audio files. The AudioPlayer makes use of the MediaPlayer library, using implementations of the play, pause, stop and onStop methods.

****

*Figure 17/18: Select Language Layout/Test Field Guided Tour Layout*

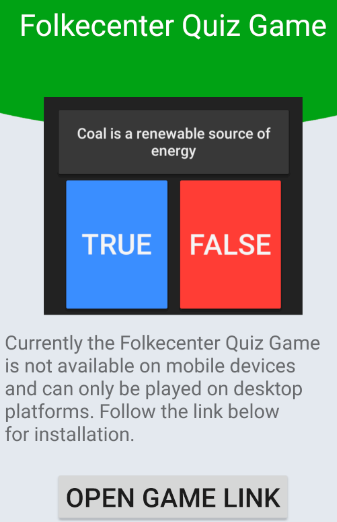
****

*Figure 19: Audio Player Java Implementation*

*Training & Quiz Game*

The Training page is very similar to the Quiz Game page – they both feature simple Android views – a title, an image, some descriptive text along with a button which opens an external page. The button on the Training page is going to open a section on the Folkecenter website, which offers people information about the trainee program at Nordisk Folkecenter, along with a guide on how to sign up for to be a trainee.

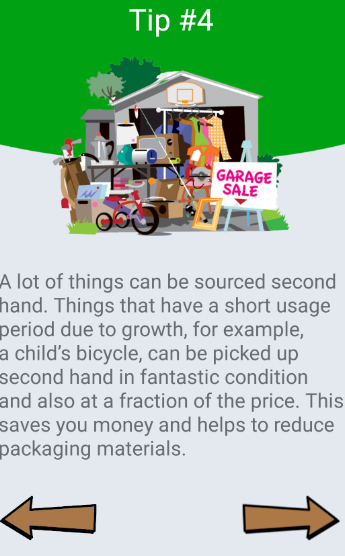
The Quiz Game page sends the user to a GitHub page, with a guide on how to install the Folkecenter Quiz Game on a desktop device (Windows, Mac or Linux), as, during the development of the Folkecenter Mobile Application, the Quiz Game didn’t make its way yet on Mobile Platforms.



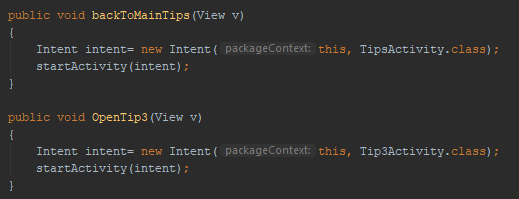
*Figure 20/21: Training Page/Quiz Game Page*

*Sustainable Tips*

The Sustainable Tips feature contains multiple pages with tips regarding on how to save energy in money in the everyday life. After tapping the icon, the user is sent to a page similar to the ones described above – the quiz game and training pages. The only difference is that the button won’t send the user to an external website, they will send him to a layout containing the first tip. The navigation between the tips is made using arrow, for each of them an intent being created, which allows the user to skip to the next tip, or go the the previous one. The “next” arrow on the last tip is sending the user to the front page of the Android Application. The user can also use the “back” button of the Android device in order to navigate to a previous tip.

****

*Figure 22: Tip Layout Example*

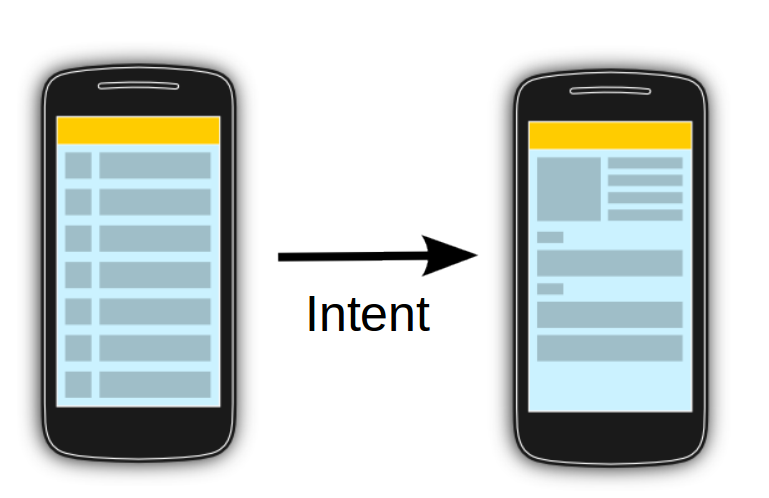
****

*Figure 23: Redirect Next/Previous Tip Java Implementation*

*Intents*

Android application components can connect to other Android applications. This connection is based on a task description represented by an Intent object.

Intents are asynchronous messages which allow application components to request functionality from other Android components. Intents allow you to interact with components from the same applications as well as with components contributed by other applications. For example, an activity can start an external activity for taking a picture.



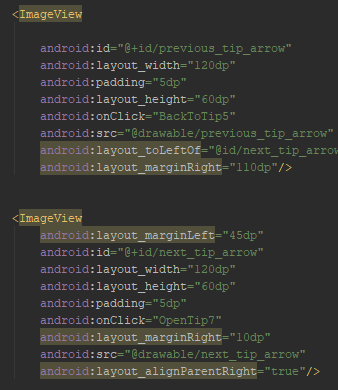
*Figure 24: Android Intent Diagram*

There are two types of intents:

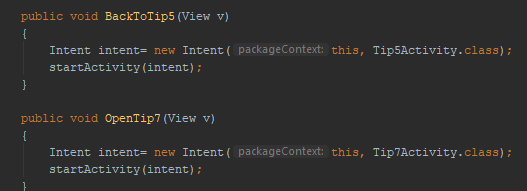
**Explicit intents** specify which application will satisfy the intent, by supplying either the target app's package name or a fully-qualified component class name. You'll typically use an explicit intent to start a component in your own app, because you know the class name of the activity or service you want to start. For example, you might start a new activity within your app in response to a user action, or start a service to download a file in the background.

**Implicit intents** do not name a specific component, but instead declare a general action to perform, which allows a component from another app to handle it. For example, if you want to show the user a location on a map, you can use an implicit intent to request that another capable app show a specified location on a map.

An intent can be triggered by multiple types of views – in the Folkecenter Mobile Application, to trigger intens, buttons, card views and images are being used.

****

*Figure 25: Redirect Next/Previous Tip XML Implementation*

****

*Figure 26: Redirect Next/Previous Tip Java Implementation*

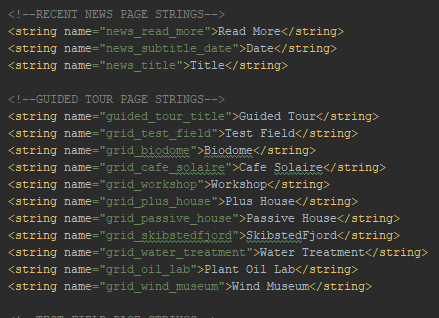
*External Resources*

In order to optimize the application, resources such as images, strings and audio files are externalized.

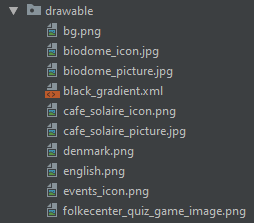
Resources are the additional files and static content that your code uses, such as bitmaps, layout definitions, user interface strings, animation instructions, and more.

You should always externalize app resources such as images and strings from your code, so that you can maintain them independently. You should also provide alternative resources for specific device configurations, by grouping them in specially-named resource directories. At runtime, Android uses the appropriate resource based on the current configuration. For example, you might want to provide a different UI layout depending on the screen size or different strings depending on the language setting.

Once you externalize your app resources, you can access them using resource IDs that are generated in your project's R class. This document shows you how to group your resources in your Android project and provide alternative resources for specific device configurations, and then access them from your app code or other XML files.

****

*Figure 27: External Strings Example*

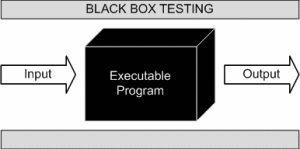
****

*Figure 28: Drawable Resources Example*

1. **Testing**

*Black Box Testing*

Black Box Testing, also known as Behavioral Testing, is a software testing method in which the internal structure/design/implementation of the item being tested is not known to the tester. These tests can be functional or non-functional, though usually functional.



*Figure 29: Black Box Testing Diagram*

The tests check if the android application works correctly from the user perspective. Does each function and feature do what it should do/what it promises and is the User Interface easy to understand?

The main purpose of the testing is also to validate if the main use cases that were defined have been implemented and work as intended.

|  |  |  |
| --- | --- | --- |
| Description | Expected Result | Result |
| The user wants to run the application on his Android device without any crashes on start-up | The application is started and the user can see the front-page, without any crashes or errors in the process | Success, works as expected |
| The user runs the Events page without being connected to the internet | The application displays an error message and sends the user back to the front-page | Success, works as expected |
| The user runs the Events page while being connected to the internet | The application displays the proper section from the Folkecenter Website | Success, works as expected |
| The user can access the Guided Tour feature and select the preferred language | The proper guided tour front-page is displayed, depending on the selected language | Success, works as expected |
| The user taps on the “play audio” button in one of the guided tour pages | The assigned audio file starts playing | Success, works as expected |
| The user taps on the “pause audio” button in one of the guided tour pages | The audio file is paused | Success, works as expected |
| The user taps on the “reset audio” button in one of the guided tour pages and then plays the file again | The audio file is restarted and, if played again, starts from the 0:00 time mark | Success, works as expected |
| The user wants to scroll through the sustainable tips using the assigned arrows | The arrows send the user to the previous tip or to the next tip, depending on which one was tapped | Success, works as expected |

*Table 17: Test Cases Results*

1. **Results, Discussion, Conclusion and Project Future**

The Folkecenter Android Application was developed in order to give the visitors at the center a better experience, and also a better understanding of the history of Folkecenter and the activities that are happening here. The step-by-step progress of the entire project, including documentation can be followed on the following GitHub link (the development process followed the version control protocols) - <https://github.com/IonutPutinica/FolkecenterMobileApp> .

Some features have been left umimplemented, thus paving the way for future development and possible new features – some of the layouts of the android application are not fully-responsive yet, a PDF reader needs to be implemented inside the application so that PDF files can be opened inside WebViews, a notification system should be implemented.

The development of the Android application is a big step forward for Folkecenter, allowing the center to promote itself easier, especially to young audiences.

**Sources of Information**

<https://www.hyperlinkinfosystem.com/blog/the-importance-of-mobile-applications-in-everyday-life>

<https://android.jlelse.eu/understanding-recyclerview-components-part-2-1fd43001a98f>

<https://acadgild.com/blog/what-is-webview-in-android-a-tutorial>

<https://www.vogella.com/tutorials/AndroidIntent/article.html>

<https://developer.android.com/guide/topics/resources/providing-resources>