EventItNow Facebook event organiser - app showcase

EventItNow is an app for organizing events, parties, meetups etc. through Facebook Events. This article describes the main elements of the app design.

Introduction

This is the description article of **EventItNow**, my entry to the Symbian Qt Quick Components Competition 2012Q1. **EventItNow** has to do with easing the organization of an event, any kind of event, through Facebook Events. The idea was created last summer when trying to organize a beach party and we messed it up. Any way, a few months later it is real thanks to Qt and QtQuick. Most of the QtQuick Components for Symbian were vital and without them the visual and functional outcome wouldn't be the same.

Facebook

To begin with, since the target of the application is the creation of a Facebook Event, it needs a Facebook user to login. So the user, the first time that enters the **EventItNow** is prompted to login to Facebook. It is important here to note that **no password is saved anywhere in the phone**. The login operation is held through a Facebook's page and by accepting, the user grands the application to be able to create events etc. From this step and on, the user will never be asked again to do the above procedure. All Facebook communication is held via HTTP requests complying to the Facebook's Graph API through Javascript. Moreover the login page to Facebook is shown through a WebView. The only thing that remains malfunctional is the success of the DELETE HTTP request. I even created a QNetworkAccessManager on the C++ side of the project but that didn't work as well.

ArcList

When i was thinking of designing this application for my mobile i kept having the thought that the common lists were to....flat. May be their real estate of the screen is the best for the mobiles but still the don't look that pretty when it comes to having them as the main screen of the project.That's the reason why i created ArcList which is the only custom component used that doesn't fully comply with the Symbian Look&Feel. Briefly, the components that are to be placed to the list are placed to the perimeter of an Ellipse. So the position of an List Item is represented by one number, the angle on which the Item was placed on the Ellipse.

Lets say that by seeing our mobile phone, every thing on screen is placed on 2D coordinates (well not everything but you get the point), X and Y axis. So z axis is vertical to the surface of our mobile screen. The ArcList is positioned to either X, Z axis (when in landscape orientation) or Y, Z axis (when in portrait orientation). The closest an object can get to the screen is at the point of the Ellipse closest to the screen which is the center of the screen. As it goes further up or down, a scaling rate is set to make them look smaller and smaller giving the feeling of depth. Moreover, at all times only half of the Ellipse is visible or else we would be able to see small items behind the topmost Items moving in opposite direction. This is done by applying opacity changes when an Item gets closer to the middle angle of the Ellipse. That is why it is called ArcList and not EllipseList. Because at all times, only the half is visible, thus an Arc.

So opacity combined with scaling gives the feeling of depth. What about moving the items to bring forward / backward the Item you want? As mentioned, each Item is placed to a certain angle of the Ellipse. This can be translated into keeping only one angle variable and place & bind all the items using offsets to this variable. Thus changing the value to the angle variable, all the Items are moved at the same time keeping their distances.

Scrolling

Next step was to add Kinetic Scrolling to the ArcList. Unfortunately, ArcList couldn't get inside a Flickable Item because they both are interested in moving on different axes. The solution was to create a "proxy" Item that would receive the scrolling of the Flickable Item and would then pass it to the Arclist. The "proxy" is an opaque *Rectangle* placed under the ArcList which is inside a Flicable item. It's size is set in that way that it can correctly provide values ranging the scrolling of the ArcList.

And that is how the main menu was created.

Operation Views

At application startup the user is able to Create a new Event, Open an already existing or open the Settings. *Create a new Event* view uses a TextField for defining the name of the event, a TextArea for setting further details (they need more room that is why the TextArea is preferred) and finally Date and Time Pickers. The Date and Time Pickers are placed into dialogs shown to specify the start and end time of the event. Everything related with dates and time is wrapped into a rectangle with white border when the date/time is valid, blue border on mouse pressed and into red borders when date/time is invalid. Invalid dates/time can occur in 2 circumstances. First if the date/time set points to the past compared to the present and second if end date is previous to the start date. So the user receives feedback for the validity through the colored rectangles plus a warning dialog raised if user tries to save the invalid date/time.

The Event picker uses a ListView to show the events available. In the list there are 3 section headers to differentiate between events **In Progress**, **Finished** and **Canceled**. As reported in the documentation of the ListView, the model list needed previous sorting to keep separated the different categories or else there would be a mixup.

Finally one InfoBanner is used in the starting view to show the current active user.

When entering in the main menu of an already created Event you are able to use 5 tools. Location, Shopping List, To-Do List, Guest List and Settings.

(C) Copyright Nokia 2013. All rights reserved.

Shopping List & To-Do List

These two pretty much follow the same way of thinking. In the main view there is a common ListView whose items have a checkbox, a text and a subtitle. The checkbox is to show that the item is either bought (Shopping List) or completed (To-Do List). The text has the title of the entity and the subtitle has some secondary info. The subtitle was placed for the user to be able to get more information about the items in the main list without having to get in the details page of each item. Finally in Shopping List, there is a List Header anchored to the top. Although this fact results to permanently hiding a portion of the visible screen, it is necessary in my opinion because in there is constantly shown the Total & Paid amount of money which is the purpose of the Shopping List after all.

Location

The Location view is a Map element from Qt Mobility 1.2. It is used to set the location of the event and additional details about it. When the location gets defined, a Google Maps link pointing to that place is set as location property to the Facebook Event. For the location marker was used a simple image which is placed in the map accordingly and is set by double-clicking somewhere on the map. I tried to use the MapImage but i had some problems that i don't still remember so rolled back to the simple image and translation of screen coordinates to map coordinates. Probably MapImage would be the best so count one for the future. So, when a location marker is set, automatically opens a details dialog where the user can set additional information about the location. Those details are visible in the end of the Facebook Event details property with the heading **Location Details**. On long press of the location marker a BusyIndicator is used to suggest that an operation can take place on long press. A second later the location marker is removed. Finally, the zoom Buttons and Bar are Rectangles with opacity.

Guest List

The main view of the Guest List is a ListView through which user can select a guest and see his/hers details page. Further deeper in the PageStack layers, lies the Facebook Friend list. Also a ListView but provides **Adaptive Search**. That is, when searching is enabled, buttons show all the unique item's first letters of the ListView. Every selection (button press) narrows down the results, ending up with fewer items and eventually the one the user been searching for. It is known that this version of **AdaptiveSearch** i have, has a few limitations but they will be addressed in the near future. Limitations such as the case where there are too many unique letter where a new "page" of buttons must be created.

General Considerations

Progress Bars & BusyIndicators

All Facebook operations have to do with network access, thus delays. To keep the user informed that something is taking place ProgressBar and BusyIndicator are placed all over the place. Hypothetically, many of the delaying operations can be masked, covered or call it what ever you like, by not waiting for it to complete and moving to the next page the user wants to see. Here there is a catch. If something goes wrong the user will remain with the idea that it was successfully done. There is no other indication for the user other than reviewing old actions to see if they were indeed successful, which is not always feasible. In my case if data needed to change to the Facebook event the user must always know that everything went fine or else he/she needs to check his/hers Facebook account from an other source to verify. Not good. This is prevented at some point by adding a tiny bit of delay to the application and the UI flow but the user is more informed about the state of his requests.

Usability

At the first time someone uses the application nothing is set. So the application itself guides the user. By the time user opens the application, any operation done will end up logging into Facebook thus been the active user. Later on, many views of the application have ListViews. At first everything is empty so at that point, apart from the normal way of initiating the creation of an object, every empty list has a Dotted Rectangle on top. Catches the eye of the user making it more usable and predictable.

Application Window

At first i kept using ApplcaitonWindow because it was way more convenient compared to the simple Window that you needed to implement everything. But ApplcaitonWindow's warning...warned me and changed into the simple Window along with manually adding the so useful PageStack, ToolBar, and StatusBar

Storage

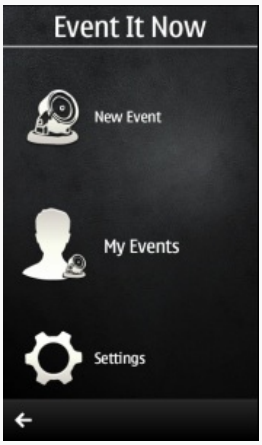
The MySql backend of the Offiline Storage was the perfect way to store and retrieve all the different data i needed. Guests, Shopping baskets, Todo-lists, User details, Event details...very usefull.

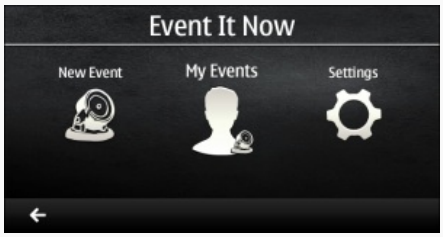
Screenshots & videos

So enough with the talking. Follows screenshots of the application in Portrait mode and two videos showing both Portrait and Landscape mode adaptation.

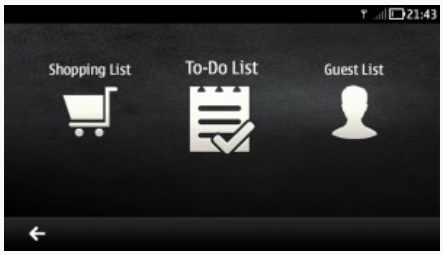
Screenshots

 (C) Copyright Nokia 2013. All rights reserved.

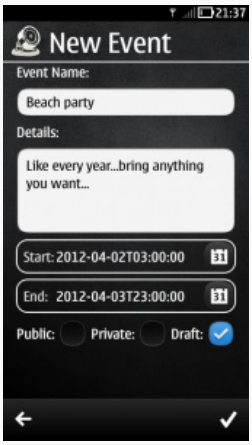


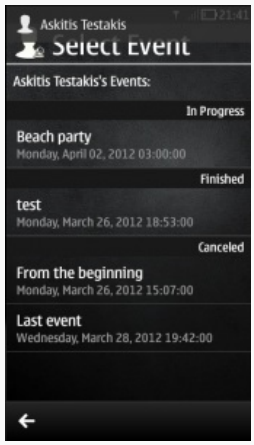


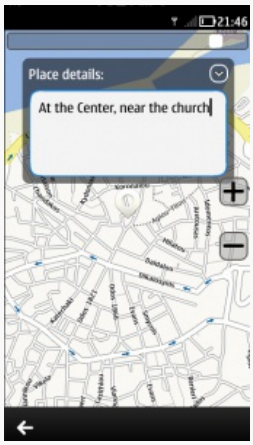
Start Screen (Portrait - Landscape)



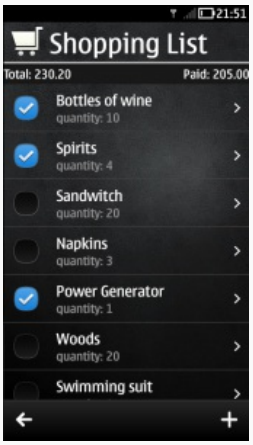
Main menu (Portrait - Landscape)

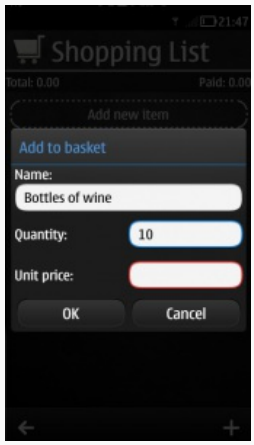


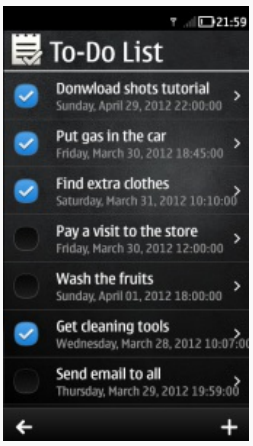


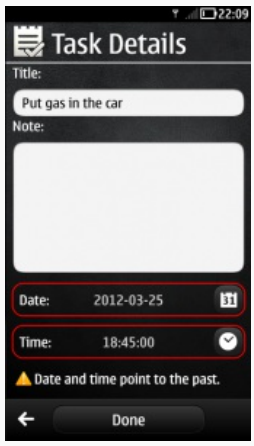


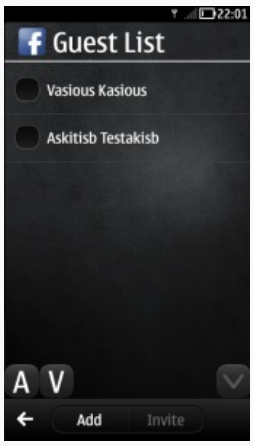
(C) Copyright Nokia 2013. All rights reserved.

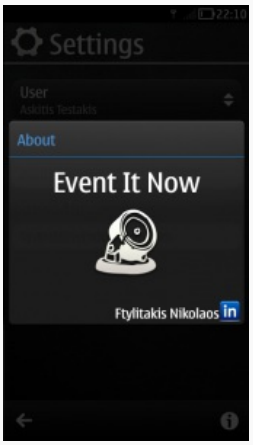






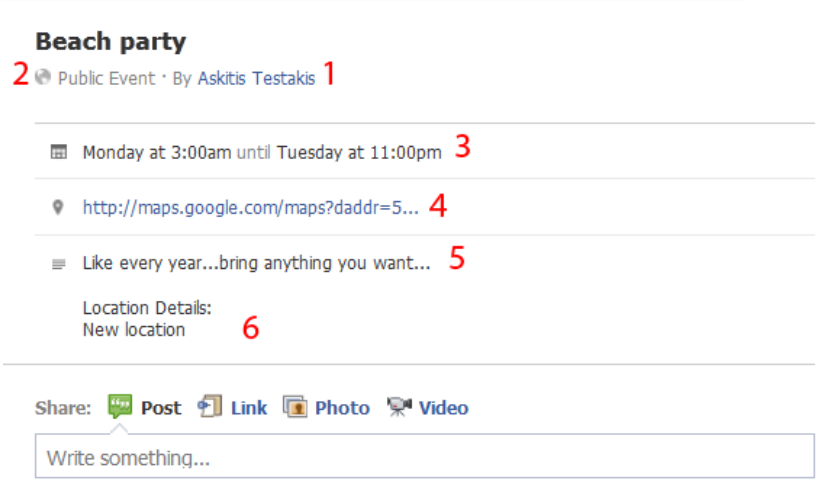






Facebook Screenshot

(C) Copyright Nokia 2013. All rights reserved.



This is how the event looks like when accessed from the Facebook itself. Follows the explanation of each field:

1. The name of the user. Set according to the user that gets logged in, in EventItNow.
2. The privacy status. Is set either when creating the event or from the Settings view in the Event menu.
3. The start and end date of the event. Same as privacy status
4. The Google Maps link created by setting a location marker in the Location View
5. The event details set at event creations of from the Settings view
6. The location details set from the details dialog in Location View.

Videos

The media player is loading...

The media player is loading...

Download

For the latest version visit Event It Now 's Nokia Store page!

To download the binary of EventItNow version 1.1 presented here press File:EventItNow installer.sis.

Note: This is an entry in the Symbian Qt Quick Components Competition 2012Q1

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.