# EventHandler – the social future of summer schools at VIA!

## Purpose of the Application

The general purpose of the EventHandler is to provide a social platform, where students attending the summer school at VIA can share (i.e. create). Thus the most vital part of the application is the ability to create a new event, display the events in an organised list and click on an event object that will open a new activity presenting the chosen event.

Another important feature of the application, which is not yet implemented, is the ability to attend a given event. This would allow the users to review the attendance of the different events in the list. For this reason there are integrated a login- and create account function. Thus the users have a unique identification that will make the “attend-to-event”-functionality more personal, and also wipe out the risk of a user attending the same event multiple times.

Additionally, while reviewing the list of events, the users are able to perform swipe-down-touch on the screen that reloads the activity and calls the Web Service for any newly added events. Finally, when a user is already logged in and reopens the application after it has been closed, it redirects the user to the activity in top of the stack.

Beside the “attend-to-event”-functionality, it is also important to enable the users to delete or edit their uploaded events, as invalid information and events hence are avoided.

Furthermore, in future versions, functionality allowing additional information about the events, as for instance adding a location by Google maps, should be implemented. Also a filter, allowing the users to filtrate their search for an event, could be added to the application.

## Early mock-ups

In order to visualize how the final product should be carried out, some mock-ups have been drawn (using Visio). Each mock-up represents a specific activity, and thus the UI of the application. Through this process and discussion the overall framework was clarified.

## Implementation of the application

### Frontend

For polishing the UI, various design features has been added to the objects in the activities presented to the user. In the following, a few examples will be highlighted and evaluated.

#### activity\_create\_event.xml: focus on a user-friendly design

When the user is about to perform an action, several descriptions and hints are given to ease the process of performing actions within the application. For instance, in the activity for creating a new event, there are displayed hints revealing what information to write in the input fields. This is illustrated in the figure 1. Furthermore the supporting xml is illustrated in figure 2.

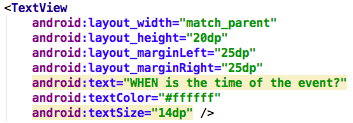
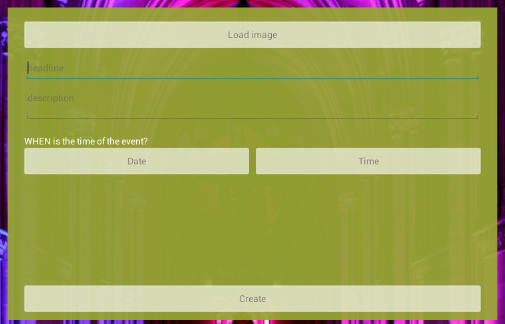


Figure 1: form field to create an event

Figure 2:XML Layout of form field

Moreover, as shown on the displayed activity above, the buttons also contain such information that ensures a frictionless user-interaction. The buttons inform the user what information is to be supplied by clicking the concerning objects. By clicking the “Date” or “Time” -button, the user will be introduced to a fragment containing either a TimePicker- or DatePicker object respectively, examples are shown on figure 3.

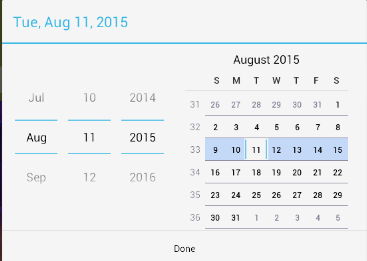
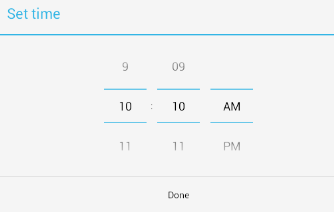


Figure 3:DatePicker to choose an event date an time

These types of objects avoid misperceptions among users about what date/time format to input. Furthermore the presented fragment serves to ensure the user that the “create event”-action is still being performed.

When the user has selected the date and time for the event being created, the chosen data will be presented as text on the buttons, confirming the user that the chosen data will be attached to the event uploaded. This is illustrated in figure 5, and to the figure 4 is the supporting java code.

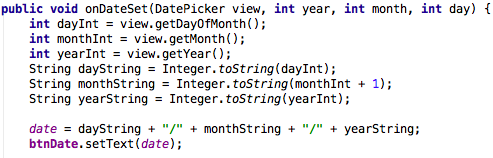


Figure 4: DatePicker and TimePicher buttons Figure 5: Java code to get input data

As it appears from the java code the first part of the method gets the integer information from the DatePicker object, and following this information is casted to a string. This allows the data to be presented in a common date format (i.e. splitting the integers by “/”), by storing all the information in a single string variable, which is more convenient when storing this information in the final event object to be added to the ArrayList in “CreateEventActivity”. Finally the setText-method is called to alter the text on the button for the above-mentioned purposes.

To provide the event better user experiences it is added a FloatingActionButton from “melnykov” library. This button gives possibility to be redirected to CreateEvent Activity to create an event from anywhere on the feed by just a click.

Furthermore there is created a custom layout for the Actionbar as XML. The reason is obviously to give it a custom style. Therefor the already existed style.xml file is used to define the style of the Actionbar and the other elements as resources. That’s give the benefit of one time creation and several time using.

Figure 6 shows the final presentation of the Feed.

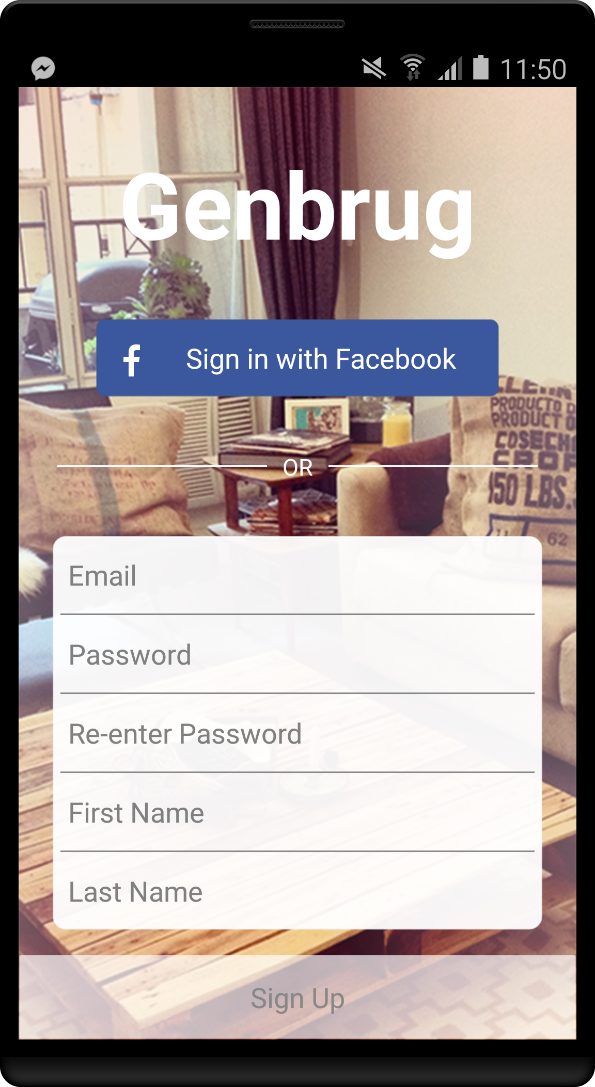
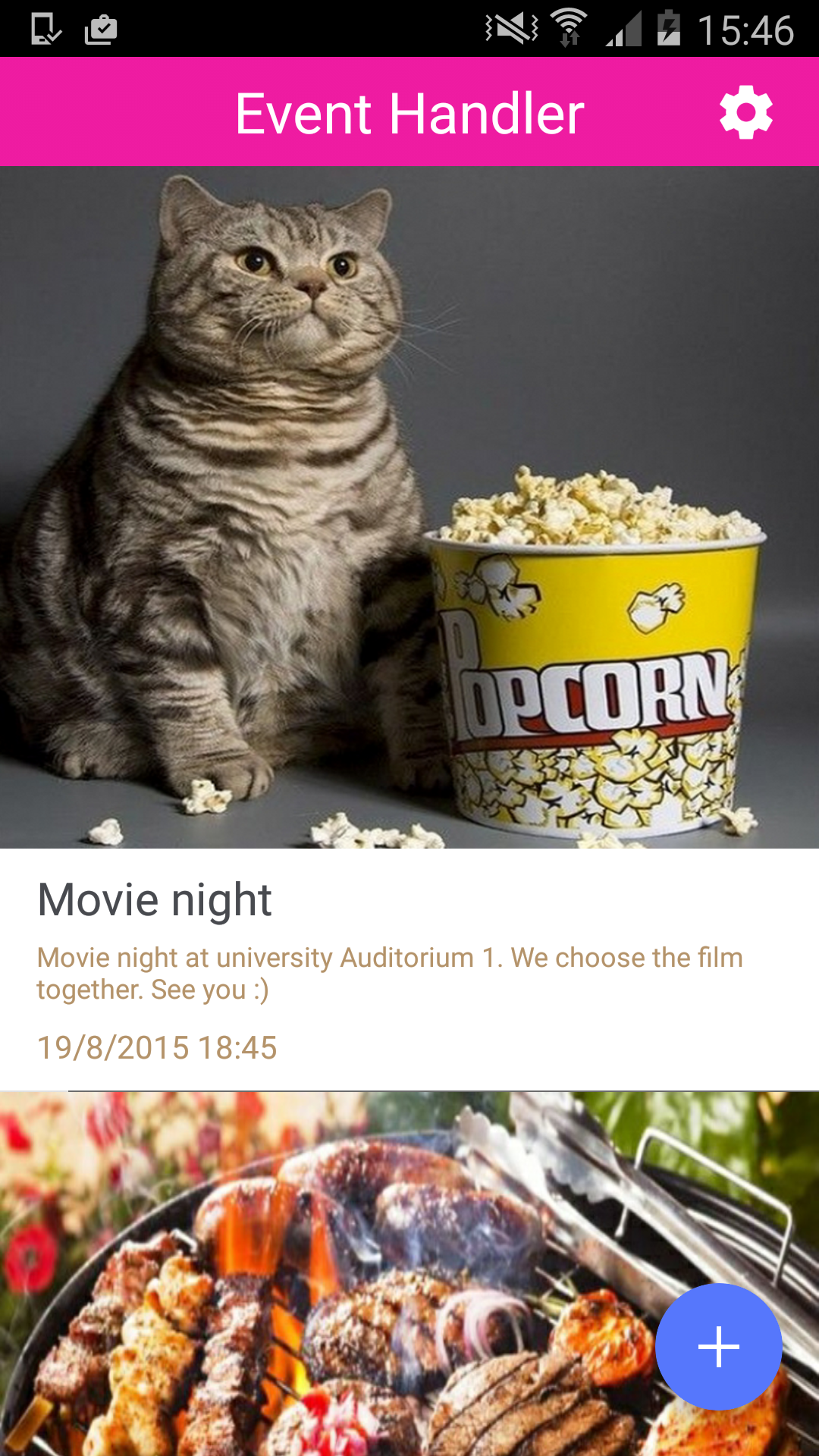


Figure 6: Feed event presentation

### Backend

For persistence of the data, which contains User objects and event objects it is used MSSQL database on a server. To “connect” to data it is implemented a JSON webservice in ASP.NET. We are with a Handler.ashx able to auto-generate methods to access the database data with JSON objects in Android side.

We started to use a service class to handle calls to webservice by using Threads. However, we got an issue to “control” the process of the thread execution and therefor problem to update the UI as expected.

For this purpose, we used classes which extends AsyncTask which gives us few methods where we can “easily” follow the process of Tasks execution. In the beginning were those classes part of our Activities which gives very unclean code. For this purpose, it is created “independent” classes using an Interface. The interface with generic type parameter is used to give us more clean code. It has one method, which is implemented, for the functionality we are expecting from AsyncTask class. This method is called in “onPostExecute” where we expect a result for our Task. We finally call this Task in our Activity for instance in an “onClicklistener” event. In those cases that we get a JSON object back from our JSON webservice we need to “parse” it to a type we can use. For instance for login we wish a Boolean to check if there is an exciting valid user.

The Class Diagram in figure 7 illustrates the association between classes and the interface.

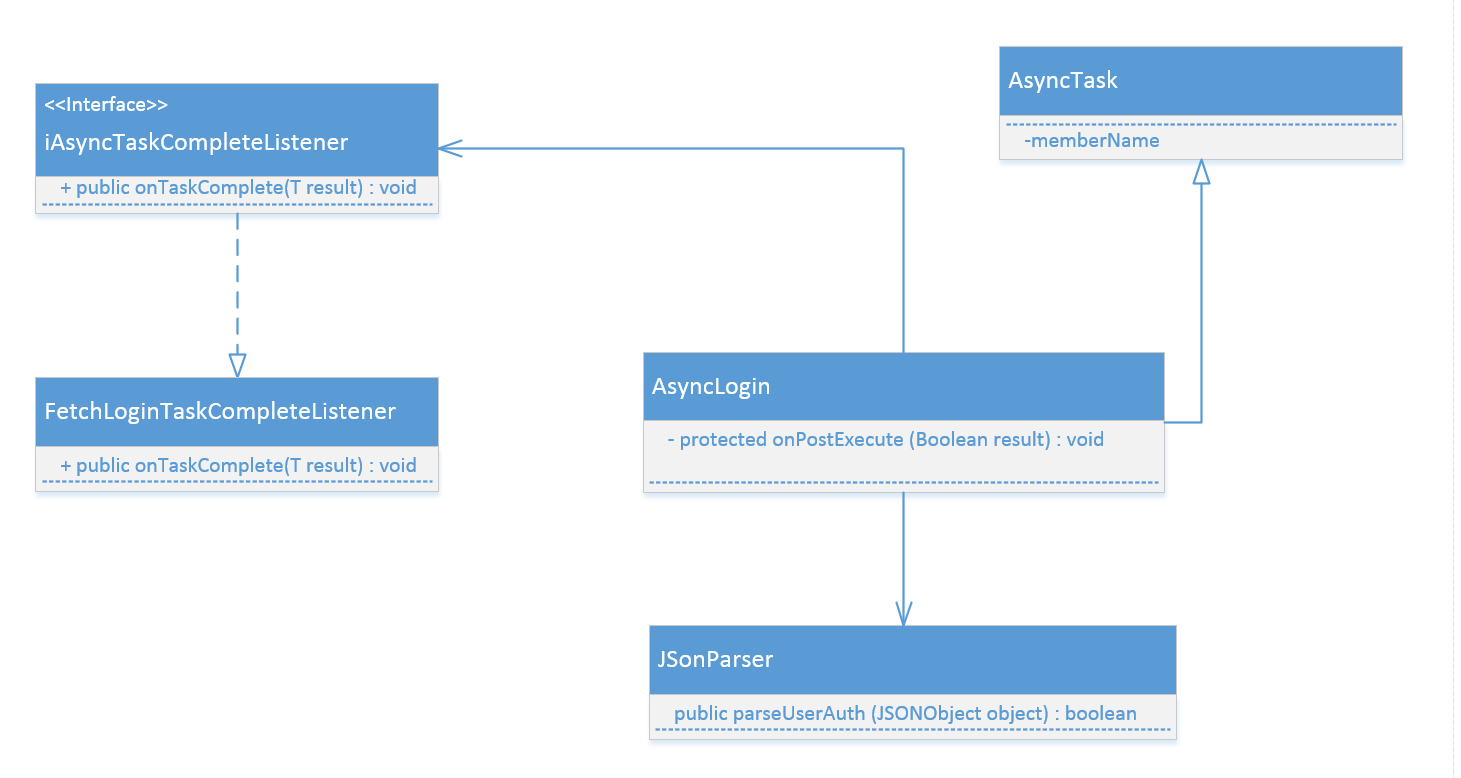


Figure 7: Classdiagram for Login functionalitet

To show created events on the Feed activity it is used a ListView with a custom view. The custom view is created as a Layout XML to show the provided data in specific way. To make the presentation possible it is used an Adapter.

The Adapter is the bridge between the ListView and the data that backs the list from our webservice called in ourActivity. In the Adapter we use the “getView” method to inflate our custom view and have access to the elements inside and fill them with data. The Data is passed from the FeedActivity, to FeedAdapter as a parameter in Constructor.

The diagram on figure 8 shows the interaction between the mentioned elements to provide data in the Custom Layout.

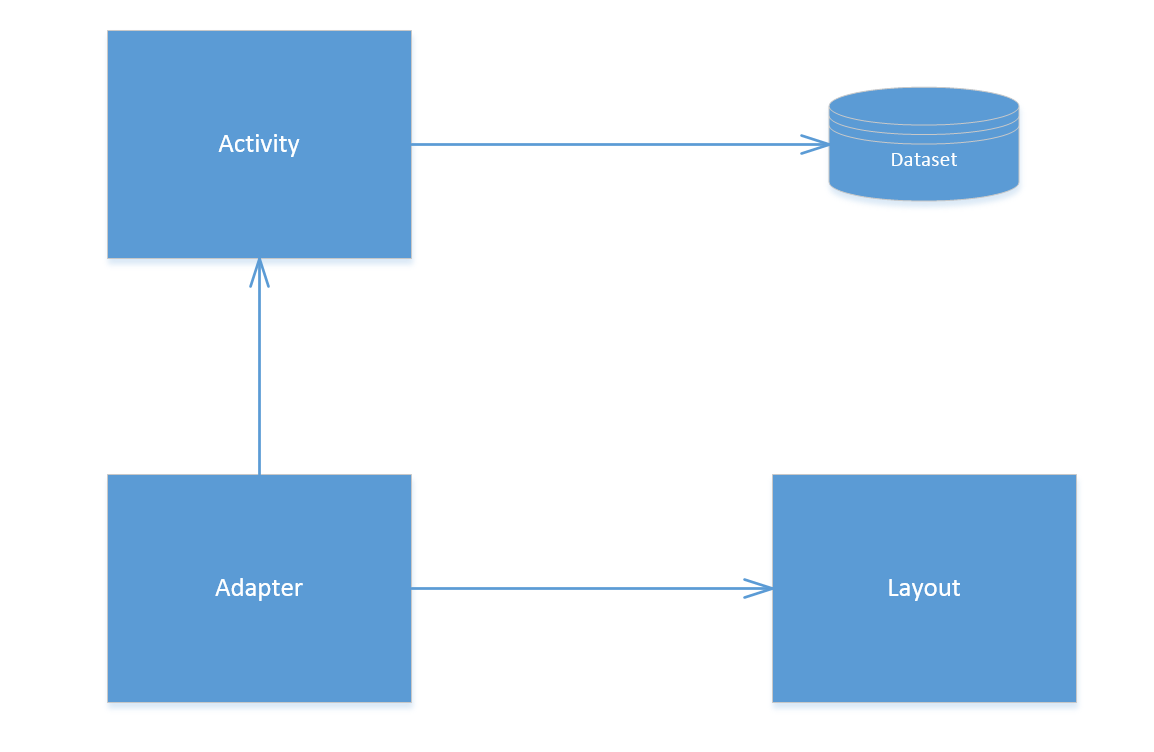


Figure 8: Overview of the interactions to show event items

## Issues while developing

**Fragment classes**: importing the wrong version

**What**: Cannot resolve method 'show(android.support.v4.app.FragmentManager, java.lang.String)'

getSupportFragmentManager() cannot be resolved

**When**: When running the methods “showDatePickerDialog(View v)” & “showTimePickerDialog(View v)”

**Success**: When extending DatePickerFragment() from android.support.v4.app.DialogFragment instead of

android.app.DialogFragment

**Using Service class and Thread**: To handle calls to JSON webservice

**What:** Issue to “control” the process of the thread execution and therefor problem to update the UI as expected.

**When**: In the beginning of the implementation of Android side to interconnect to JSON webservice.

**Success**: Using classes which extends AsyncTask, which gives us few methods where we can “easily” follow the process of Tasks execution.