ContactManager Documentation

This is a project assignment for the *Mobile Development* module.
The application was made by **Mathieu GUÉRIN** (3026954).

Table of contents

The application

The application meets the subject specifications.
It has activities and features for:

- Displaying the list of contacts
- Displaying information on a single contact
- Adding, editing and deleting a contact
- Displaying a simple *About* page

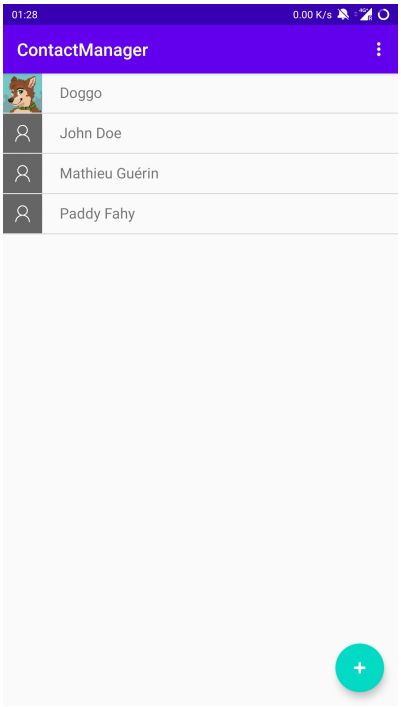
In total, the application uses four activities for these. It also uses a modified `DBOpenHelper` as an handler for using `SQLite`, as well as a `ContactInfo` class for managing the data in the database.

The subject specifications require implementing one additional feature and I've chosen to support **contact pictures**.

The User Interface

When opened, the application shows the list of contacts.

List of contacts (ListContacts)



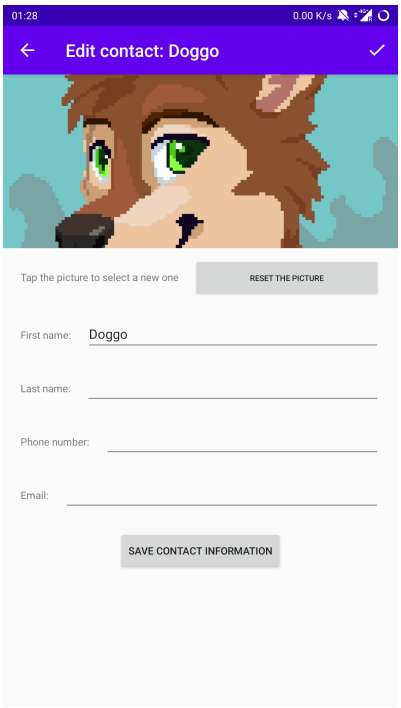
Contacts are sorted by first name then last name in the list. The layout for this activity is mainly a `ListView`, using a custom Adapter in order to show each contact with their picture and full name.
Clicking a row in the `ListView` will take the user to another activity with more details about the selected contact.

The toolbar of this activity has a menu with two items:

- One takes the user to the form for adding a contact
- The other takes them to the *About* page.

Lastly, the layout includes a floating button at the bottom of the screen, which also takes the user to the form for adding a contact.

Adding a contact (AddContact)



The form for adding a contact shows four text fields, for saving:

- The first name
- The last name
- A phone number
- An email address

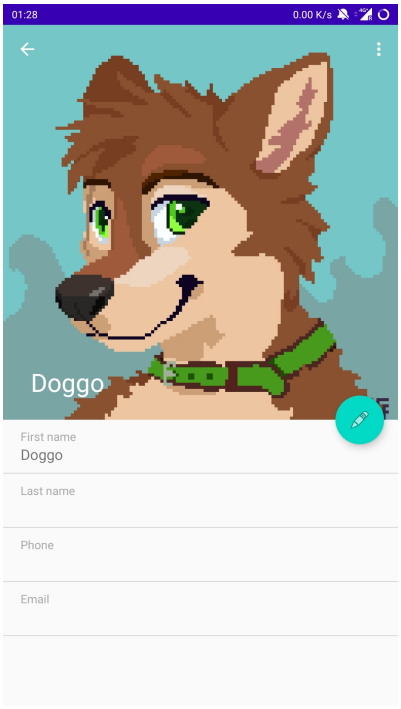
There is a button below the form to submit it, as well as a submit icon in the activity toolbar. The toolbar also includes a “back” icon to go back to the previous activity.

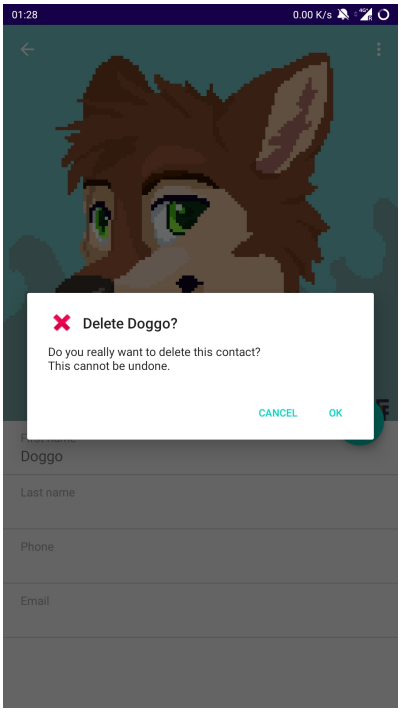
Additionally, an `ImageView` is present, which, when clicked, ask the user for a picture to save in the contact information. A

button is there to delete the picture and fall back to the default one.

Finally, this activity also serves the purpose of editing a contact. This is triggered from the detailed view of a contact when the user chooses to edit it. The form is then filled with the current contact information, and the text in the submit button is updated to reflect the new behavior of the form.

Detailed contact view (ShowContact)





The contact information is displayed in a nested scroll view as a `ListView`, in which each field has its row. The contact full name and picture are shown in the collapsing toolbar at the top of the screen. The user can scroll down to bring up the fields, shrink the toolbar and make the picture fade out.

The toolbar includes a “back” icon to go back to the previous activity, as well as a menu with two items:

- One for editing the contact
- The other for deleting it.

The layout also includes a floating button between the toolbar and the list of fields, which is only visible when the toolbar is not collapsed and will take the user to the form for editing the contact when clicked.

Additionally, a single field can be quickly edited from the list of fields itself by clicking the row of the field the user wish to edit.

Finally, the action of deleting the contact from the menu will open a confirmation prompt (using an `AlertDialog`). The action can then be confirmed or canceled.

About page (About)

The *About* page is just a simple page, to name the developer, me, and to specify the nature of this project: an assignment at Griffith College, for the *Mobile Development* module.

Code documentation

Activity Classes

ListContacts

- `protected void onCreate(Bundle savedInstanceState);`
- `private void setListeners();`

  Called from `onCreate()`.
  Sets event listeners for: adding a contact, opening details on a contact.

- `public boolean onCreateOptionsMenu(Menu menu);`
- `public boolean onOptionsItemSelected(MenuItem item);`

  Handles toolbar menu events for: adding a contact, opening the *About* page.

- `protected void onActivityResult(int requestCode, int resultCode, Intent data);`

  Handles results from other activities, triggers `reloadData()` if necessary.

- `private void reloadData()`

  Reloads the list of contacts.

ShowContact

- `protected void onCreate(Bundle savedInstanceState);`
- `private void setListeners();`

  Called from `onCreate()`.
  Sets event listeners for: editing the contact, quick-editing a field.

- `public boolean onCreateOptionsMenu(Menu menu);`

- `public boolean onOptionsItemSelected(MenuItem item);`

  Handles toolbar menu events for: leaving the activity, editing the contact, deleting the contact.

- `protected void onActivityResult(int requestCode, int resultCode, Intent data);`

  Handles results from other activities, triggers `reloadData()` if necessary.

- `private void reloadData()`

  Reloads the list of contacts.

- `private boolean updateDB(int id, String field, String value);`

  Updates a single field about a contact.
  Uses `ContactInfo.updateById()`.

AddContact

- `protected void onCreate(Bundle savedInstanceState);`

- `private void setListeners();`

  Called from `onCreate()`.
  Sets event listeners for: choosing a picture, resetting the picture, submitting the form.

- `public boolean onCreateOptionsMenu(Menu menu);`

- `public boolean onOptionsItemSelected(MenuItem item);`

  Handles toolbar menu events for: leaving the activity, submitting the form.

- `protected void onActivityResult(int requestCode, int resultCode, Intent data);`

  Handles results from picking a contact picture. Updates the form accordingly.

- `private void actionSubmit();`

  Handles the form submission.

- `private boolean insertContact(String lastname, String firstname, String phone, String email, Bitmap picture);`

  Adds a new contact in the database.
  Uses `ContactInfo.insert()`.

- `private boolean updateContact(long id, String lastname, String firstname, String phone, String email, Bitmap picture);`

  Updates an existing contact in the database.
  Uses `ContactInfo.updateById()`.

About

- `protected void onCreate(Bundle savedInstanceState);`

- `public boolean onOptionsItemSelected(MenuItem item);`

  Handles toolbar menu events for: leaving the activity.

Adapters

ListContactsAdapter

- `public ListContactsAdapter(Context context, List<ContactInfo> objects);`

  Constructor of the class.
  It defines `objects`, the list of elements that will be in the `ListView`.

- `public View getView(int position, View convertView, ViewGroup parent);`

  Method that will construct each row of the `ListView`.
  Uses `ViewHolder`.

- ```
  private static class ViewHolder {
          ImageView contact_picture;
          TextView txv_list_contacts;
  }
  ```

  Structure element for a single row in the `ListView`.
  ```

ShowContactsAdapter

- `public ShowContactAdapter(Context context, List<FieldInfo> objects);`

  Constructor of the class.
  It defines `objects`, the list of elements that will be in the `ListView`.

- `public View getView(int position, View convertView, ViewGroup parent);`

  Method that will construct each row of the `ListView`.
  Uses `ViewHolder`.

- ```
  private static class ViewHolder {
          TextView txv_field_name;
          TextView txv_field_value;
  }
  ```

  Structure element for a single row in the `ListView`.

Structures

ContactInfo

Structure class for a contact: stores information and implement a method to get the contact picture.
Also provides a range of static methods to get, update or delete contacts.

- ```
  public final long id;
  public final String lastname;
  public final String firstname;
  public final String phone;
  public final String email;
  ```

- `private Bitmap picture;`

- `public ContactInfo(long id, String lastname, String firstname, String phone, String email);`

  Constructor of the class. It sets the `sdb` field for use.

- `public String getFullName();`

  Returns the contact full name.

- `public Bitmap getPicture(Context context, boolean fallbackDefault);`

  Fetch the contact picture if necessary, and returns it.
  If the picture cannot be found and `fallbackDefault` is true, the default contact picture will be returned. Otherwise, null is returned.

- ```
  public static final String DB_NAME;
  public static final String TABLE_NAME;
  ```

- ```
  public static final String CREATE_TABLE;
  public static final String DROP_TABLE;
  public static final String UPGRADE_TABLE;
  ```

- `public static String getFieldById(DBOpenHelper db, String column, long id);`

  Returns a single field about a contact.

- `public static ContactInfo getById(DBOpenHelper db, long id);`

  Returns a `ContactInfo` object containing information about a contact.

- `public static List<ContactInfo> getAll(DBOpenHelper db);`

  Returns a list of all contacts in the database, as `ContactInfo` objects.

- `public static boolean insert(DBOpenHelper db, ContentValues cv);`

  Adds a new contact in the database. `cv` defines the field values.

- `public static boolean updateById(DBOpenHelper db, ContentValues cv, long id);`

  Updates an existing contact in the database. `cv` defines the new field values.

- `public static boolean deleteById(DBOpenHelper db, long id);`

  Deletes a contact from the database.

- `public static Bitmap getDefaultPicture(Context context);`

  Returns the default picture for a contact.

FieldInfo

Simple structure class for use in `ShowContactsAdapter`.

- ```
  public final String field;
  public final String name;
  public final String value;
  ```

- `public FieldInfo(String field, String name, String value);`

  Constructor of the class. It initializes the fields.

Helper Classes

DBOpenHelper

This class is only used to connect to a database.
SQL queries are handled outside of this class: mainly in `ContactInfo`.

- `public final SQLiteDatabase sdb;`

- `public DBOpenHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int version);`

  Constructor of the class. It sets the `sdb` field for use.

- `public void onCreate(SQLiteDatabase sdb);`

- `public void onUpgrade(SQLiteDatabase sdb, int oldVersion, int newVersion);`

BitmapHelper

This class is used to encode and decode Bitmap in base64.
This is part of the additional feature implementation.
  ```

- `public static String encodeBitmap(Bitmap bitmap);`

- `public static Bitmap decodeBitmap(String encoded);`