APPLICATION ON ANDROID - “CHECK INBOX MAIL”

Gina GUȘAN, scientific supervisor Mihail KULEV

Technical University of Moldova

Abstract: *”CHECK INBOX MAIL” application creates a connection between the user’s mobile phone and a certain e-mail adress, which guarantees access to all the messages stored already in Inbox. Also, the application is designed to provide writing new messages, sending them to other e-mails, downloading attachements. Practically, it offers the possibility of handling virtually all of users’ e-mail needs.*

Keywords: *mobile phone apps, mobile internet connection, e-mail, virtual message, Android OS.*

1. Introduction

The Android operating system includes a large variety of applications or features. One of the most significant improvements, used by all the people in the world, was accomplished in the internet domain. This means that, when you want to connect to your e-mail account, you don’t have to use a computer anymore. It is sufficient to have a mobile phone running on Android operating system [1].

2. Description of the application

*”CHECK INBOX MAIL”*ensures users with the necessary tools for increasing their mobility. Meaning that, by starting the application (Fig.1), you can just log in to your e-mail account (Fig.2). Instantly, you’ll receive acces to all needed data.

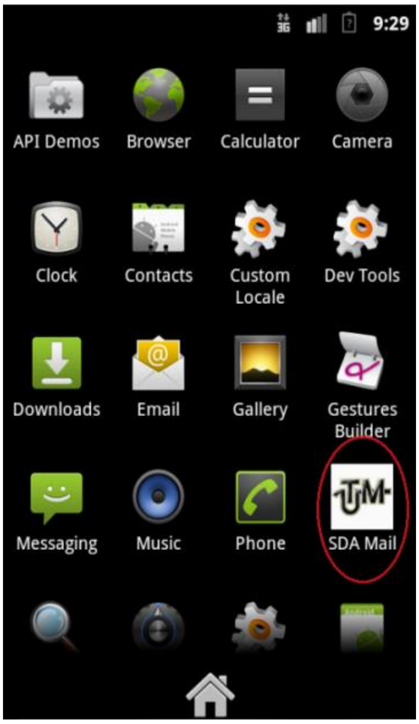
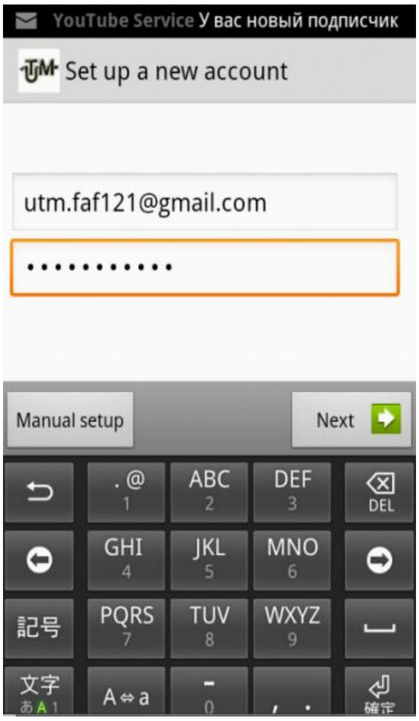


Fig.1. Starting app



Set up a new account

utm.faf121@gmail.com

•••••••••••

Fig.2. Log-in

By clicking the button „Next”, the application will run and the new interface will allow you to perform several actions. For example, the user can check Inbox for new messages: in our case, all messages are considered as new, because the application was running the first time (Fig.3).

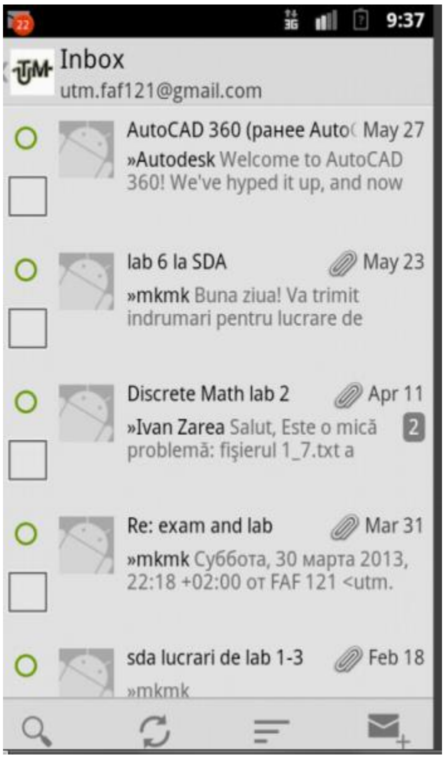
Another possible action can be the visualising of the old messages, stocked in the mail. The income messages, the sent messages, the drafts are accessible.

Also, if desired, the user can write a new e-mail, by indicating the e-mail adress of the receiver, the subject of the conversation and the message text itself (Fig.4).

If a text was written whithout the intention of being send, it could be saved to Drafts.

Resending an already sent message is possible.

When needen, the applicationis ensured with the possibility of showing the attachements from received messages (Fig.5)


9:37

Inbox
utm.faf121@gmail.com

AutoCAD 360 (ранее Auto(May 27
»Autodesk Welcome to AutoCAD 360! We've hyped it up, and now

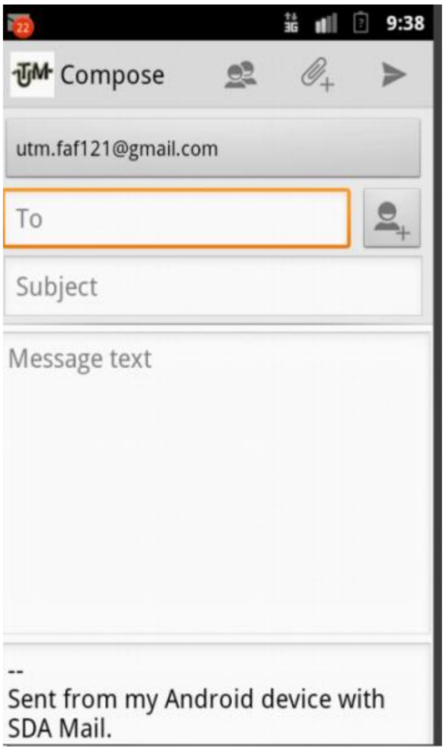
lab 6 la SDA May 23
»mkmk Buna ziua! Va trimit indrumari pentru lucrare de

Discrete Math lab 2 Apr 11
»Ivan Zarea Salut, Este o mică problemă: fişierul 1_7.txt a 2

Re: exam and lab Mar 31
»mkmk Субота, 30 марта 2013, 22:18 +02:00 от FAF 121 <utm.

sda lucrari de lab 1-3 Feb 18
»mkmk

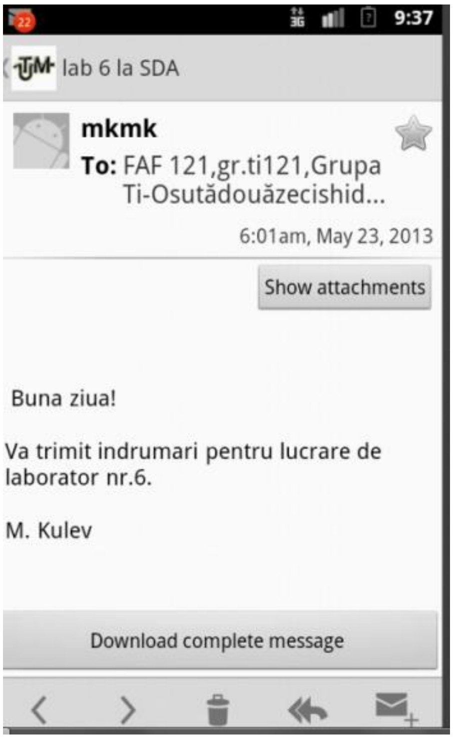
Fig.3


9:38

Compose
utm.faf121@gmail.com
To
Subject
Message text

--
Sent from my Android device with SDA Mail.

Fig.4


9:37

lab 6 la SDA

mkmk
To: FAF 121,gr.ti121,Grupa Ti-Osutădouăzecishid...
6:01am, May 23, 2013

Buna ziua!

Va trimit indrumari pentru lucrare de laborator nr.6.

M. Kulev

Fig.5

3. Tools and codes

This application is a complex one, that’s why it has a lot of classes that can be considered trivial [2]. The first class presented is used for starting a long thread that will run through commands that require remote mailbox access. This class is used to serialize and prioritize these commands. The code is presented below:

```
   public      static      class      UidReverseComparator      implements
Comparator<Message> {
           @Override
   publicint compare(Message o1, Message o2) {
   if (o1 == null || o2 == null || o1.getUid() == null || o2.getUid() ==
null) {
      return 0;
                }
   int id1, id2;
   try {
               id1 = Integer.parseInt(o1.getUid());
               id2 = Integer.parseInt(o2.getUid());
                }
      catch (NumberFormatException e) {
   return 0;
                }
   //reversed intentionally.
   if (id1 < id2)
   return 1;
   if (id1 > id2)
   return -1;
   return 0;

            }

        }
```

Another essential class can be considered the class that saves messages to drafts [3]. It will return Message representing the entry in the local store. Here is the representative piece of code:

```
   public   Message   saveDraft(final   Account   account,   final   Message
message, long existingDraftId) {
           Message localMessage = null;
      try {
   LocalStorelocalStore = account.getLocalStore();
   LocalFolderlocalFolder
=localStore.getFolder(account.getDraftsFolderName());
   localFolder.open(OpenMode.READ_WRITE);

   if (existingDraftId != INVALID_MESSAGE_ID) {
                  String                    uid                      =
localFolder.getMessageUidById(existingDraftId);
   message.setUid(uid);
                }

                // Save the message to the store.
   localFolder.appendMessages(new Message[] {
   message
                                          });
                // Bring the message back from the store.
                  //This is the Message that's returned to the caller.
   localMessage = localFolder.getMessage(message.getUid());
   localMessage.setFlag(Flag.X_DOWNLOADED_FULL, true);

   PendingCommand command = new PendingCommand();
   command.command = PENDING_COMMAND_APPEND;
   command.arguments = new String[] {
   localFolder.getName(),
   localMessage.getUid()
```

```
              };
queuePendingCommand(account, command);
processPendingCommands(account);

        } catch (MessagingException e) {
Log.e(sda.LOG_TAG, "Unable to save message as draft.", e);
addErrorMessage(account, null, e);
        }
returnlocalMessage;
    }
```

Also, an important class can be considered the one that packs an address list into a String which can be quickly read and parsed [3]. Packed lists can be unpacked with unpackAddressList() function:

```
public static String pack(Address[] addresses) {
        if (addresses == null) {
            return null;
        }
        StringBuilder sb = new StringBuilder();
        for (int i = 0, count = addresses.length; i < count; i++) {
            Address address = addresses[i];
            sb.append(address.getAddress());
            String personal = address.getPersonal();
            if (personal != null) {
                sb.append(";\u0000");
                // Escape quotes in the address part on the way in
                personal = personal.replaceAll("\"", "\\\"");
                sb.append(personal);
            }
            if (i < count - 1) {
                sb.append(",\u0000");
            }
        }
        return sb.toString();
    }
}
```

4. Conclusion

Nowadays, people tend to facilitate interchanging information as much as possible. Android is an open source platform that offers all the necessary utilities to fix this problem. One of the biggest contributing factors to mobile phones is the ability to receive notifications and check messages at any instant. This is especially true for e-mails, which fly into our inbox all throughout the day. *"CHECH INBOX MAIL"* guarantees an easy handling for being in touch with everything that happens on our e-mail accounts. It uses existing folder structure, can synchronize drafts, sent, and deleted messages. The application itself is quite intuitive and clear in the description. It represents a good solution for everyday usage.

5. Bibliography

1. HASHIMI, Sayed, KOMATINENI, Satya, McLEAN, Dave. *Pro Android 2,* Apress, 2010 ,736 p.
2. *How to parse in Java:* *http://www.tutorialspoint.com/java/number_parseint.htm* (accessed15.10.2013).
3. *Parsing XML data :* http://developer.android.com/training/basics/network-ops/xml.htmlp (accessed 15.10.2013)