katholieke hogeschool
associatie KU Leuven
vives

A mobile application for manuscript signing of remote documents

industriële wetenschappen en technologie
bachelor in de elektronica-ICT

Eindwerk aangeboden tot het
behalen van het diploma van
bachelor in de elektronica-ICT

door Arne Debergh

Campus Oostende
Academiejaar 2015 - 2016

o.l.v. Tom Cordemans, Vives
Luc Vanhee, Vives
Carlos Vivaracho Pascual, Escuela de Ingeniería Informática de Valladolid
Javier Bastida Ibáñez, Escuela de Ingeniería Informática de Valladolid

Announcement

This final project was an exam. The discovered errors during the defence presentation are not included.

Deze eindverhandeling was een examen. De tijdens de verdediging geformuleerde opmerkingen werden niet opgenomen.

Abstract in English

This bachelor thesis is about an Android and web application for handling the signing of documents electronically. The goal is to be able to send, receive and possibly sign documents electronically, thus limiting the use of paper as well as speeding up the process. This book will cover everything I've been working on, from design and coding to the implementation of the application.The usage of the application will be explained with a graphic user manual.

Abstract in Dutch

Dit eindwerk gaat over een Android- en webapplicatie voor het elektronisch afhandelen van het ondertekenen van documenten. Het doel is om documenten elektronisch te verzenden, te ontvangen en eventueel te ondertekenen. Hierdoor wordt het gebruik van papier geminimaliseerd terwijl het ganse proces sneller kan plaatsvinden. Dit boek omvat alles waar ik aan gewerkt heb, van het ontwerp en codering tot de implementatie van de applicatie. Het gebruik van de applicatie wordt toegelicht door middel van een gebruikershandleiding.

Abstract in Spanish

El Proyecto que se describe en esta memoria consiste en una aplicación Android y una aplicación web para la firma electrónica de documentos. El principal objetivo es que la aplicación pueda enviar, recibir y firmar documentos electrónicamente, limitando así el uso de papel y agilizando el proceso. Esta memoria describe todo el proceso de desarrollo de la aplicación, tanto el diseño de la misma como la implementación. Esta memoria incluye a su vez un manual de usuario donde se explica el uso de la herramienta.

Keywords

Android, mobile application, manuscript signing, remote documents

Citation

Arne Debergh: A mobile application for manuscript signing of remote documents,

Bachelor Thesis, Valladolid, E.T.S. De Ingeniería Informatica, 2016

Dedication in English

I would like to thank Vives Oostende for giving me the opportunity of completing my studies with an Erasmus+ program. More specifically I would like to thank my internal promotor Cordemans Tom and supervising professor Vanhee Luc. I would also like to thank my parents for supporting me in my decision of a semester abroad. It would not have been possible to have such a unique experience without their support. Also, I would like to thank my external promotor Carlos Vivaracho Pascual and supervising professor Javier Bastida Ibáñez for helping me choose a project. I was always welcome at their offices to ask for help or feedback with my work. This final project has given me a taste of the field in general as well as a working culture quite different from the one in Belgium where I'm probably going to end up in. This experience as a whole will most likely prove to be an asset on the job market in the future. I have met a lot of new friends and made a lot of memories here, all of those I will hold dear for the rest of my life. My Erasmus+ period is something I will never forget.

Dedication in Dutch

Ik zou graag Vives Oostende bedanken om me de kans te geven mijn studies te vervolledigen met een Erasmus+ programma. Meer bepaald zou ik graag mijn interne promotor Cordemans Tom en de toezicht houdende professor Vanhee Luc bedanken. Ook mijn ouders zou ik willen bedanken om mij te steunen in mijn keuze voor een semester in het buitenland. Het zou voor mij onmogelijk geweest zijn zo'n unieke ervaring te beleven zonder hun ondersteuning. Ook zou ik graag mijn externe promotor Carlos Vivaracho Pascual en toezicht houdende professor Javier Bastida Ibáñez bedanken voor hun hulp bij het kiezen van mijn project. Ik was altijd welkom in hun kantoor om hulp of feedback op mijn gemaakte werk te vragen. Dit eindwerk heeft me kennis laten maken met het werkveld in het algemeen, alsook met een werkcultuur verschillend van de Belgische waar ik waarschijnlijk in terecht zal komen. Deze volledige ervaring zal zeer waarschijnlijk een grote troef zijn op de arbeidsmarkt. Ik heb hier veel nieuwe vrienden leren kennen, alsook veel herinneringen gemaakt, die ik voor altijd zal koesteren. Mijn Erasmus+ periode is iets wat ik nooit zal vergeten.

© Arne Debergh
This work was created as a thesis at the University of Valladolid, Departamento de Informática.
This work is protected by copyright and law, its use without authorisation by the author is illegal, except in cases defined by law.

1 Table of contents

2 List of figures

3 Introduction

Signing documents mostly happens manually, which takes up time and use of paper. Therefore an application was made capable of signing documents remotely. By doing this the occupied time of this process is being significantly reduced, as well as the use of paper.

The application developed in this project has been made for a fictional bank, which needed a solution for handling contracts with their customers in a more efficient and less polluting way. Beforehand they would print out contracts and send them to their customers through mail or meet up with them.

With this application, the users are able to view their pending contracts at any given place or time. The users can respond to these contracts, accepting or declining them, with the touch of a finger. This project is an example of how technology can simplify and speed up common tasks.

Technology has become a big part of our daily lives. Nowadays, almost everyone owns a smartphone and has immediate access to the internet. Google's Android operating system or OS is the most common OS in the world, as well as being the most accessible OS to write applications for. It currently holds over 80% of the worldwide market share in smartphone operating systems.

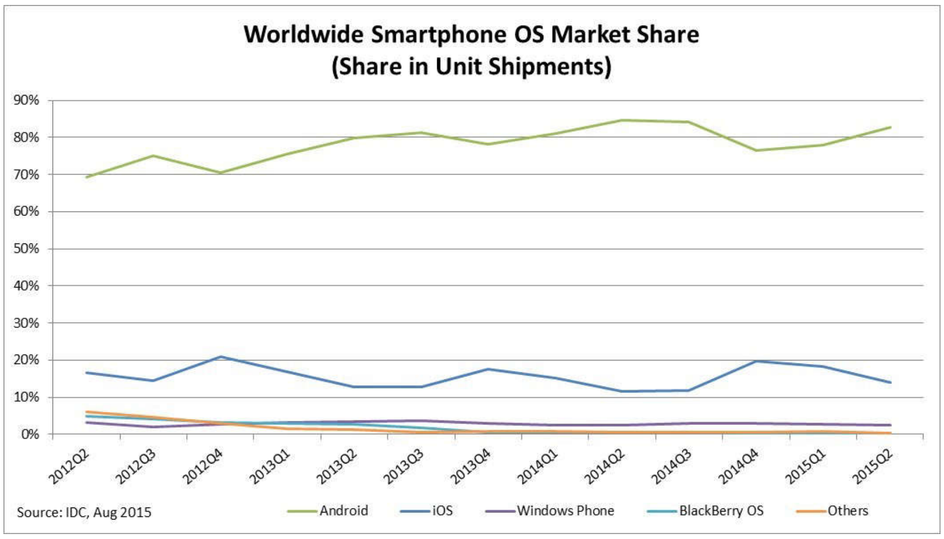


Figure 1: Smartphone OS market share (1)

The biggest market share competitor is Apple's iOS, which currently holds around 13% of the worldwide market share, as seen in figure 1. iOS is much less accessible to write applications for, it's not even free if you want to do so. The other mobile operating systems are negligible. That's why our choice of OS was obvious from the beginning: Android.

Android was created by the Open Handset Alliance, which is led by Google. Developing software for Android is usually done in the Java programming language using the Android software development kit (SDK) , but other development environments are available as well.

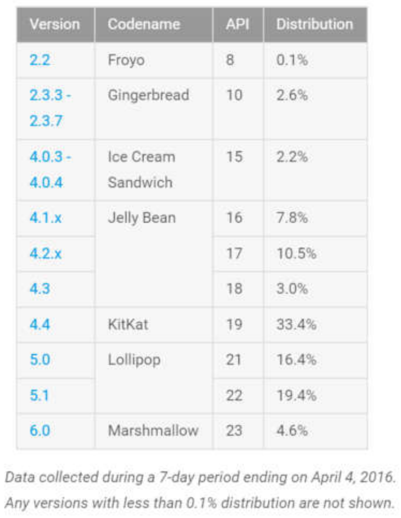
Since the popularity of Android started rising, a lot of applications have been created to make the lives of its users easier. Because of the constant improvement of the hardware, smartphones are becoming bigger, more powerful and faster every year.

This project consists of an Android application and a web application, who are working together as one single application. There are also some webpages that can be accessed, this will mainly be done by the employees of the bank rather than its customers.

For the creation of this project, I had to implement my knowledge of Android, HTML, CSS, PHP and MySQL. I've learned all of these coding languages in separate courses over the past few years, but this was actually the first time I used them all together. These languages are, just like hardware, constantly evolving. Undoubtedly this will lead to faster and more powerful programming

The application built is pretty hardware-independent, because the contracts and signatures are stored in the database and only accessed when called upon. The most important requirements are having a touchscreen and internet connection.

The mobile application has been designed for Android API version 23, but the minimum API version is 16. This means the application will work on any Android operating system from version 4.1 (Jelly Bean) upwards to the newest version to date, 6.0 (Marshmallow). The application was also tested on OxygenOS 2.0 and 3.0, these are custom ROM's based on Google's Android mobile system. Because of the support starting from Android 4.1 up to 6.0, this application can at the moment be used on 95.1% of all the Android smartphones in the world, as verified in figure 2.



Version	Codename	API	Distribution
2.2	Froyo	8	0.1%
2.3.3 - 2.3.7	Gingerbread	10	2.6%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	2.2%
4.1.x	Jelly Bean	16	7.8%
4.2.x		17	10.5%
4.3		18	3.0%
4.4	KitKat	19	33.4%
5.0	Lollipop	21	16.4%
5.1		22	19.4%
6.0	Marshmallow	23	4.6%

Data collected during a 7-day period ending on April 4, 2016.
Any versions with less than 0.1% distribution are not shown.

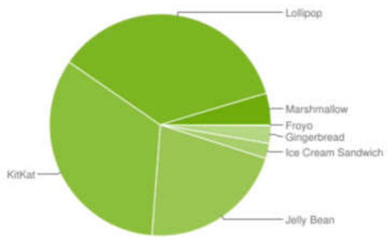


Figure 2: Android version spread as of April 4, 2016 (2)

The database has been written in MySQL and executed on a public server of the University of Valladolid. This database was managed using phpMyAdmin. All the connections in the code have been made to this server, so when afterwards the database would be taken offline, it would have to be put on another server for the entire project to function again.

4 General description of the project

This project is in fact my bachelor thesis, for which I had 4 months of time to complete. The general goal of it is to achieve a system that allows the user to sign documents remotely by means of a mobile device with tactile screen. The project exists of an Android application, a web application and a website. The Java code was written for the user interface on the Android Application. The PHP code was used on the server side, to make the connection between server and database and adjust the database when needed. MySQL was only used for setting up the database. I used HTML and CSS to make a simple web interface, with the purpose of serving the banks' employees. In this chapter, the general goal will be divided into the mobile application, database and web application, which will be divided in smaller, more specific goals.

4.1 Project Requirements

In this part I will explain the specific goals set out during the course of my project. Here will also be explained briefly how these goals were achieved.

4.1.1 Android application requirements

The mobile application must allow the users to read their documents and to perform a manuscript signature being sent to the server.

4.1.1.1 Creating a login

To have a login on the Android-side is necessary because users should not be able to view documents meant for other users. This is why the login screen is prompted when starting the application. It's possible to register as well as log in when starting the application. This action makes use of the PostClass mentioned on the next page. The flowchart of this process can be seen in figure 3.

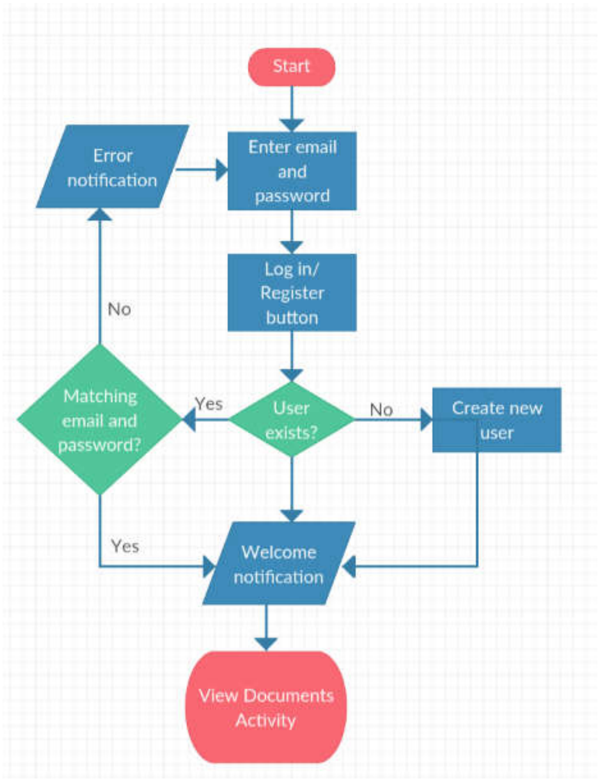


Figure 3:Android Login Flowchart

4.1.1.2 Making connection to the database

Establishing a connection with the database is needed several times when using the application. That's why a class called 'PostClass' was implemented, which is able to contact a given PHP script on the server, send some parameters to it and receive a response code.

4.1.1.3 List the user documents

After logging in, the users should be able to view all their documents in a table, with some basic information as title and status. For this part, a ListView was implemented which is being populated by a custom adapter. This adapter will put all documents of the logged in user in separate rows of the table.

4.1.1.4 Selecting a document

When the users have access to the table containing their documents, they have to be able to respond to a single document. Therefore an extra column has been put in the table containing a button, which leads the user to a new activity.

4.1.1.5 Viewing a PDF file

When the users select a document, they must be able to read it in order to respond to it. A web view is used so the document will be consulted through a URI and not downloaded to the device.

4.1.1.6 Responding to the document

If the users select a document, they will get to see the document in a web view as mentioned previously. There are two buttons at the bottom of the screen, to accept or decline the document. When the users accept the document, they will be redirected to the signing activity. When the users decline it, a PHP script on the server will be called declining said document. The flowchart of this operation can be seen in figure 4.

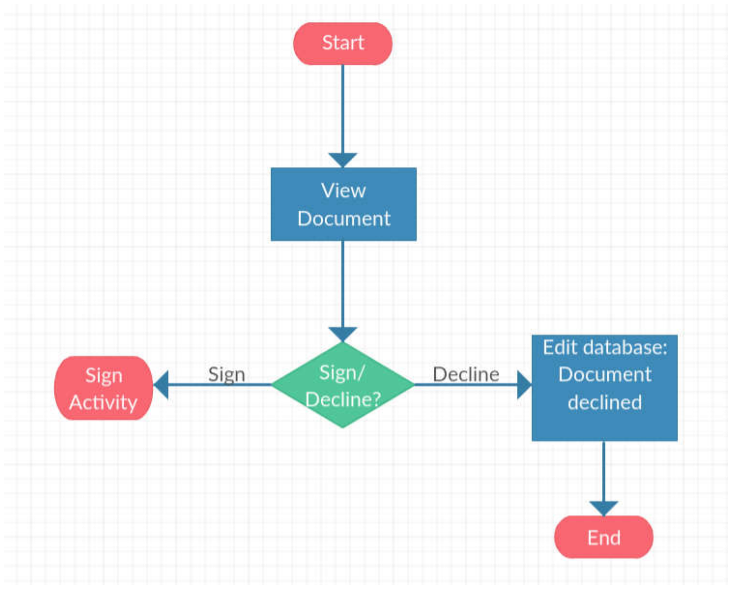


Figure 4:Respond to Document Flowchart

4.1.1.7 Drawing and saving a signature

Since the most important part of this application is signing documents, It is needed to implement a way to draw a signature with the touchscreen. This signature is saved temporarily and uploaded to the database when needed. The flowchart of this operation can be seen in figure 5.

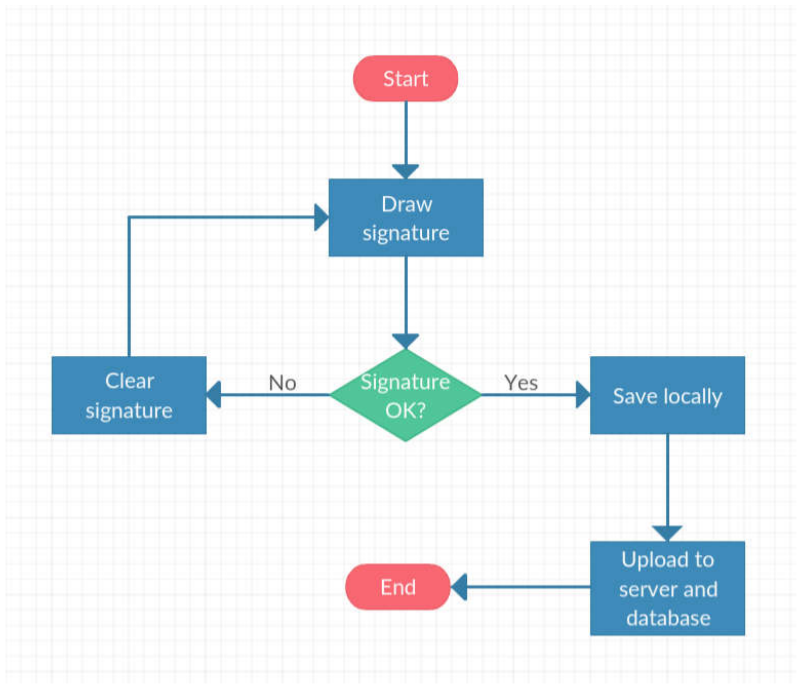
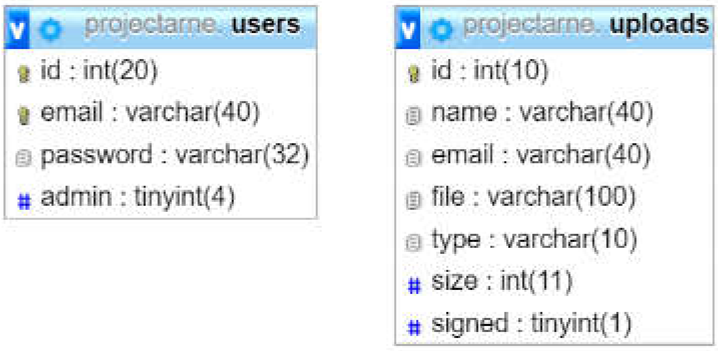


Figure 5:Flowchart Sign Document

4.1.2 Database requirements

This application would not function correctly without the use of a database. The one that is used for this application contains two tables: users and uploads. Each user has an id, email and password, the latter being stored as a md5 hash. The users table also contains a ‘admin’ column, which can be manually adjusted to the value ‘1’ to give the desired user administrator rights. Each row in the uploads table contains an id, name, email, file, type, size and signed column.



projectarne. **users**

- id : int(20)
- email : varchar(40)
- password : varchar(32)
- admin : tinyint(4)

projectarne. **uploads**

- id : int(10)
- name : varchar(40)
- email : varchar(40)
- file : varchar(100)
- type : varchar(10)
- size : int(11)
- signed : tinyint(1)

Figure 6: Database Tables

4.1.3 Web application requirements

The web application must simulate a real world application where the users can review the documents and their signatures.

4.1.3.2 Creating a login

Although the web application should be hosted local, it must simulate a real web application, therefore this part is mandatory. This login form is situated on the home page and will call a PHP script which validates if the user has administrator rights, meaning he can upload and view documents. If the login is invalid or the users do not have administrator rights they are limited to the home and contact page. There also is a logout button so the users can log out when they’re done with their work.

4.1.3.2 Uploading documents

The employees of the bank need to be able to upload documents and link them to a customer. Therefore a webpage with a form is made where the employees can upload PDF documents and enter an email address. When the form is being posted, the document will be saved on the server and the database will be updated.

4.1.3.3 Documents table

The employees of the bank have to be able to quickly view the status of a document. Therefore there is a webpage which displays a table containing row with the document name, recipient and a button to view it. When the recipient has responded to the document, the row colour will appear green or red whether the recipient signed or declined it.

4.1.3.4 View a single document

When employees want to view a document, they can click the button in the corresponding row of the table previously mentioned and another tab will open containing the document.

4.2 Methodology

In this part I will explain how the Android application connects to the server and database. Some diagrams will be used to simplify the explanation.

4.2.1 Connecting to the database: Android login

When the user presses the login button on the application, the entered email and password are being posted to a specific URL. The server uses the user_control.php file to control if the values are valid. The flowchart of this script can be seen in figure 7. It will also add a new user when the entered values do not exist in the database yet.

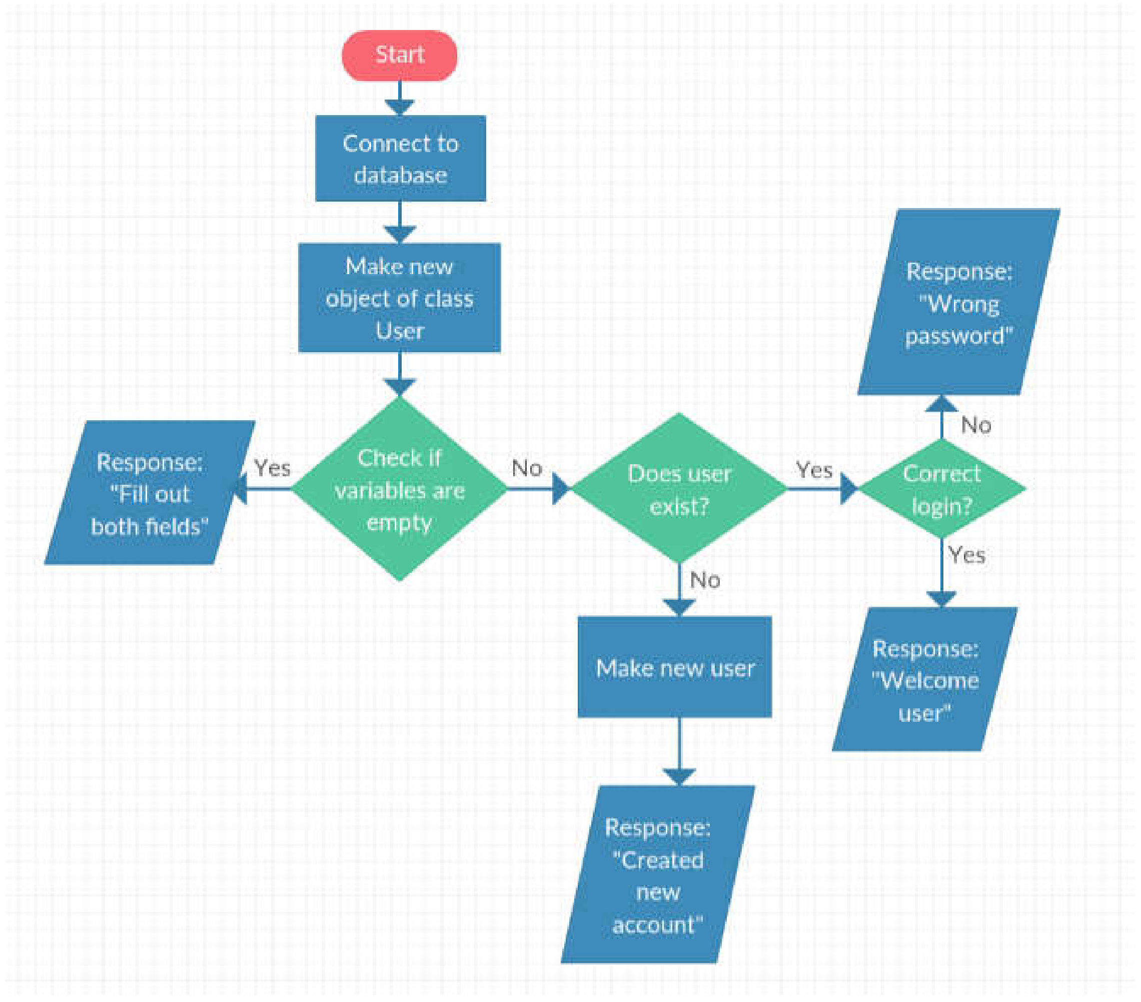

Start
Connect to database
Make new object of class User
Check if variables are empty
Yes
Response: "Fill out both fields"
No
Does user exist?
Yes
Correct login?
No
Response: "Wrong password"
Yes
Response: "Welcome user"
No
Make new user
Response: "Created new account"

Figure 7: Flowchart of user_control.php script

Firstly, the script will establish a connection with the database.

A new object of the User class will be made, this class will be explained later on. Then, the script will look if the post contains an 'email' and 'password' value. If so, local values will be set accordingly to the posted ones. Next, we will check if both the values are not empty. When they are both not empty, the password value will be hashed with an md5 hash and both values will be given as arguments to the function 'does_user_exist'. When at least one of the values is empty, the script answers with the message "You must fill out both fields".

The User class contains two functions, __construct and does_user_exist. The construct function will establish the database connection through the get_connection function of the connection.php script.

The function ‘does_user_exist’ will firstly send a query to the database selecting the row where the email and password fields match the entered ones. If this MySQL query contains a row, the script will answer with a welcome message, personalized to the user logging in and the connection will close.

If that query would appear to be empty, a new query will be executed trying to insert a new row to the database containing the email and hashed password of the user. When the result of the executed query is a ‘1’, that means the new row was inserted successfully and the application will receive a message stating a new account was created. If the result would not be equal to ‘1’, that means the corresponding user already has a row in the database but the password was incorrect, so the script answers with a wrong password message.

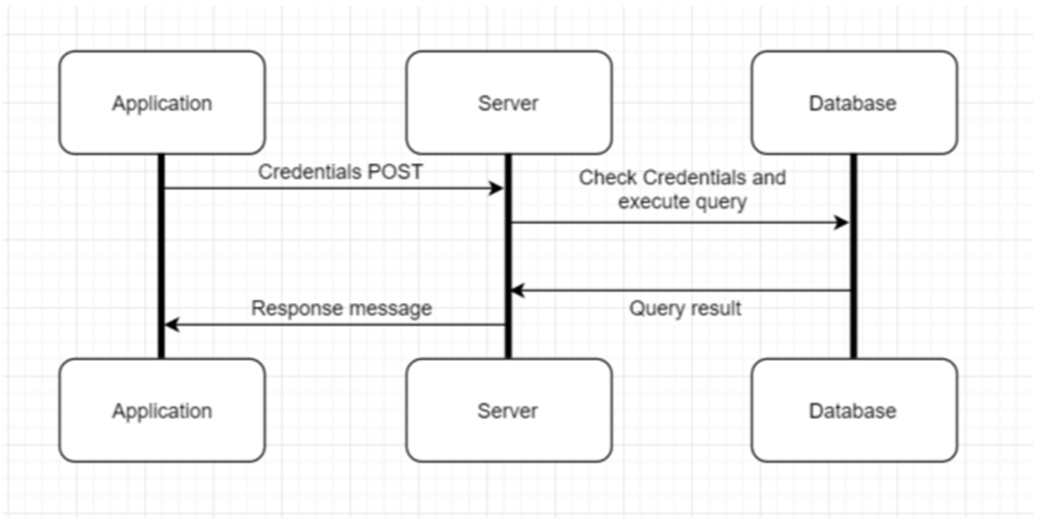


Figure 8: Sequence diagram for Android login

4.2.2 Connecting to the database: Webpage login

The webpage login script is similar to the Android login script but I have made some changes to the 'does_user_exist' function:

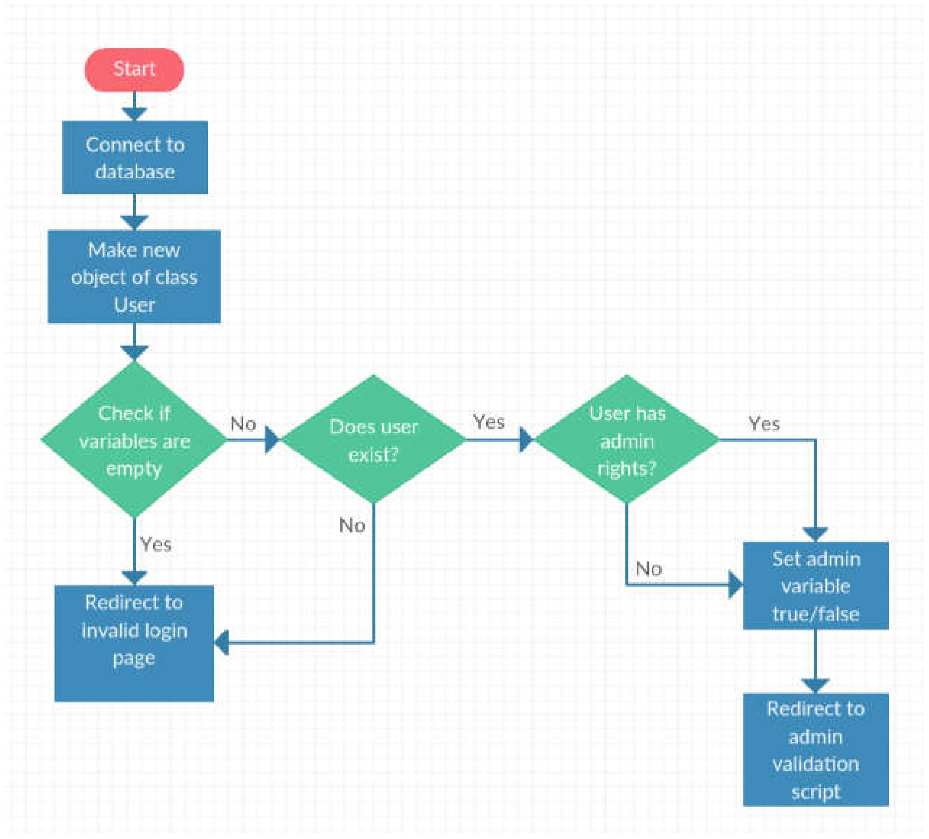

Start
Connect to database
Make new object of class User
Check if variables are empty
No
Does user exist?
Yes
User has admin rights?
Yes
Yes
No
No
Redirect to invalid login page
Set admin variable true/false
Redirect to admin validation script

Figure 9:Website Login Flowchart

If the user exists, the script will validate if the 'admin' column in the users table is equal to '1'. If so, a session variable will be made and set true stating that the current user is in fact an administrator. If not, this variable will be set false. If the user is non-existing, you will be redirected to an invalid login webpage. After the user is recognized, the browser will be redirected to a script which is used when the documents table webpage is being accessed. That script will look if the session variable is true or not, and later on redirect to the desired page.

```
session_start();
if ($_SESSION['admin'] != 'true')
   {
   header("Location: http://rea.eca-simm.uva.es/projectarne/index.html");
}else {
            header("Location: http://rea.eca-simm.uva.es/projectarne/documents.php");
};
```

To view the webpage for adding documents, a similar script is used redirecting to the webpage for adding documents when the user has administrator rights.

4.2.3 Database Operation: Uploading documents

It is possible for the employees to upload new documents and assign them to a recipient. This is done in the 'Add Document' webpage, which can only be accessed after logging in. When the form is posted, it will start a script called 'upload.php', which will handle the uploading of the file and alter the database.

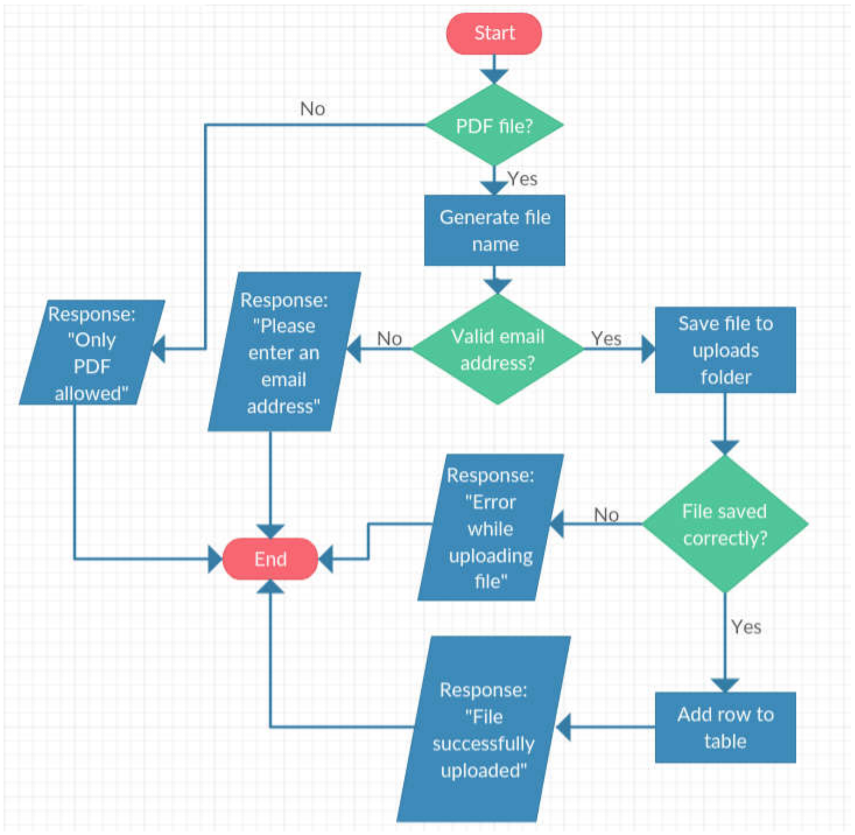

Start
PDF file?
No
Yes
Generate file name
Response: "Only PDF allowed"
Response: "Please enter an email address"
No
Valid email address?
Yes
Save file to uploads folder
Response: "Error while uploading file"
No
File saved correctly?
End
Yes
Response: "File successfully uploaded"
Add row to table

Figure 10: Flowchart of upload.php

In the flowchart seen in figure 10, you see that the script will firstly check if the file being uploaded is a PDF file. When a valid file is used, a file name will be generated with the given email and name. The name that was entered will be converted to lower case and each space will be converted to a dash. For example when the user entered 'THIS is Your DocUment' as the name and 'demo@gmail.com' as the email, the generated filename will be as follows: 'this-is-your-document-demo@gmail.com.pdf'.

Next the email address will be validated. If the form contained a valid email address the file will be uploaded to the server, this action will be checked. The checking of file upload also takes care of documents having the exact same name. If the file was uploaded correctly, a new row containing the info of the document will be put in a new row in the database table.

When the uploaded file was not a PDF, the email address was invalid, the file upload failed or a new row was added to the table, a corresponding response will be sent back to the webpage.

4.2.4 Database Operation: Webpage documents table

After uploading documents the employees must be able to view them. Therefore exists a webpage which displays a table containing all documents. I have embedded some pieces of PHP code into the HTML code because it made for a very simple solution. To make as much rows as needed, I select all documents from the database and run them through in a while loop. The only database operation that is happening here is 'SELECT * FROM uploads'.

Every loop will generate a new row, containing the information of the document in that row. It will show the name, email of the recipient, size of the document, a button to view the PDF and a button to view the signature, when the document has been signed. Also, the background colour of the row changes depending on its status. If the document is signed, the background colour will appear green, when declined it will be red. In case the recipient has not yet responded to the document, the background colour will be neutrally white.

4.2.5 Database Operation: Android list view own documents

To make a list view table of all the documents of the end user, a script called 'list_view.php' is being called. This script puts all rows from the uploads tale in an array containing encoded JSON strings. Every JSON string in the array contains a response message. This array will be sent to the device and it will be decoded. Only the strings containing documents of that end user will be decoded and displayed.

4.2.6 Database Operation: Android signature upload

After drawing and confirming a signature on the Android application, the PNG file will be posted to another script called 'uploadSig.php'. This script will save the uploaded file into the uploads folder and respond to the application when its operation was successful or failed. The file name of the signature will be exactly the same as the corresponding document, but with the PNG extension instead of PDF .

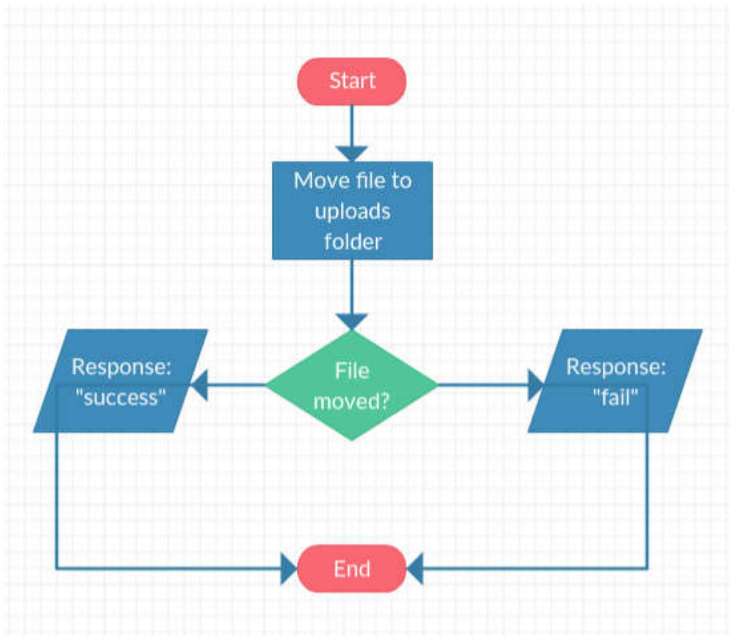


Figure 11: Flowchart of uploadSig.php

4.2.7 Database Operation: Android Accept/Decline document

When the users have read their document, they have 2 choices: going to the signing of the document, or declining it. If the decline button is clicked, the PostClass will be called and connect to decline.php. This script will receive the name of the document that is being declined, and use a sql query to set the signed value to ‘0’. The ‘0’ value means this document has been declined.

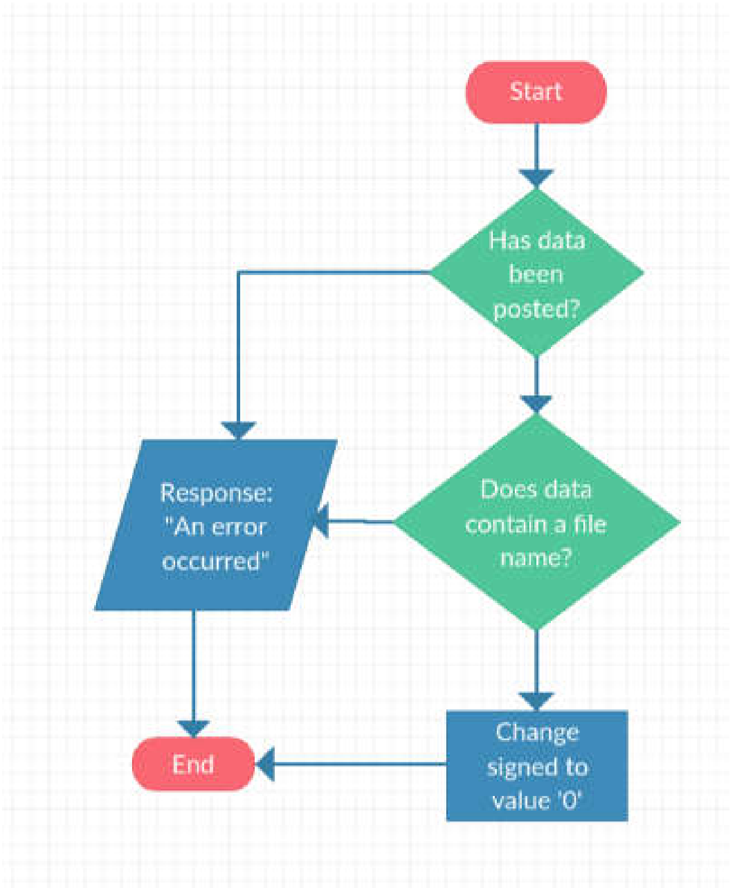

Start
Has data been posted?
Does data contain a file name?
Response: "An error occurred"
Change signed to value '0'
End

Figure 12: Flowchart of decline.php

When the users sign the document and accept their signature, the signature will be uploaded as previously mentioned. But another script called sign.php will be called as well.

This script works in the same way as decline.php, but will set the signed value to ‘1’. The value ‘1’ means that the document has already been signed.

4.2.8 Webpage logout

When the administrator users are done with their work on the website, they have the possibility to log out. They can do this by pressing the logout button which will call the logout.php script. This script will set the admin value to false and redirect them to the index page. After that, the users have to log back in if they want to access certain pages.

```
<?php

session_start();
$_SESSION['admin'] = 'false';
header("Location: http://rea.eca-simm.uva.es/projectarne/index.html");

 ?>
```

5 General description of the product

The application itself will be described here. Firstly I will explain the software and programming languages I have used to make this application. Then I will use a flowchart describing the entire lifecycle of the application. After that I will explain the layout of the application followed by an explanation of the code used.

5.1 Programming languages and tools

First I will briefly explain the software tools I have used and then the programming languages in which the application has been written.

5.1.1 Android Studio

The Android application was created using Android Studio. Android Studio is an IDE (Integrated Development Environment) based on the intelliJ platform. It is available for Windows, Mac OS X and Linux, and has replaced Eclipse Android Development Tools as Google's primary IDE for native Android application development. This software is the official IDE for Android and can be freely downloaded from the Android developers website. The SDK or Software Developer Kit is also needed to start developing.

5.1.2 Java (Android)

The Android programming language used is Java. The Java bytecode is not executed, instead the classes are compiled into a proprietary bytecode format and run on a virtual machine. Dalvik is a specialized virtual machine designed for Android, it is a register-based architecture, unlike Java VMs, which are stack machines. Since Android version 5.0 Dalvik was entirely replaced by Android RunTime, or ART. ART has two major advantages to Dalvik:

- Ahead-of-Time (AOT) compilation, which improves speed and reduces memory footprint
- Improved Garbage Collection (GC)

5.1.3 PHP

Hypertext Pre-processor or PHP is a server-side scripting language designed for web development. It is also used as a general-purpose programming language. In this project I have used some PHP scripts to include SQL injections. These scripts are stored on the server and executed on call. They will make the server connection with the database and perform actions on the tables.

5.1.4 MySQL

My Structured Query Language or MySQL is an open-source relational database management system. It is a popular choice of database for use in web applications, and is a central component of the widely used LAMP open-source web application software stack. In this application the server makes use of a MySQL Database to store the users and documents.

5.1.5 HTML

HyperText Markup Language or HTML is the standard markup language being used to create web pages. Web browsers can read these files and render them into web pages. It describes the structure of a website, this makes it a markup language rather than a programming language. I wrote the web pages for the bank's employees in HTML, supported by CSS.

5.1.6 CSS

Cascading Style Sheets or CSS is a style sheet language used for the presentation of a document written in a markup language, such as HTML. CSS is a cornerstone technology used by most websites to create more visually engaging web pages and user interfaces.

5.1.7 Server

The server used for hosting the web application is located at http://rea.eca-simm.uva.es . It's a server of the University of Valladolid and the Apache, PHP5 and MySQL services are installed on it. These services are needed to host this web application.

5.2 Complete application flowchart

5.2.1 Android

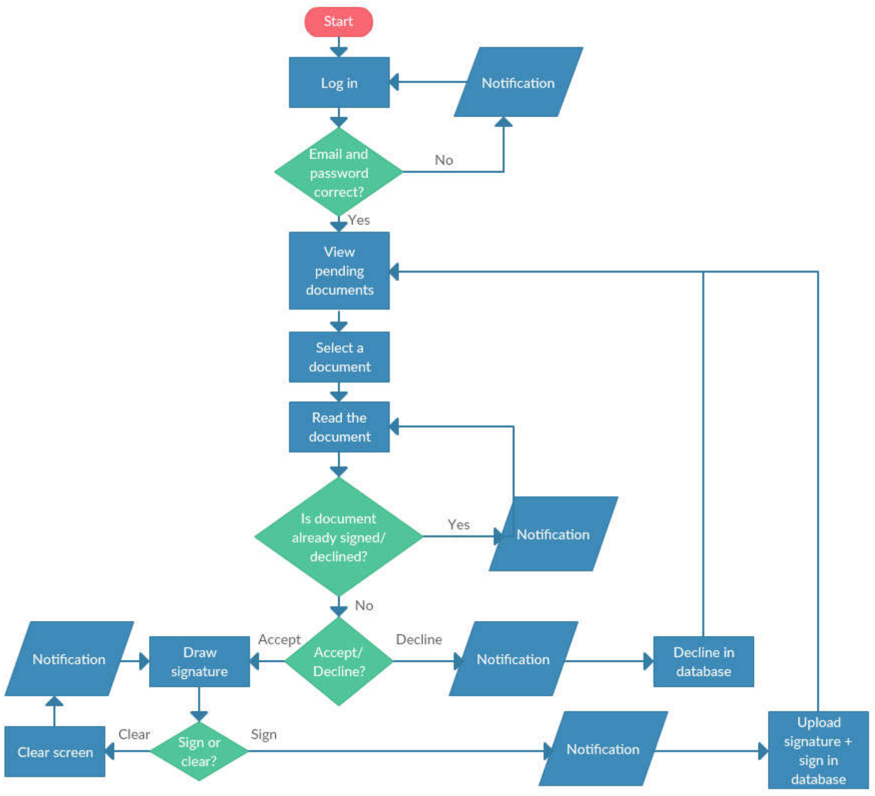


Figure 13: Flowchart Android application

As you can see from the flowchart in figure 13, the Android application boots to the login screen. Without registering or logging in, you can't see any documents.

When logging in, the application connects to the PHP script mentioned earlier which will connect to the database and verify the credentials. If the credentials are correct the users will get to see a list of their documents with their information.

The users can select a document and the application will take them to a screen in which they can view the document and respond to it. It is only possible to respond to a document when it has not been signed or declined yet. If the document has already been signed or declined the user will get a corresponding error message on his phone. When the users decline the document the application will call a script which will decline it in the database as well. Thereafter they are taken back to the list of documents.

When the users accept the document they will see a blank drawing board where they can draw their signature. If necessary the screen can be cleared so that the users can start drawing again.

If the signature is acceptable they can sign the document, the application will call a server-side script uploading the PNG file and accepting the document in the database. After this the users is taken to the list of documents.

When the users are done using the application they can close it. The next time the app will open they have to log in once again.

5.2.2 Web pages

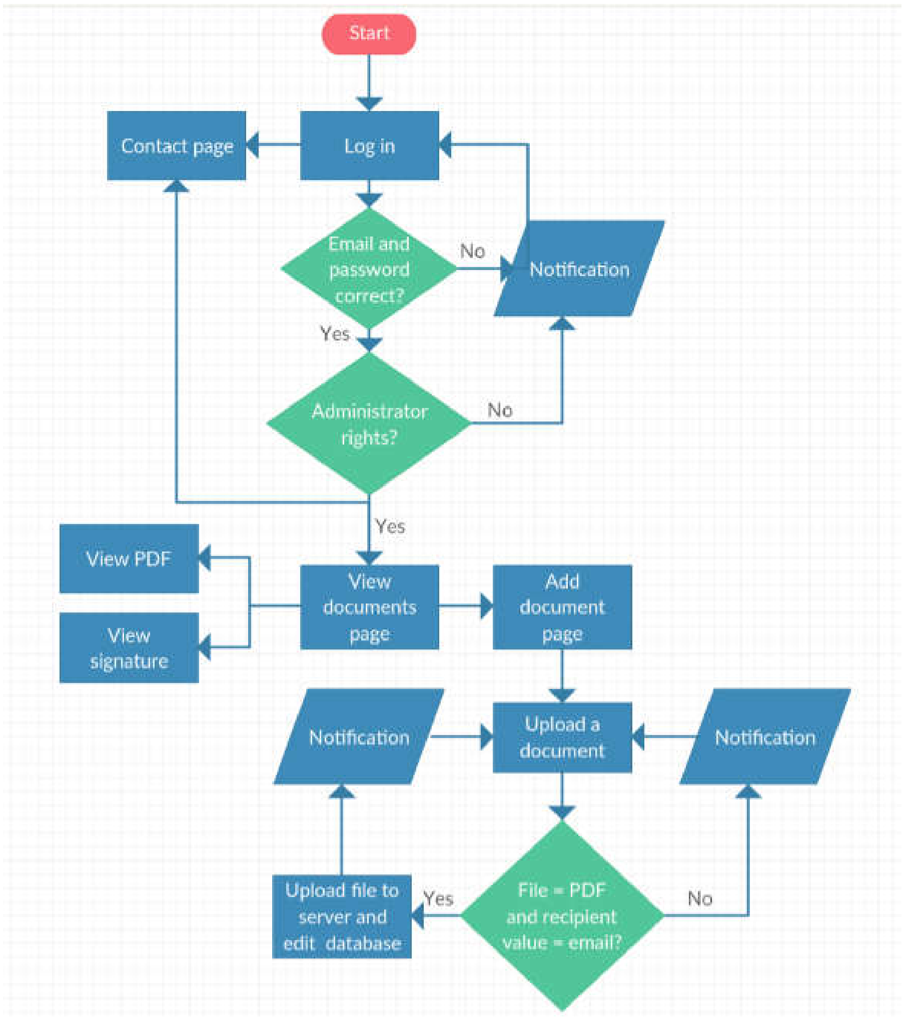


Figure 14: Flowchart Web pages

As seen from the flowchart in figure 14, the landing page doubles as a login page as well. The website can be used without logging in, but only the landing page and contact page are available.

When the users log in, a script will verify if they are indeed a registered user and if they have administrator rights or not. If the credentials are incorrect they will be redirected to an invalid login version of the index page. If the credentials are correct but the users do not have administrator rights, they will log in correctly but will not be able to access the pages for viewing and adding documents.

If the users have administrator rights they will be redirected to the page containing a table with all the documents and their information after logging in. There they have a clear overview of the documents and their status, thanks to the colouring of the table. If needed, the administrator can review the document and when signed, he can view the signature as well.

To upload new documents the administrator has to navigate to the 'upload documents' page, where he can fill out the form. After filling out the form a script will be called verifying the recipient value consists of an email and the uploaded file is a PDF file. If not, the users will get an error on the page telling them what went wrong and they can try again.

After uploading a correct document the webpage tells them their action was successful and gives them a quick link to viewing the document.

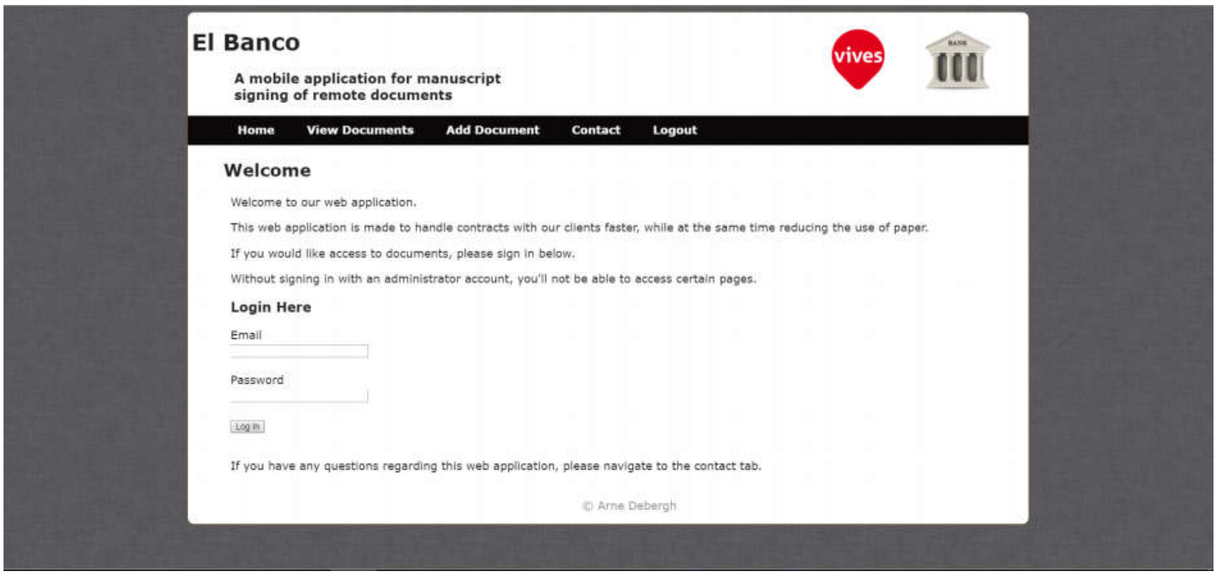
When the administrator is done he can log out using the logout button in the top right corner of every web page.

5.3 Webpage User Interface

All of the webpages shown in this chapter are written in HTML, CSS and PHP. If needed, the source code can be found on the disc provided at the back of the book.

5.3.1 Index

This webpage, as seen in figure 15, is the landing page for the website: http://rea.eca-simm.uva.es/projectarne/ . Here, the visitors can see a welcome message and log in when they want. Navigating to the view documents or add document page is not possible unless logging in with an account that has administrator rights.


El Banco
A mobile application for manuscript signing of remote documents
vives
BANK
Home
View Documents
Add Document
Contact
Logout

Welcome

Welcome to our web application.

This web application is made to handle contracts with our clients faster, while at the same time reducing the use of paper.

If you would like access to documents, please sign in below.

Without signing in with an administrator account, you'll not be able to access certain pages.

Login Here

Email

Password

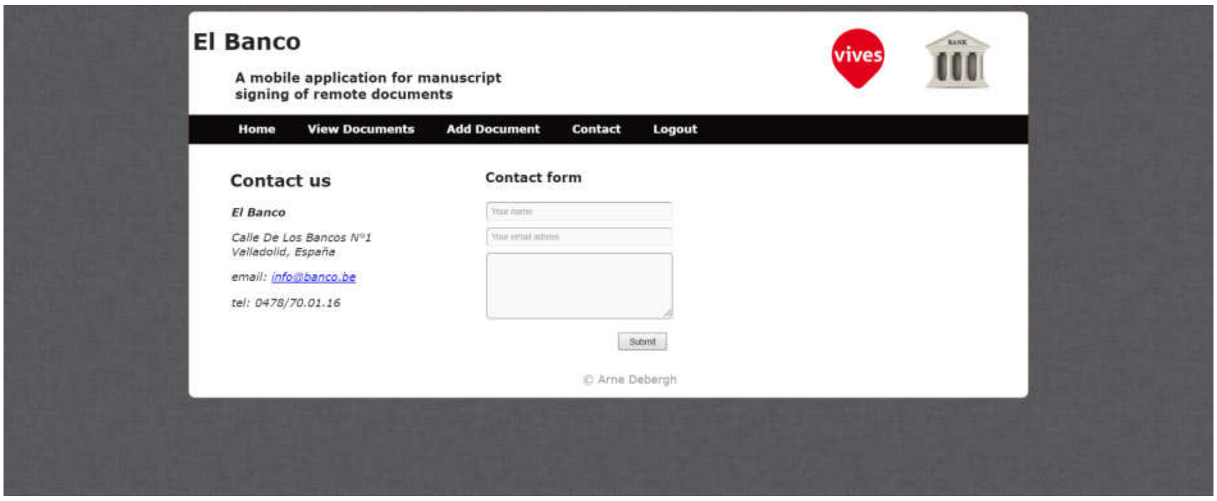
If you have any questions regarding this web application, please navigate to the contact tab.

© Arne Debergh

Figure 15: Index webpage

5.3.2 Contact

The contact webpage, as seen in figure 16, is a page everyone can visit, even when not logged in. The visitors can view contact information about the bank and fill out a contact form if needed.



Contact us

El Banco

Calle De Los Bancos N°1
Valladolid, España

email: info@banco.be

tel: 0478/70.01.16

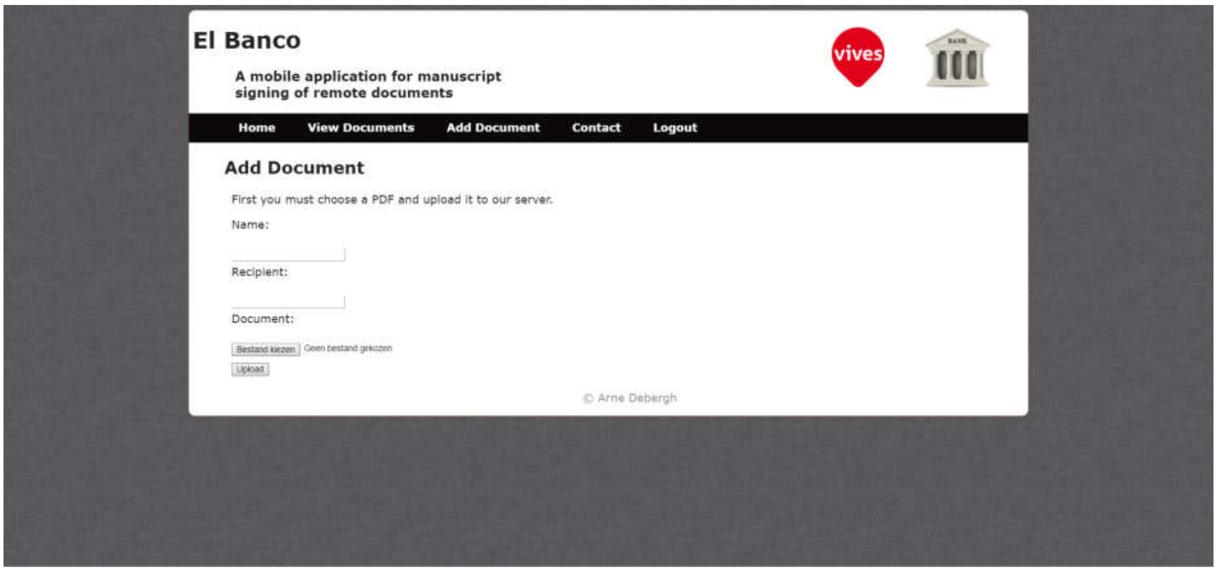
Contact form

© Arne Debergh

Figure 16: Contact Webpage

5.3.3 Adding Documents

On the add document webpage seen in figure 17, which can only be accessed by users with administrator rights, the users can upload a new document to the database. They will need to name the document, the recipients' email address and select a document in order to fill out the form correctly.



El Banco

A mobile application for manuscript signing of remote documents

Add Document

First you must choose a PDF and upload it to our server.

Name:

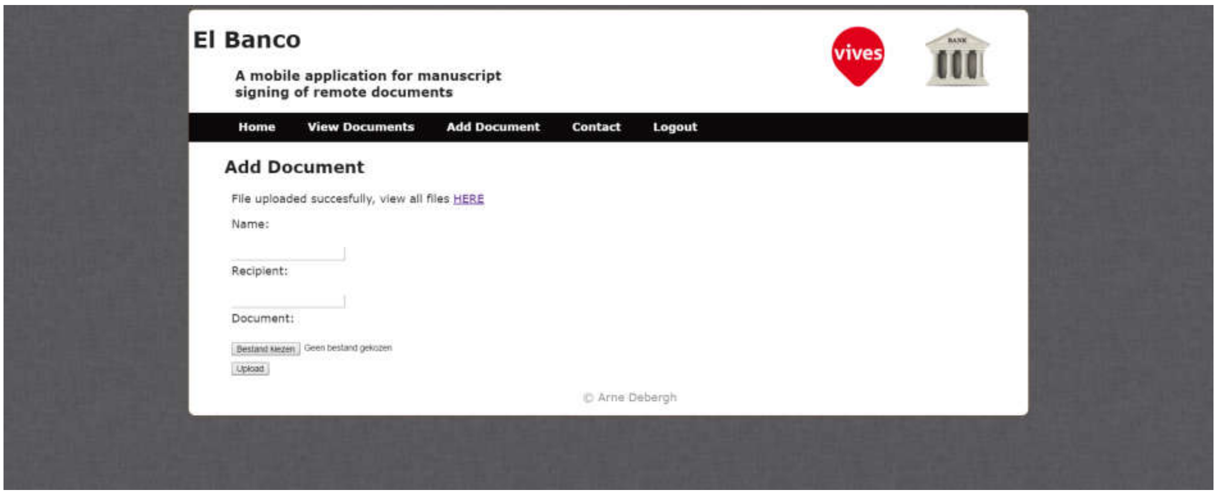
Recipient:

Document:

© Arne Debergh

Figure 17: Add Documents Webpage

After successfully uploading a document the page will refresh and show a link to the view documents webpage, as seen in figure 18:



El Banco

A mobile application for manuscript signing of remote documents

Add Document

File uploaded succesfully, view all files HERE

Name:

Recipient:

Document:

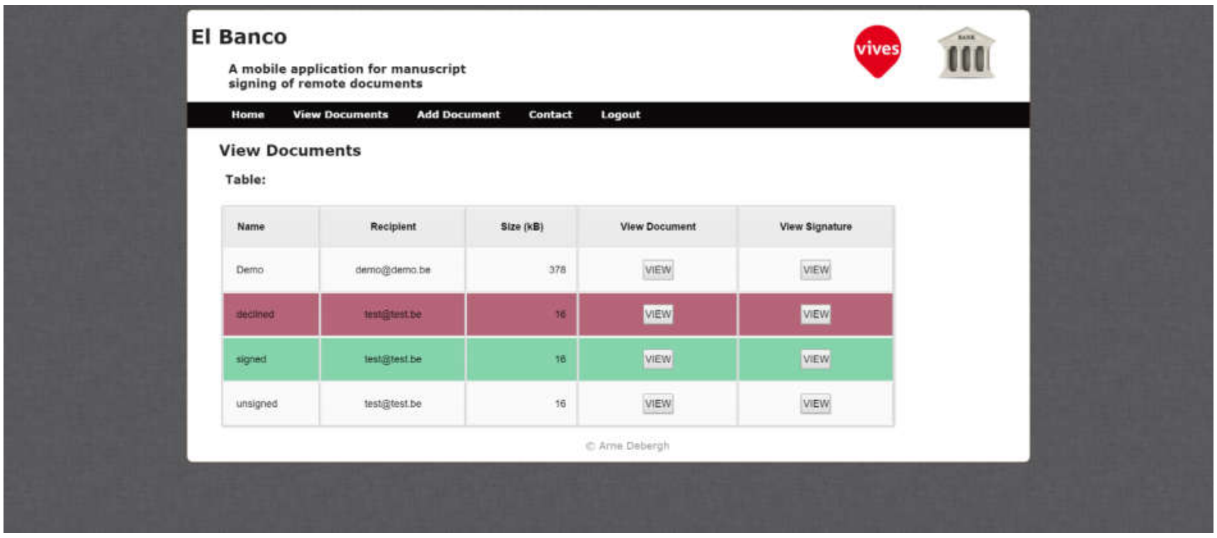
© Arne Debergh

Figure 18: Added Document Response

5.3.4 Viewing Documents

The webpage for viewing documents seen in figure 19 is also only accessible by logged in administrators. Over here, they can get a quick overview of all the documents with their name, the recipient and the size of the uploaded document in kB.

There are also 2 buttons included in the table, one for viewing the document and another for viewing the signature. If the document and signature would be merged later on, the second button column can be removed.


El Banco

A mobile application for manuscript signing of remote documents

View Documents

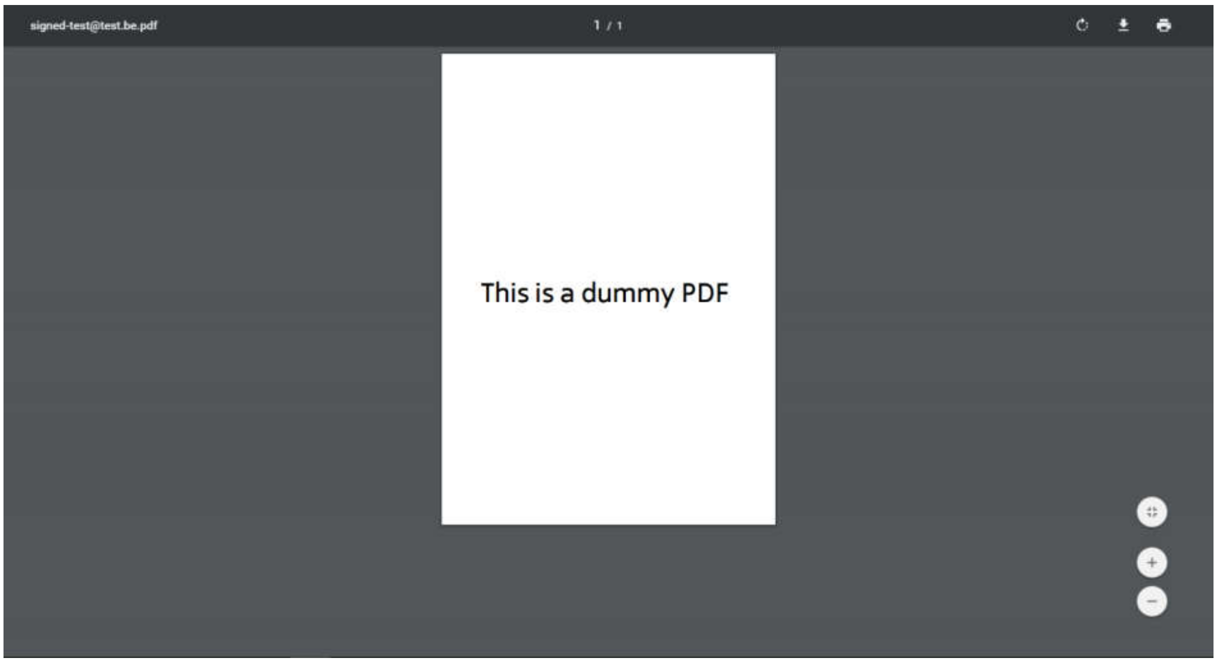
Table:

Name	Recipient	Size (kB)	View Document	View Signature
Demo	demo@demo.be	378	VIEW	VIEW
declined	test@test.be	16	VIEW	VIEW
signed	test@test.be	16	VIEW	VIEW
unsigned	test@test.be	16	VIEW	VIEW

© Arne Debergh

Figure 19: View Documents Webpage

When an administrator clicks on the button to view the document, a new tab will open showing him the document as seen in figure 20. The same happens when viewing a signature.



This is a dummy PDF

Figure 20: View Document In New Tab

5.4 Android User Interface

The entire Android application is using a relative layout. Relative layouts are able to place the elements in a structured manner. All of the XML files describing the layouts can be found in the Android source code on the disc provided at the back of the book.

5.4.1 Main Activity (Login)

When starting the Android application, the login screen is firstly seen. There are four text views, two edit text fields and one button. The text views are not editable and are used to provide the users with hints of use. The edit text fields are being used as inputs for the login activity and the button is used to confirm the login.

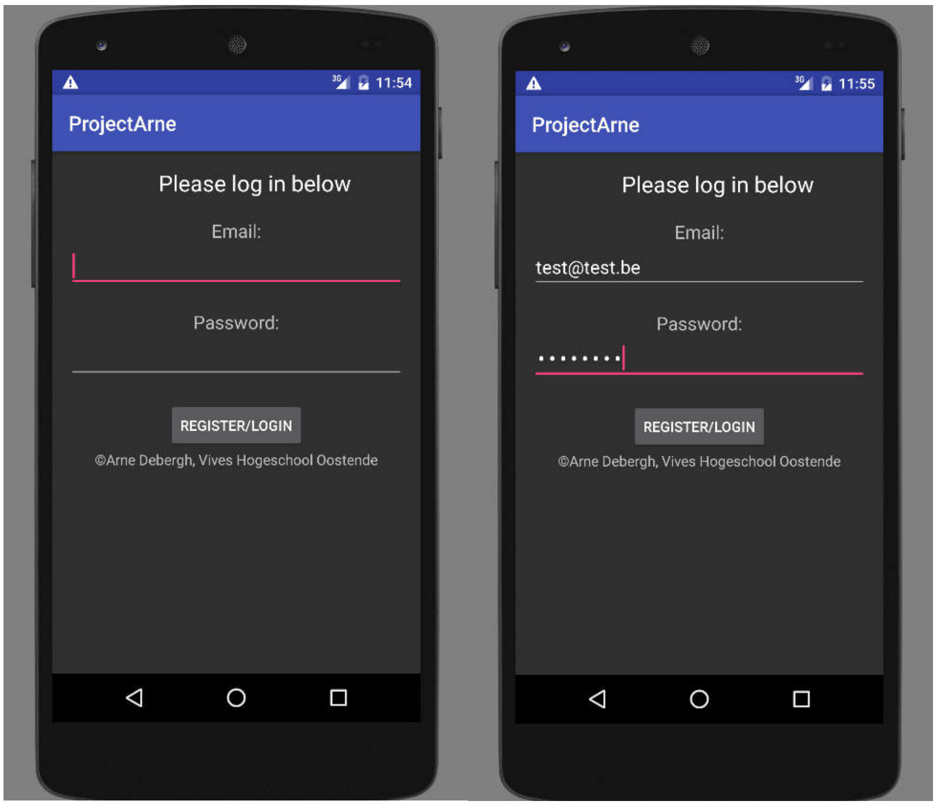


Figure 21: Screenshot of Login Activity

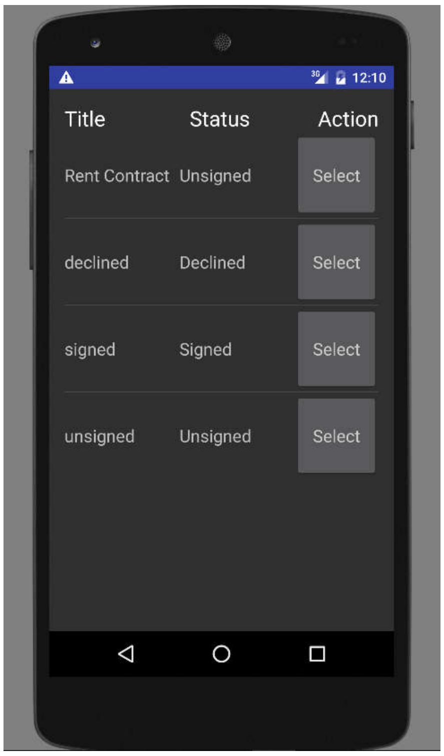
As seen in the figure 21, all of the elements in this activity are centered. This ensures the application will look the same on different devices. The password field will not show the entered characters because this information is confidential. When the smartphone makes use of keyboard software with suggestions, such as SwiftKey, there will be no input suggestions to ensure confidentiality.

After filling out the text fields, users can click the Register/Login button. If the email and password match they will be taken to the next activity.

5.4.2 List view Activity

When the users have successfully logged in, they get to see a table containing all of the documents to their name, see figure 22 as example. There is a column containing the title of the document, a column containing the status of the document and a column with a button, which can be clicked to select the document in that row.

The list view XML file only contains the top row containing Title, Status and Action. The actual list is being populated with row layouts defined in the row_layout.xml file. The document adapter class makes a row for each document and puts that row in the list view, until all rows are shown.



Title	Status	Action
Rent Contract	Unsigned	Select
declined	Declined	Select
signed	Signed	Select
unsigned	Unsigned	Select

Figure 22: Screenshot of List view Activity

When the users select a document by clicking the button in the row of that document, they will be taken to the next activity.

5.4.3 View Document Activity

In this activity, the users can read the selected document and respond to it. The main part of the layout consists of a web view, which displays the PDF file. On the bottom of the screen there are two buttons, one to sign and one to decline the selected document. This layout can be seen in figure 23.

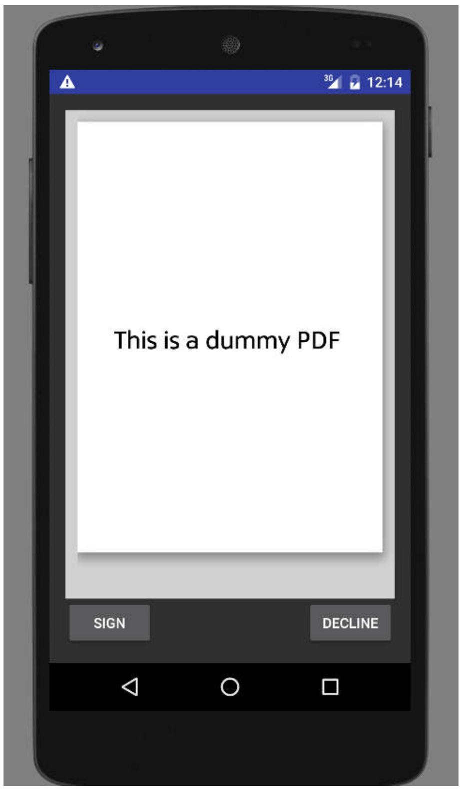


Figure 23: Screenshot of View Document Activity

When the document contains multiple pages, the users are able to swipe through them using the touchscreen. The users are able to return to the list view by pressing the back button of their smartphone. The buttons will perform certain actions. When the decline button is clicked, the document will be declined and the users will be taken back to the list view where they can see the new status. When the sign button is clicked the users will be taken to the signature activity where they can draw their signature and confirm the document.

5.4.4 Signature Activity

This activity has the exact same layout as the view document activity, but the web view has been replaced with a drawing screen. The users can draw their signature with the touchscreen and perform actions on it with the two buttons. This activity can be seen in figure 24.

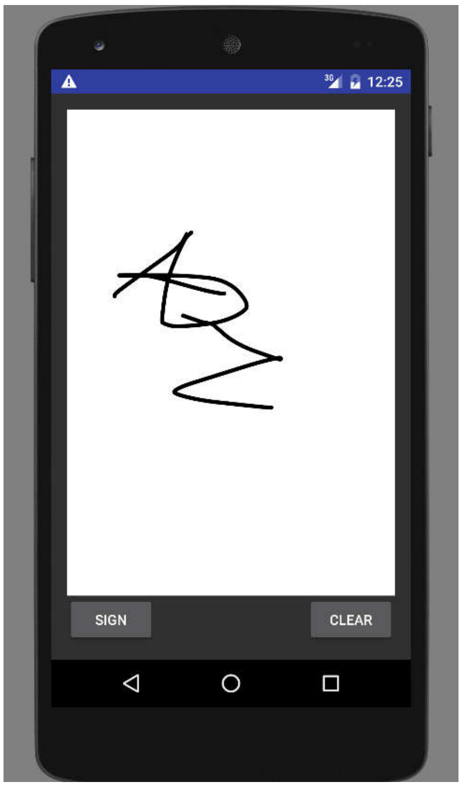


Figure 24: Screenshot of Signature Activity

The button on the bottom left corner will call a script confirming the document as well as uploading the signature to the database.

To clear the screen and start drawing again, the user can press the button on the bottom right corner. If the users haven't drawn anything for 500ms, the screen will also clear upon touching it again. This prevents users trying to slowly forge someone else's signature.

5.5 Webpage code

In this chapter I will explain the code of the webpages written in HTML and CSS. The PHP scripts have already been explained in chapter 4.2. I will insert some code in this chapter but not all of it. If a reference is made to pieces of code but it has not been inserted, the file name and lines of the code will be mentioned.

5.5.1 General features

5.5.1.1 Page

The head element of all pages contain the title of the page, a link to the styles.css file and some metadata about the page. All of the content of the webpages will be placed within a ‘div’ element of the class page. Implementing this method makes the webpages a little more attractive because it will look more or less as a piece of paper on top of a desk or board, depending of the background used. The page will always take up 70% of the total width and is centered. This class can be found in the styles.css file on lines 266-276. The background of the page is taken from an image and repeated in the body class in the styles.css file on lines 1-6. Each page also includes a header and footer.

5.5.1.2 Header and footer

The header and footer files are included in each individual webpage. The header contains the name of the bank, the title of this project, two images and the navigation bar.

```
<img unselectable="on"  class="bank" src="img/banco.jpg" alt="Company Logo"</img>
<img unselectable="on"  class="vives" src="img/vives.png" alt="Vives Logo"</img>

<h1>El Banco</h1>

<h3 class="top">A mobile application for manuscript <br>signing of remote documents</h3>
```

As seen in the above code, the images have been made unselectable. They each have their own class referencing to the styles.css file which will give them certain behaviour. The name of the bank is a classic ‘h1’ and the title of the project a ‘h3’ with its own CSS class.

The shared ‘vives’ and ‘bank’ class arguments are:

```
float: right;
padding: 1px;
margin-right: 40px;
user-drag: none;
user-select: none;
-moz-user-select: none;
-webkit-user-drag: none;
-webkit-user-select: none;
-ms-user-select: none;
```

These arguments will make sure the images are in the top right corner of the webpage, as well as not being draggable or selectable. The only argument in the project title head class is ‘margin-left: 5px;’ , which leaves a little space to the left of the title.

For the navigation bar I used a HTML ‘nav’ element containing a unordered list or ‘ul’ element. This contains ‘li’ elements of the class menu which will refer to other webpages or scripts on the server through ‘a’ elements.

```
<nav>

  <ul class="menu">

    <li class="menu"><a class="menu" href="index.html">Home</a></th>
    <li class="menu"><a class="menu" href="documentsCheck.php">View Documents</a></th>
    <li class="menu"><a class="menu" href="addDocumentCheck.php">Add Document</a></th>
    <li class="menu"><a class="menu" href="contact.html">Contact</a></th>
    <li class="menu"><a class="menu" href="logout.php" >Logout</a></th>

  </ul>

</nav>
```

The CSS a.menu class will defines the behaviour of the text displayed, such as colour and weight of the font. It also changes the colour of the text when users hover over it. These classes can be found in the styles.css file on line 11-18. The ul.menu and li.menu classes describe the look of the navigation bar, such as background colour, and can be found on line 29-42.

The footer file only contains one unselectable 'p' element of the footer class (line 277) declaring a copyright.

5.5.2 Index page

The index page displays a small welcome message for the users which also points at the login form below it. This login form uses a 'POST' method and will post the entered credentials to the webuser_control.php script explained in chapter 4.2.2. The source file of this page is index.html.

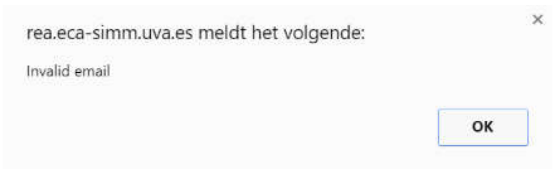
5.5.3 View documents page

The view documents page contains a table taking up 80% of the width of the page, it will display as many rows as there currently are documents. This is done by injecting some PHP lines into the HTML code that will use a query and a while loop to display rows of documents, as explained in chapter 4.2.4. To view the source file of this page look into documents.php.

5.5.4 Add documents page

Upon visiting this webpage, the users will only get to see a upload form. This form uses a 'POST' method to post the entered variables to the upload.php script, explained in chapter 4.2.3. The HTML code in this file also has been injected with PHP to change its look after the users posted the form. Depending on the response of the script, the webpage will show some extra information regarding the status of the posted form. This status can either be 'successful', 'unsuccessful' or 'invalid email'.

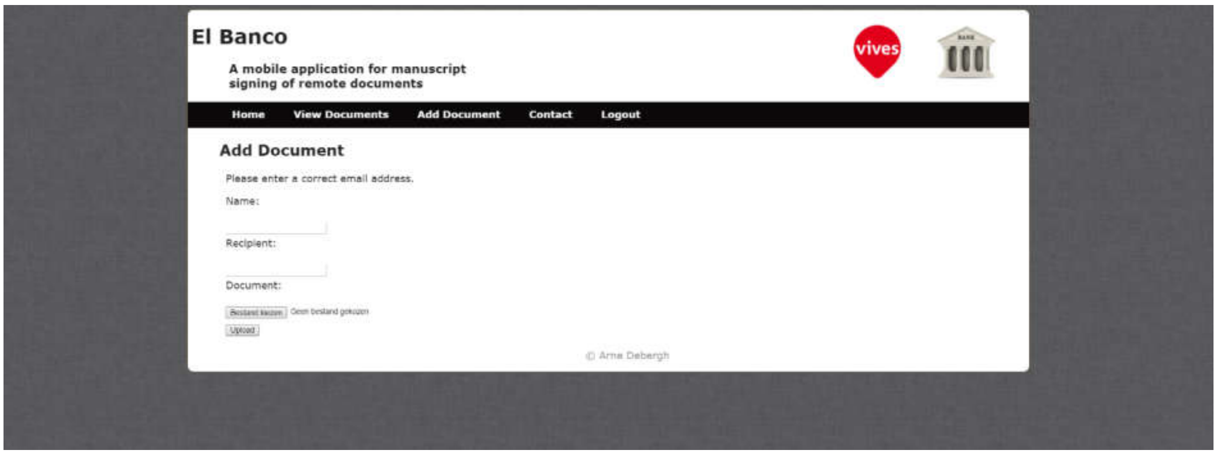
If we use 'John' as recipient, not being a valid email address, the response of the script will be 'invalid email'. Therefore we will see an error popup, as shown in figure 25.


rea.eca-simm.uva.es meldt het volgende:

Invalid email

Figure 25: Invalid email popup

After clicking the OK button, the page will refresh and the users get to see the same page but with a personal message giving them some advice, as seen in figure 26.


El Banco

A mobile application for manuscript signing of remote documents

Add Document

Please enter a correct email address.

Name:

Recipient:

Document:

© Arne Debergh

Figure 26: Add document email error

5.6 Android Code

In this chapter I will explain the written Android classes, supported by the UML presentation I will try to make the code more understandable than when looking at the source code. If needed, the source code can be found on the disc provided at the end of this book.

5.6.1 MainActivity

The MainActivity class, with its UML diagram shown in figure 27, is the class being used to generate the login screen. It has two methods: onCreate and openHome.

onCreate is the method being called when starting the login screen. It will set the content and layout according to the activity_main.xml file and will connect the edittexts and button to the public variables.

openHome is the method being used when the user clicks the button, it will take content of the edittexts and put them in strings, after that it will call upon the PostClass and start the ListViewActivity if the login was successful.

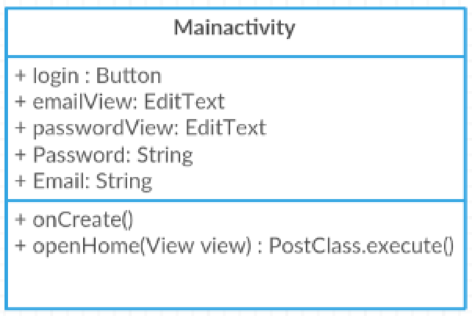


Figure 27: MainActivity class

5.6.2 PostClass

The PostClass is a class being used every time the application connects to the server and/or database. The UML diagram of this class can be seen in figure 28. It has a doInBackground() method that will open a connection to a given URL and post the parameters to it. This method will set the methods and properties of the connection and create a DataOutputStream to the URL. It will also retrieve the response of the script that has been called with the getResponseCode() method.

All of the output will then be placed in a two StringBuilders, one being placed in the logs, the other to create toast messages and provide values to the application when needed.

For example when the MainActivity has called the PostClass, the toast message being generated will be dependent of the response of the script. It can be a ‘welcome existing user’, ‘welcome new user’, ‘wrong password’ or ‘invalid email address’ response.

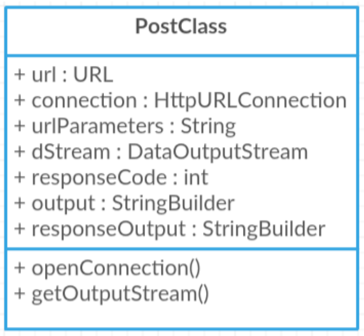


Figure 28: PostClass

5.6.3 ListviewActivity

The listviewActivity class has two methods: onCreate() and viewListItem(). It also has an intern class called BackgroundTask. The UML diagram of this class can be seen in figure 29.

The onCreate method will set the content view according to the activity_listview.xml file. It will also get the user's email address out of the login activity and place it in a string called userString. It will create a new listView and documentAdapter and will set the the documentAdapter to populate the listView. Afterwards it will call on the BackgroundTask, populating the listView.

The viewListItem method is used when the user has clicked a select button, it will get the information from the selected document and put them in strings. Then it will start the ViewDocsActivity and with it the document name and status will be sent.

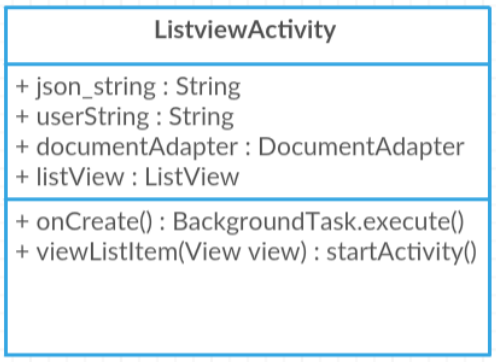

ListviewActivity
+ json_string : String
+ userString : String
+ documentAdapter : DocumentAdapter
+ listView : ListView
+ onCreate() : BackgroundTask.execute()
+ viewListItem(View view) : startActivity()

Figure 29: ListviewActivity

The BackgroundTask class, as seen in figure 30, will call upon the PostClass and start a connection, the response of the called script will be a JSON string. This string will be dissected and put into a stringbuilder.

When the stringbuilder is filled, it will use a while loop to take every document and put all of its' values in strings and afterwards in a document of the Documents class. This document will then be added to the documentadapter. All of these steps will be repeated with the while loop until all documents have been placed into the adapter.

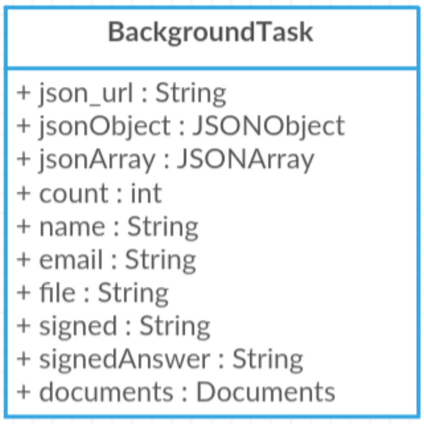


Figure 30: BackgroundTask

5.6.4 Documents

The documents class describes the architecture of the documents used. This class will be called when documents are being put into a list by the DocumentAdapter to populate the ListView. A UML diagram of the class can be seen in figure 31.

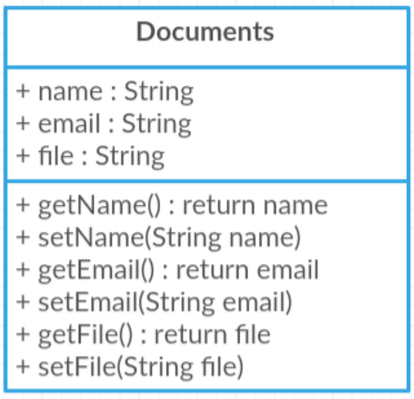


Figure 31: Documents

The class contains 3 strings: name, email and file, which describe the corresponding information of the document. Each string has a setter and a getter, used to set or get the values of these strings when needed.

5.6.5 DocumentAdapter

The DocumentAdapter class describes the adapter that has been written to place all needed documents into a list that will later on populate the ListView. The UML diagram of the DocumentAdapter class can be seen in figure 32.

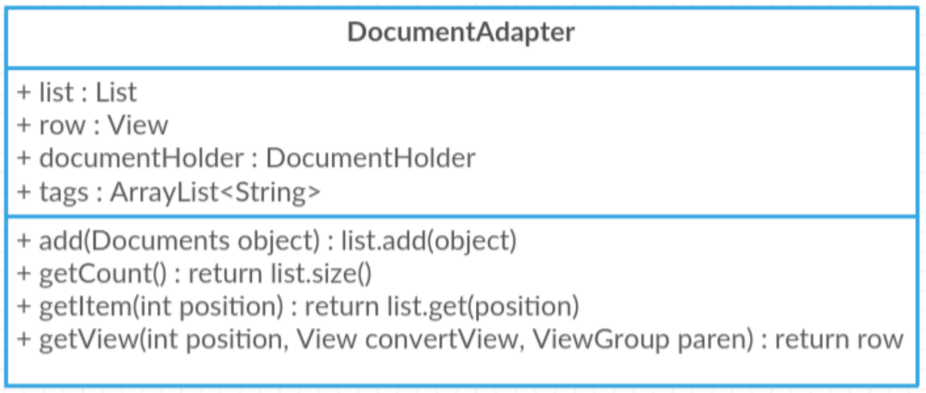

DocumentAdapter
+ list : List
+ row : View
+ documentHolder : DocumentHolder
+ tags : ArrayList<String>
+ add(Documents object) : list.add(object)
+ getCount() : return list.size()
+ getItem(int position) : return list.get(position)
+ getView(int position, View convertView, ViewGroup paren) : return row

Figure 32: DocumentAdapter

The adapter has a List that will contain all documents and a a row that will display the view. It also has a DocumentHolder which will hold all the information of the document and a ArrayList of tags containing extra information.

The add method will add the current document to the list when called. getCount will return the size of the list at any given moment. The getItem method will return a document at the given position in the list. The getView method will generate a personalized list to display in the ListView. It uses a LayoutInflater to inflate the row with all documents. If a document has been selected the information will be put in the DocumentHolder and the tags containing the status of the document will be set.

5.6.6 ViewDocsActivity

The ViewDocsActivity is the activity displaying the document that has been selected from the ListView. The UML diagram of the class can be seen in figure 33. It has 3 variables: document, status and browser. The document and status strings will be set in the onCreate method. The values of these strings will have been put as extras when the intent started this activity. The onCreate method will also set the browser WebView as new WebViewClient and load the URL of the document selected.

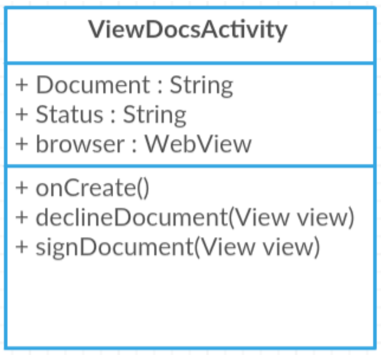


Figure 33: ViewDocsActivity

The declineDocument method will call on the PostClass which will update the database. It will start the listviewActivity once again after the document has been successfully declined. When the document has already been responded to it will generate a corresponding toast message.

The signDocument method will firstly verify if the document hasn't been responded to yet. If it has been, a corresponding error toast message will be generated. If not, the method will start the SignatureActivity and put the document string as an extra.

5.6.7 SignatureActivity

The SignatureActivity class will be called after the user has decided to accept a document. It contains a document string, containing the name of the document. When the activity is started, the onCreate method will set the document string from the extras that have been given through the intent. It will also set the uploadUri string, referring to the uploadSig.php script on the server and make a temporary saving path called signaturePath. The UML diagram of the class can be seen in figure 34.

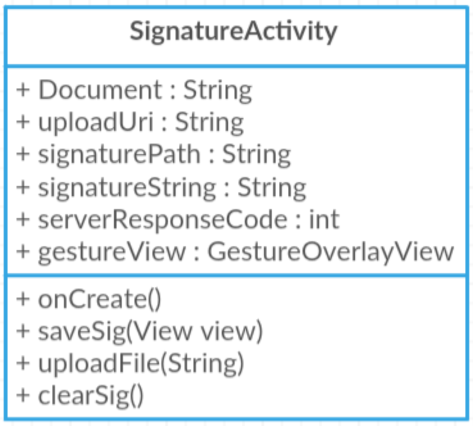


Figure 34: SignatureActivity

When the saveSig method is called, after clicking the 'Sign' button, the drawing will be captured and saved temporarily to the signaturePath. Then it will call the uploadFile method uploading the recently saved signature to the server. The uploadFile method will the signature to the server through the PostClass. The serverResponseCode tells the application if the action was a success and if so, the local signature is removed.

The clearSig method is called when the user clicked the 'Clear' button. It will clear the gestureView and refresh it so that the user can draw his signature once again.

5.6.8 Manifest file

The manifest file contains the permissions and settings of the application. In this project, it states that the minimum SDK version must be 16 and the target version is 23. It also states that this application needs internet, access network state, read phone state, write external storage and read external storage permission. This file was also used to change the launcher icon of the application.

6 User and system requirements

During this chapter I will explain the needed factors for both the users and the company to be able to use this application. I will firstly cover the smartphone requirements and then the server requirements.

6.1 User requirements

6.1.1 Android end user

The end users should be in possession of an Android smartphone meeting the minimum required specifications stated on the next page. They should have an email address and a password to use as well as the capability of reading the document and drawing their signature. The use of this application needs an Internet connection, this can be done with mobile data or Wi-Fi connection. The application can be used almost everywhere because mobile data connections have significantly improved over the last decade. Nowadays you can connect almost everywhere to a (free) Wi-Fi network.

6.1.2 Webpage admin user

The admin users using the webpages should be in possession of an administrator account. They will need a personal computer with internet connection and recent browser.

6.2 System requirements

6.2.1 Minimum required specifications

Please keep in mind that these are the minimum requirements for having a pleasant experience with the application.

The minimum processor speed should be 1GHZ. Most of the smartphones being produced right now are more powerful so normally this will not prove to be a problem.

The graphics processing unit should have a clock speed of 200 MHz. The minimum RAM requirement is 512MB. A few years ago 512MB RAM was quite a lot for a smartphone, but in the past years this has become a standard value.

The smartphone needs to have an internet connection through its mobile data or a Wi-Fi network. When using mobile data, a 3G connection or higher is recommended. For Wi-Fi, the connection speed should be at least 54Mbps, being a G or A connection. Since 2013 all smartphones have a Wi-Fi chip capable of connecting to B/G/N networks.

The smartphone also needs a touchscreen display. If this is not present, the application will be able to run but the user will not be able to draw his signature.

The minimum Android version should be 4.1 (Jelly Bean) or higher. Custom ROMs can make use of the application as well if they match this minimum version of Android.

6.2.2 Recommended specifications

The following specifications are recommended if the users want to have a smooth experience with the application.

The recommended processor is a Quad Core processor clocked at 1.4GHz or higher. The recommended GPU has more than 300 MHz. The recommended RAM is 1.5GB. The recommended mobile data connection is 4G (LTE).

The recommended Android version for this application is Android 5.0, on which the application has mainly been tested, or higher.

The application has been tested on a Samsung Galaxy S5 mini, which matches these specifications. During these tests there were no speed or rendering issues, therefore I recommend having these specifications.

6.2.3 Server Requirements

The server does not have any specific hardware requirements. If it is capable of hosting the web application and implementing a MySQL database there should not be a problem.

7 Planning

To make this final project and write the thesis I had a timespan of four months. In February I started working on the project and by the 27th of May I had to present my work. During these four months I had to prepare, do research, develop and test the application as well is preparing my defence.

7.1 February

During this month I mainly have been doing research as well as planning out the steps to complete the final project. I looked into various websites to refresh my knowledge of Android programming, as well as PHP and web development.

I reviewed some of the exercises I've made throughout the past 3 years. I also started creating a basic Android application and webpages which would be used as the foundation of my project.

Afterwards I spent a lot of time trying to draw and save a signature locally on the Android device. When looking back at this it actually wasn't a lot of work to do, but the research and trial-and-error method took up a lot of the time.

For the Android application, I also implemented a way of viewing PDF documents, which was used later on.

For the web pages I implemented the uploading of documents, without being linked to the database. I did set up the database in February but later on this part has totally changed.

7.2 March

In March I created the login activity for Android, as well as the receiving of the documents. I created a simple and fast way to log in or register for the users. The activity was tested in a separate Android Studio project and implemented to the main project when it was working.

The biggest part of my time this month went to setting up the database and being able to connect the Android application with the server and database. It proved to be one of the hardest parts of this application and took up quite some time.

Writing the PHP scripts was hard at first but once I got the hang of it, writing PHP lines went pretty fluently.

For the web application I implemented the log in, using PHP scripting as well. The uploading of documents was reviewed since I now had a decent connection with the database. After being able to upload documents I also made the view documents webpage containing a table displaying the necessary information.

I also changed a lot of the design of the web pages using CSS. This was a fun part because I could see the pages change from ordinary text and forms to pages with a certain design and character, making the entire website more attractive.

7.3 April

In April, I made the list view of documents for the Android application. This needed a couple of different elements working together, which wasn't easy. I set up a script sending all the documents and their information in a JSON string to the android application, which would take out all the documents of the logged in user and display them to a listview table.

To view the selected documents I ran into some problems because I couldn't get the standard listview item onclick method to work. So I set up a column containing a button which contained a tag identifying the document. I wanted to place two tags into the buttons but this was not possible, so I had to make an arraylist of two strings and give that as tag to the button. In the view document activity the strings are then taken out of the arraylist so that they can be used.

Responding to the document in a negative way went pretty good, but the uploading of the drawn signature as well proved to be a difficult element. All of the documentation I found online was using deprecated classes which are not supported by newer Android versions.

I had to invent my own way around these classes. At first I tried to convert the bitmap to a base64 string and then decode the string on the server, but this did not work out as planned. Eventually I was able to use my PostClass to just post the document into a script. Once again, I was looking for a solution in a difficult way which proved to be impossible, the actual solution was pretty simple in the end.

The web application has been optimized in April, I also started testing and debugging it. I encountered some errors with security and the uploading of documents but they were quickly fixed.

7.4 May

In May I started writing this documentation, documenting the whole project and remembering what and how was a lot harder than I thought beforehand.

If I could start over I would document a lot more while working on the project instead of doing it all afterwards.

I also continued the testing and debugging of the application while writing the documentation. When looking into some code while reviewing it, I found some parts to be badly written so I rewrote them. All the files were also given some commenting so that people reviewing the code will be able to understand it faster.

All code and documents concerning this project can be found on the disc provided at the back of the book.

8 Software Tests

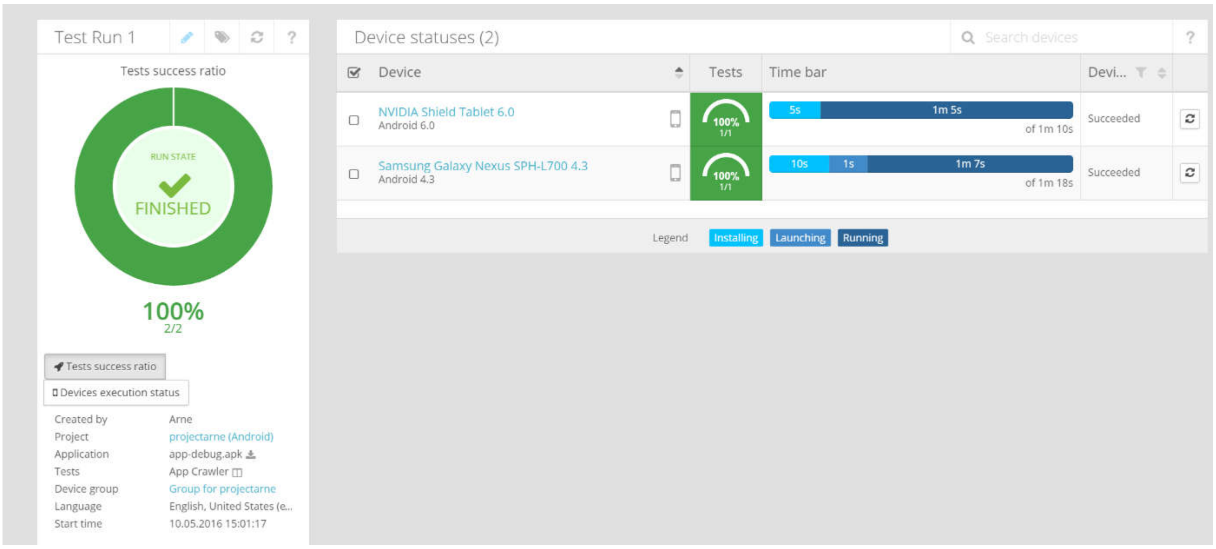
During the entire development of the project, everything was tested on the go. The database, webpages and scripts were constantly being hosted on a server of the University of Valladolid and tested while being edited.

The Android application was also elaborately tested on my own smartphone, running OxygenOS. I performed a lot of tests with different emulators in Android Studio, running different versions of Android, as well as a Samsung Galaxy S5 Mini, running Android version 5.1.

I have ran a test on cloud.testdroid.com, this website has tested my Android application on two devices:

- NVIDIA Shield Tablet, Android 6.0
- Samsung Galaxy Nexus SPH-L700, Android 4.3

As a free user, I was only able to test on these two Android devices. When you have a paid account, it’s possible to test it on almost every Android device you can think of. There were more free to use devices, but the Android versions of these were below 4.1 so they were not compatible.



Test Run 1

Tests success ratio

RUN STATE FINISHED

100%
2/2

Tests success ratio
Devices execution status

Created by	Arne
Project	projectarne (Android)
Application	app-debug.apk
Tests	App Crawler
Device group	Group for projectarne
Language	English, United States (e...
Start time	10.05.2016 15:01:17

Device statuses (2)

Device	Tests	Time bar	Devi...
NVIDIA Shield Tablet 6.0 Android 6.0	100% 1/1	5s · 1m 5s of 1m 10s	Succeeded
Samsung Galaxy Nexus SPH-L700 4.3 Android 4.3	100% 1/1	10s · 1s · 1m 7s of 1m 18s	Succeeded

Legend: Installing, Launching, Running

Figure 35: Testdroid Software Test (3)

In figure 35, we can see the time that was needed for installing and launching the application on the devices. The runtime is being displayed as well. The performance and error logs of these tests can be found on the disc provided at the end of the book.

9 Android User Manual

During this chapter detailed instructions on how to use all of the functions of the application will be given. In each step all the different possibilities will be explained.

9.1 Open the application launcher and select the app

Select the application launcher on your smartphone. Search for the icon being displayed in figure 36 and select it.

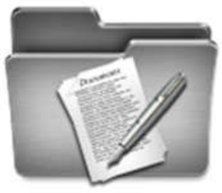


Figure 36: Application Icon (4)

The application will now open and show you the login screen. There are 2 fields to that can be filled out: 'Email' and 'Password'. There is also a button that can be clicked to try logging in or registering.

9.2 Log in

On the login screen seen in figure 37, you can enter your email address and password. After that you can click on the button. The application will then try to make a connection with the server. When your login was incorrect you will get to see a small toast message on the bottom of your screen, according to the error. This can be invalid email address, unmatching password or server not found. If the server has not been found please contact the developer.

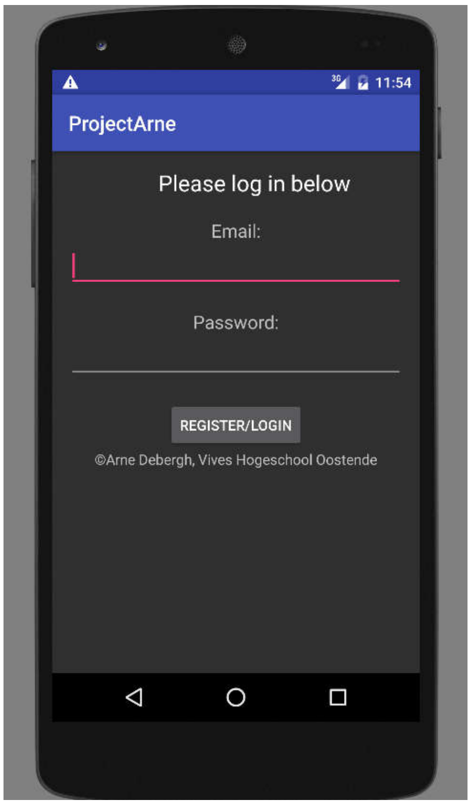
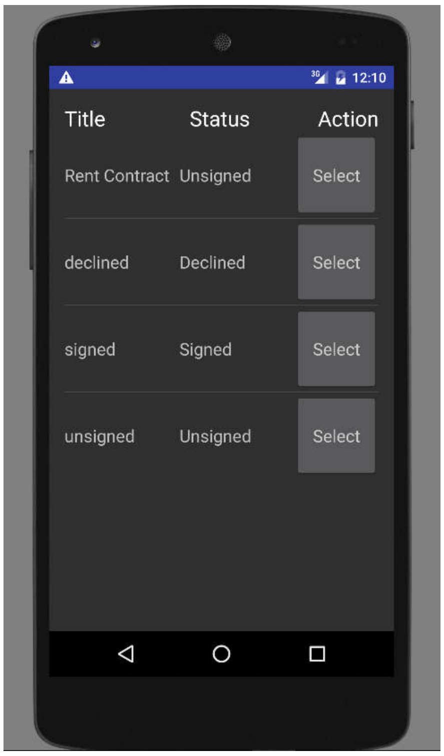


Figure 37: Log in

9.3 Document overview

After logging in with the correct credentials, you get to see an overview of your documents. The table shown will display a row per document, with its title, status and a response button. This screen can be seen in figure 38. If you want to select a document, please press the select button responding to the document. After doing this you can always go back to the overview of your documents by pressing the back button on your smartphone.



Title	Status	Action
Rent Contract	Unsigned	Select
declined	Declined	Select
signed	Signed	Select
unsigned	Unsigned	Select

Figure 38: Document overview

9.4 Reading and responding to a document

When you have selected a document, you will be able to read it on the screen that opened. This screen can be seen in figure 39. You are also able to scroll to the document. After reading the document, you can choose to respond to it by pressing the accept or decline button or you can choose to ignore it for now and go back to the overview of documents.

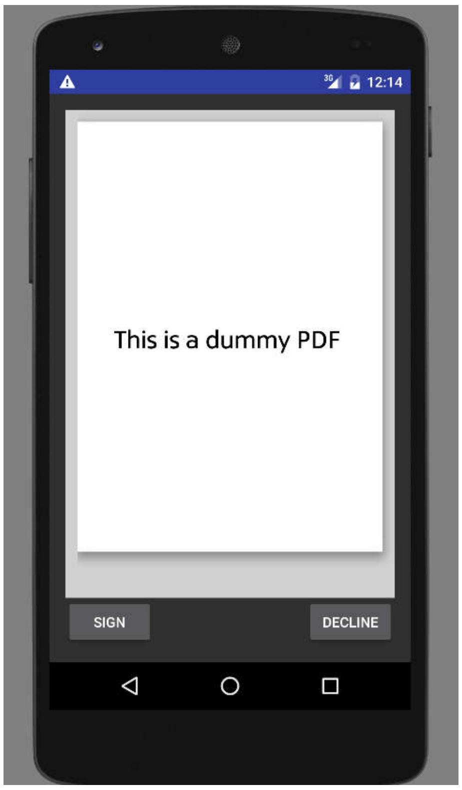


Figure 39: Reading and responding to a document

After declining the document you will be taken back to the document overview and will see that the new status of that document will be 'declined'. When you press the accept button a new screen will open where you can draw your signature. If the document has already been signed you will be unable to sign or decline the document. When clicking one of the buttons a toast message will appear telling you the document has already been signed or declined.

9.5 Drawing your signature

After accepting the document, a new screen will have opened. This screen can be seen in figure 40. You are able to draw your signature over there. If the drawn signature is incorrect, you can press the clear button to clear the drawing screen. If you wait a small time and press the drawing screen, it will also clear. When you are ready with drawing your signature, you can press the sign button and the document will be accepted, as well as your signature being uploaded. After signing the document you can go back to the document overview where the status of the document will have changed.

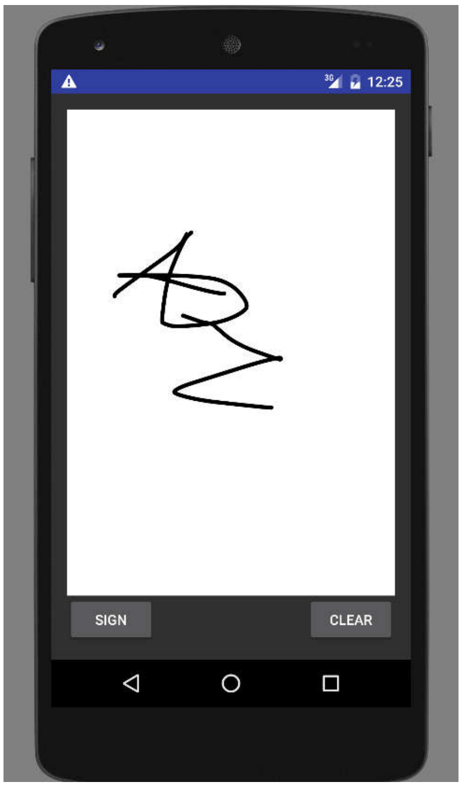


Figure 40: Drawing your signature

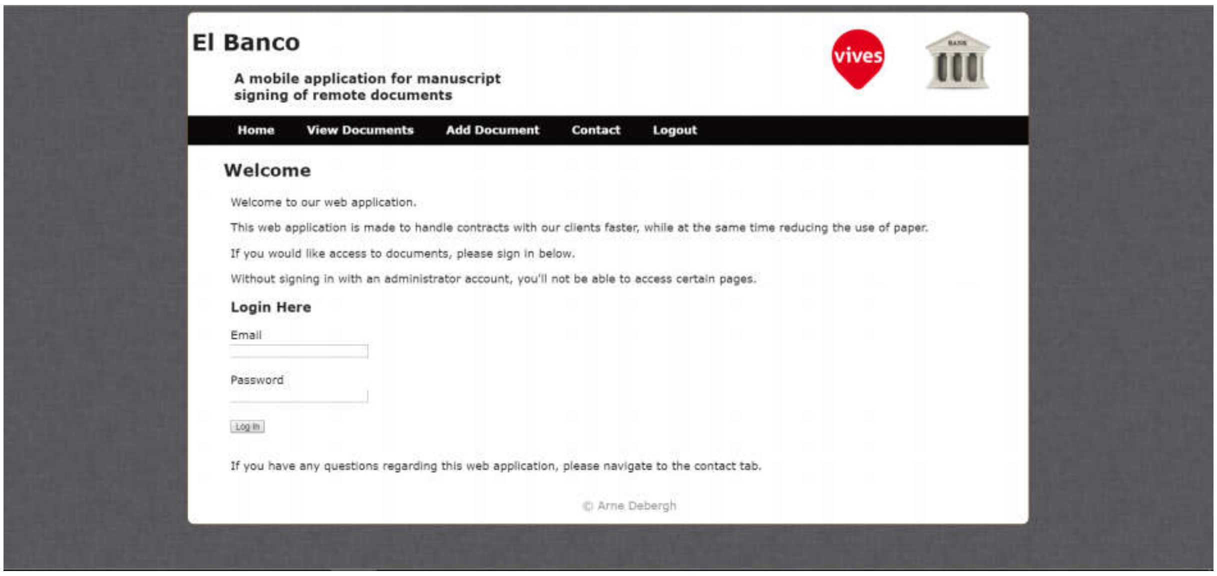
9.6 Logging out

When your work with the application is over, you can close the application as you would close any other application on your smartphone. The next time the application is started, you will have to log in again.

10 Webpages user manual

10.1 Navigating to the webpages

Administrators are able to upload new documents or view the status of all the documents on the application’s website. These webpages can be found at: http://rea.eca-simm.uva.es/projectarne/ . You have to log in on the homepage to be able to perform actions with documents. If not logged in, you can only view the home and contact webpages seen in figures 41 and 42.


El Banco
A mobile application for manuscript signing of remote documents
vives
Home
View Documents
Add Document
Contact
Logout

Welcome

Welcome to our web application.

This web application is made to handle contracts with our clients faster, while at the same time reducing the use of paper.

If you would like access to documents, please sign in below.

Without signing in with an administrator account, you'll not be able to access certain pages.

Login Here

Email

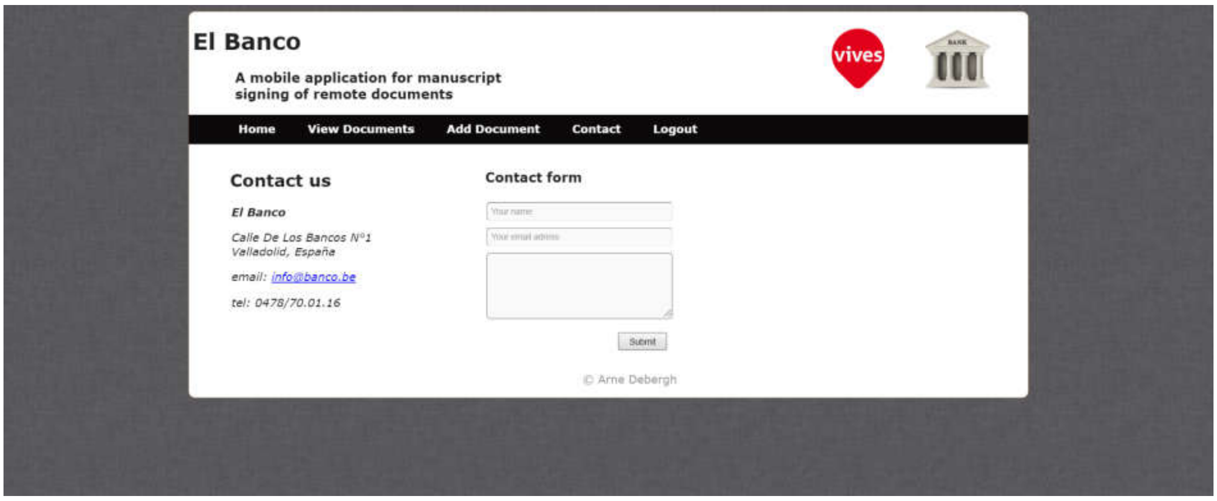
Password

Log in

If you have any questions regarding this web application, please navigate to the contact tab.

© Arne Debergh

Figure 41: Index Page


El Banco
A mobile application for manuscript signing of remote documents
vives
Home
View Documents
Add Document
Contact
Logout

Contact us

El Banco

Calle De Los Bancos Nº1
Valladolid, España

email: info@banco.be

tel: 0478/70.01.16

Contact form

Your name

Your email adress

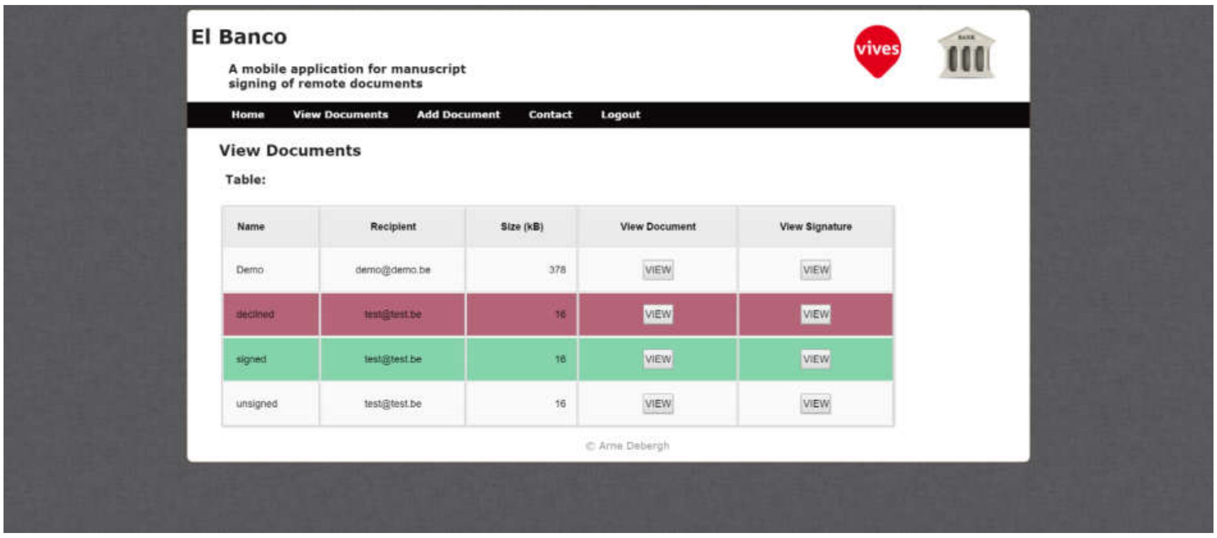
Submit

© Arne Debergh

Figure 42: Contact page

10.2 View documents

After logging in correctly, you will be redirected to the view documents webpage seen in figure 43. Over there you get to see a table containing all of the documents currently in the database. You can quickly see if the document has been responded to due to the background colour of the row. If signed, the row will appear green. If declined, it will appear red. When the document has not been responded to yet, the background of the row will appear white. You can quickly view the original document and/or signature by pressing the 'PDF' or 'PNG' button in the document row.


El Banco
A mobile application for manuscript signing of remote documents
vives
Home
View Documents
Add Document
Contact
Logout

View Documents

Table:

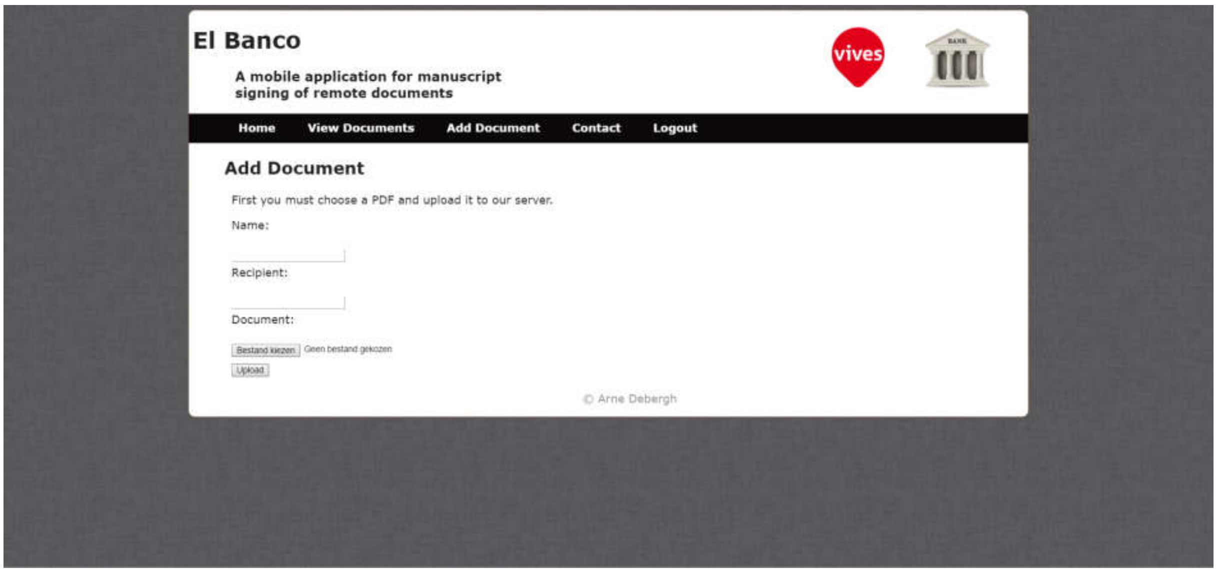
Name	Recipient	Size (kB)	View Document	View Signature
Demo	demo@demo.be	378	VIEW	VIEW
declined	test@test.be	16	VIEW	VIEW
signed	test@test.be	16	VIEW	VIEW
unsigned	test@test.be	16	VIEW	VIEW

© Arne Debergh

Figure 43: Viewing documents

10.3 Adding documents

When you want to send a new document to a client, you have to navigate to the add document page as seen in figure 44. Over there you have to fill out the form. To fill it out correctly, please enter a title, correct email address and PDF document.



Add Document

First you must choose a PDF and upload it to our server.

Name:

Recipient:

Document:

Geen bestand gekozen

© Arne Debergh

Figure 44: Adding documents

After uploading the new document you will see a link to the document overview page. If you click on it you will be redirected to the webpage.

10.4 Logging out

If you are finished working on the webpages, you can logout by pressing the logout link in the navigation bar. This will log the current user out and redirect you to the homepage.

11 Installation Manual

This chapter will explain how the Android application can be installed on your smartphone. This process will be explained step-by-step. It is not a difficult process since your smartphone will perform most of the work by itself.

11.1 Placing the .apk file on your computer

To retrieve the .apk file, you have to insert the disk, provided at the end of this book, in your computer and copy it to your computer. The file can be found in the APK folder.

11.2 Transfer the .apk file to your mobile phone

Please connect your mobile phone to your computer, and when prompted, select to use it as an external memory stick. Then you have to locate the earlier copied package and drag it to the mobile phone. Make sure you place it in a easy to be found location of your smartphone's internal memory.

11.3 Enable installation from unknown sources

This step is split up for Android versions lower than 5.0 and versions equal to or higher than 5.0.

11.3.1 Android 5.0 or newer

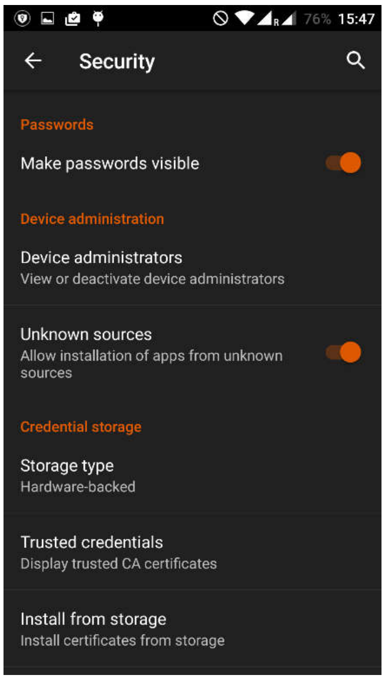
Open up the settings menu of your smartphone. This can be done by opening the application launcher and locating the settings menu, or by dragging down the notification bar twice and pushing on the gear wheel.

11.3.2 Android versions lower than 5.0

Open your application launcher and locate the settings menu. Press it and the settings menu will open up.

11.3.3 Enabling installation

This part is again equal for all versions of Android. After opening up the settings menu locate the 'security' tab. Under device administration you should see the option 'unknown sources – Allow installation of apps from sources other than the Play Store'. Activate this option as seen in figure 45.


76% 15:47

Security

Passwords

Make passwords visible

Device administration

Device administrators
View or deactivate device administrators

Unknown sources
Allow installation of apps from unknown sources

Credential storage

Storage type
Hardware-backed

Trusted credentials
Display trusted CA certificates

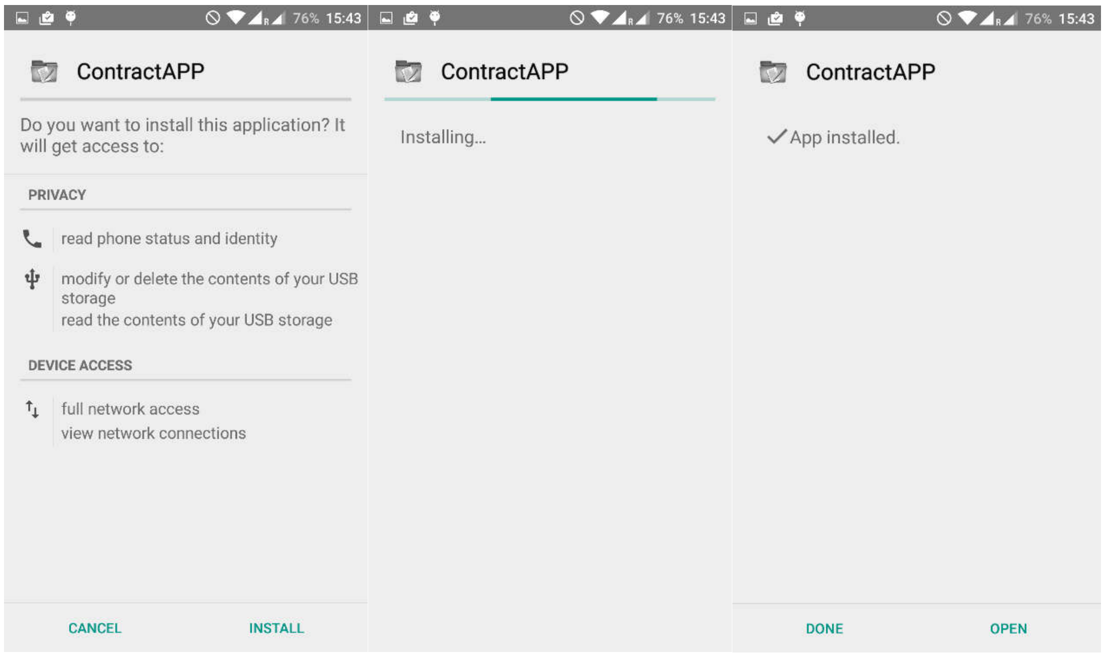
Install from storage
Install certificates from storage

Figure 45: Unknown Sources

11.4 Executing the package

Use a file explorer to locate the installation package (.apk file) you have previously copied to the internal memory of your device. If you do not have a native file explorer application, you will have to download one from the Play Store in order to explore the internal memory of your smartphone.

After locating the installation package on the phone click on it, this will open an installation dialog. Please read and accept the permissions the application uses. This process can be seen on figure 46.


76% 15:43

ContractAPP

Do you want to install this application? It will get access to:

PRIVACY

read phone status and identity

modify or delete the contents of your USB storage

read the contents of your USB storage

DEVICE ACCESS

full network access

view network connections

CANCEL INSTALL

76% 15:43

ContractAPP

Installing...

76% 15:43

ContractAPP

✓App installed.

DONE OPEN

Figure 46: Installing the application

When this is done, the application will be installed. Your application is now ready for use.

12 Conclusion

This application can become a handy tool for its users. It is able to save valuable time and therefore also money.

The application I've made has completed the asked objectives. It has a log in/ register system to implement a certain level of security. It implements the reading of the documents through connecting to the database, which saves space on the smartphone. This makes the application use a little bit more internet traffic but that is not a big problem nowadays.

I found this application a very interesting project since I had to use my basic coding knowledge from the courses of the previous years and combine them all together. I also learned a lot by doing online research, the used languages are very well documented. Android has a very open developer community and the developers website comes in very helpful. The use of PHP and MySQL was interesting, I have used them in a totally different way than before.

Some improvements could still be made to the application. These were not included since I didn't have the time to get them fully working. The biggest improvement in my opinion would be the merging of the original document with the signature.

If this application would be used by a company, I would change the layout of the Android application a bit to make it more attractive. Also the webpages would have to be personalized to the company.

These are possible improvements that can be added to this project. This application can, and hopefully will, become useful in a productive environment.

13 Bibliography

(1) International Data Corporation. (sd). *Smartphone OS Market Share.* Last checked on 05 11, 2016, from the IDC website: http://www.idc.com/prodserv/smartphone-os-market-share.jsp

(2) Android Developer. (sd). *Dashboards.* Last checked on 05 11, 2016, from the Android Developers website: http://developer.android.com/about/dashboards/index.html

(3) Testdroid. (sd). *Testdroid – Mobile app testing.* Last checked on 05 12, 2016, from the Testdroid website: http://testdroid.com/

(4) Application Icon (free for non-commercial use). (sd) *Application Icon.* Last checked on 05 09, 2016, from iconarchive:
http://www.iconarchive.com/show/steel-system-icons-by-uriy1966/Documents-icon.html

(5) Start activity from button. (sd) *Start activity from button.* Last checked on 05 09, 2016, from developer.android.com:
http://developer.android.com/training/basics/firstapp/starting-activity.html

(6) Using GestureOverlayView. (sd) *Drawing in Android with GestureOverlayView.* Last checked on 05 09, 2016, from intertech:
http://www.intertech.com/Blog/android-gestureoverlayview-to-capture-a-quick-signature-or-drawing/

(7) Saving BitMap to device. (sd) *Saving BitMap to device.* Last checked on 05 09, 2016, from stackoverflow:
http://stackoverflow.com/questions/17175680/error-when-save-bitmap-to-file-on-sdcard-android

(8) Clear GestureOverlayView. (sd) *Clearing the drawing screen.* Last checked on 05 09, 2016, from stackoverflow:
http://stackoverflow.com/questions/12740733/clear-gesture-from-gestureoverlayview-in-android

(9) Uploading documents. (sd) *Uploading documents.* Last checked on 05 09, 2016, from codingcage:
http://www.codingcage.com/2014/12/file-upload-and-view-with-php-and-mysql.html

(10) Save PDFs. (sd) *Save PDF MySQL.* Last checked on 05 09, 2016, from php-mysql-tutorial:
http://www.php-mysql-tutorial.com/wikis/mysql-tutorials/uploading-files-to-mysql-database.aspx

(11) Android Login. (sd) *Android login with MySQL, PHP.* Last checked on 05 09, 2016, from youtube:
https://www.youtube.com/watch?v=0Fnk9ZgsTVw

(12) Android POST request. (sd) *POST request.* Last checked on 05 09, 2016, from wikihow:
http://www.wikihow.com/Execute-HTTP-POST-Requests-in-Android

(13) HTTPClient. (sd) *HttpClient class.* Last checked on 05 09, 2016, from codeofaninja:
https://www.codeofaninja.com/2013/04/android-http-client.html

(14) Upload signature. (sd) *Upload file to server - Android.* Last checked on 05 09, 2016, from androidexample:
http://androidexample.com/Upload_File_To_Server_-_Android_Example/index.php?view=article_discription&aid=83&aaid=106

(15) ListView Button onClick. (sd) *How to set onClick event for Button in List item of ListView.* Last checked on 05 09, 2016, from stackoverflow:
http://stackoverflow.com/questions/12596199/android-how-to-set-onclick-event-for-button-in-list-item-of-listview

(16) JSON implementation. (sd) *Insert JSON data to server.* Last checked on 05 09, 2016, androidlift:
http://androidlift.info/2015/09/21/android-restful-web-service-with-json-to-insert-data-to-server-mysql-database-using-httpurlconnection-class/

(17) JSON Parsing. (sd) *Perform parsing & display result in a ListView.* Last checked on 05 09, 2016, from youtube:
https://www.youtube.com/watch?v=KSX4zIhiZlM&nohtml5=False