UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

PROYECTO FIN DE CARRERA

Design and creation of a drag-and-drop editor that allows the customization of a sales application in NFC payment terminals

José María Angulo Pinedo

Abril 2014

INGENIERÍA DE TELECOMUNICACIÓN

Design and creation of a drag-and-drop editor that allows the customization of a sales application in NFC payment terminals

Autor: José María Angulo Pinedo
Tutor: Heinz Bircher-Nagy
Ponente: Germán Montoro Manrique

Software Development
Avance Pay AG
Abril de 2014

Resumen

El objetivo de este proyecto es el diseño y creación de un editor drag & drop que permita la personalización de aplicaciones utilizadas en terminales móviles, como TPV (Terminales Punto de Venta). El editor será desarrollado tanto para entornos Android como para clientes web, y tiene como objetivo facilitar la personalización de sus aplicaciones de venta a los pequeños comerciantes que utilicen el sistema de pago de Avance Pay.

En esta memoria primero se ofrece una visión general de los sistemas de pago móviles mediante la tecnología NFC y las ventajas de utilizar un terminal móvil como TPV. Luego se realiza un estudio del funcionamiento de los sistemas drag & drop, así como de los programas WYSIWYG. A su vez se hace una comparación de los distintos editores existentes para Android.

Antes de crear el editor drag & drop se aborda el diseño de la aplicación TPV para entornos Android. Para ello se analizan las características de aplicaciones TPV ya existentes en el mercado, con el propósito de reunir los requerimientos oportunos. Además, se desarrolla un prototipo acorde al diseño propuesto.

Una sección se dedica a la introducción a la programación web. Tanto del lado del cliente como del servidor, presentando algunas de las tecnologías existentes. Estos conocimientos serán necesarios para entender el diseño propuesto para la creación del editor, el cual es a su vez explicado en otra de las secciones, donde se explica el diseño y se detalla el desarrollo del código de los distintos elementos que integran el editor.

Palabras clave

Drag & drop, WYSIWYG, NFC, TPV (Terminal Punto de Venta), pago móvil

Abstract

The main goal of this Project is the design and creation of a drag & drop editor that allows de customization of applications utilized in mobile terminals as POS (Points of Sale). The editor has been developed on Android and web-client environments, and its purpose is to facilitate the customization of Sales Applications to sales men who use the Avance Pay system.

In this report firstly, a general overview of NFC-based Mobile Payments and the advantages of using a mobile terminal as a POS are given. Afterwards, a study about drag & drop editors and WYSIWYG programs functioning is made. Moreover, a comparison about the different existing editors for Android is done.

Before creating the Drag & Drop Editor the design of the POS application is accomplished. For that purpose the POS applications already existing in the market are analyzed in order to gather the appropriate requirements. Furthermore, a functioning prototype consistent with the proposed layout is developed.

One section is dedicated to show the state of the art regarding web programming. Both, client side and server side are explained, presenting some of the existing technologies. This knowledge will be necessary to understand the solutions proposed for the creation of the editor, which is as well explained in detail in other section, where the implementation of the design is explained as well as the code development of the various elements that integrate the drag & drop editor.

Key words

Drag & drop, WYSIWYG, NFC, POS (Point of Sale), mobile payment

Agradecimientos

Muchas gracias a todos los que me apoyaron durante la realización de este Proyecto Fin de Carrera. Por momentos resultó complicado pero vuestros ánimos resultaron clave para la finalización del mismo.

En especial agradezco todo el apoyo que mi familia me dio. Siempre estuvisteis ahí, apoyándome y no dejando que me desesperara. Os quiero.

También quiero darles las gracias a mis compis telecos, muchos de ellos amigos del alma, con los que tan buenos momentos he pasado durante estos años.

Agradecer igualmente a todo el equipo de Avance Pay, con especial mención a Heinz y Peter que hicieron posible este proyecto.

Y como no podía ser de otra manera, gracias a ti María que me escuchaste cuando lo necesitaba, me animaste a mejorar y a no dejarlo nunca. Sin duda fuiste el mejor apoyo que se pueda tener.

Contents

Figures Index

Table index

1. Introduction

1.1. Motivation

NFC payments are already here. Lots are the advantages of having everything integrated in your mobile phone. Therefore, Avance Pay develops corresponding mobile phone based payment terminals. Avance Pay system pretends to be fast, secure, economic, reliable and extensible. It will present many advantages for customers and vendors.

The main motivation of developing a drag-and-drop application is to facilitate to those persons without any programming skill, the customization of their own sales applications. This tool provides the users the possibility of designing and making changes in their own business applications, in a highly intuitive way.

In the same way, the intention is to provide the Avance Pay payment system with a distinguishing element with respect to other payment systems, and for that the idea is to provide the company clients with a tool that allows them to customize their business sales applications.

1.2. Objectives

The main purpose of this thesis is to design and create a drag and drop application for editing sales applications on, mobile phones and web clients. The idea is that vendors, who are using an Avance Pay system, can edit their own sales application: change the interface look, change prices, add new products, add new options, etc.

The system in mind will consist of a drag-and-drop editor for an android environment and other for a web-client environment, which should be designed and created throughout the present thesis.

To develop the application in mind different editors will be analyzed as well as different mobile Points of Sale (mPOS). The idea is to understand how these programs are implemented

1.3. Methodology and working plan

To achieve the goals of this thesis a strict methodology will be followed. First of all modern literature about NFC technology will be reviewed, in order to present a general overview in this work. Besides, articles about Drag and Drop editors and XML code will be also reviewed, among others.

Once all the literature has been reviewed, a general idea of the state of art has been reached and the student has got acquainted with some existing editors, there will be defined the requirements for the Avance Pay whole system as well as a development road map.

After the functional analysis, the intermediate file code (XML) requirements will be defined and the drag-and-drop interfaces will be programmed. Once the base code is ready it will be evaluated. Then, any eventual error will be corrected and any improvement considered will be added.

The work plan will consist of the following steps:

1. Modern literature review about NFC technology and drag and drop editors.
2. Become familiar with the existing editors.
3. Definition of the requirements for the whole system.
4. Define development road map.
5. Implement first release (proof of concept).
6. Evaluate and discuss results.
7. Finalize report.

1.4. Report structure

Chapter 1. Introduction: Introduction of the report, where the motivation and the objectives are exposed.

Chapter 2. Introduction to NFC: The NFC technology is introduced. The modes, features and the standard are explained, as well as an introduction to Mobile Payments, focusing on NFC payments in the last section of the chapter.

Chapter 3. Introduction to drag & drop editors and WYSIWYG: What these terms mean is exposed and some android editors reviewed and compared.

Chapter 4. Web programming: Web programming is introduced. The client and the server sides are explained. The main web standards are also analyzed.

Chapter 5. POS software design: First of all, the features and GUI elements of the different mPOS applications are analyzed and compared. Then, the design of an own POS application is introduced and finally the development of a prototype application is explained.

Chapter 6. Design and development of the Drag and Drop Editor: Firstly, the requirements of the Drag and Drop Editor are exposed as well as the proposed solutions. Then the design of the editor is showcased and the development of the code explained.

Chapter 7. Conclusions and future work: To finalize the conclusions of the present thesis and the future work are written.

2. Introduction to NFC

2.1. Near Field Communication origins

NFC technology is based on RFID (**R**adio **F**requency **Id**entification). RFID technology origins are not clear, but Harry Stockman's work published in 1948, *Communication by Means of Reflected Power*, is commonly accepted as the origin of RFID. In this work, the point-to-point communication method in which the device with the information is a passive element which does not generate any energy, but reflects and modulate the field generated by the initiator to transmit its information is defined [1]. This technology is still commonly used in electronic identification cards.

Philips and Sony jointly developed a standard for two-way contactless communication, NFC (**N**ear **F**ield **C**ommunication). The ECMA (**E**uropean **C**omputer **M**anufacturers **A**ssociation) standardized the *RFID Mobile* or NFC in 2002, within the ECMA-340 standard and was approved by the ISO in 2003, within the ISO/IEC 18092 standard.

This International Standard defines communication modes for **N**ear **F**ield **C**ommunication **I**nterface and **P**rotocol (NFCIP-1) using inductive coupled devices operating at the center frequency of 13,56 MHz for interconnection of computer peripherals. It also defines both the Active and the Passive communication modes of Near Field Communication Interface and Protocol (NFCIP-1) to realize a communication network using Near Field Communication devices for networked products and also for consumer equipment. This International Standard specifies, in particular, modulation schemes, coding, transfer speeds, and frame format of the RF interface, as well as initialization schemes and conditions required for data collision control during initialization. Furthermore, this International Standard defines a transport protocol including protocol activation and data exchange methods [2][3].

However, the most important milestone for the technology was in the 2004 when Nokia, Philips and Sony created the NFC Forum. The Near Field Communication Forum was formed to advance the use of Near Field Communication technology by developing specifications, ensuring interoperability among devices and services, and educating the market about NFC technology [4]. Besides, the NFC Forum promotes the N Mark as an emblem of the technology, pretending to be a universal symbol for NFC, so that consumers can easily identify NFC-enabled products as well as the locations where NFC services are available.

Fig. 1 NFC Forum N Mark

2.2. NFC modes

There are many ways to use the NFC technology. These ways of using it depend on the mode being used. There are three main NFC modes, defined by the NFC forum:

- **Peer to peer:** This mode is defined for device to device link-level communication.
- **Reader and writer:** This mode allows applications for the transmission of NFC Forum-defined messages.
- **Card emulation:** This mode allows the NFC-handset behave as a standard Smartcard.

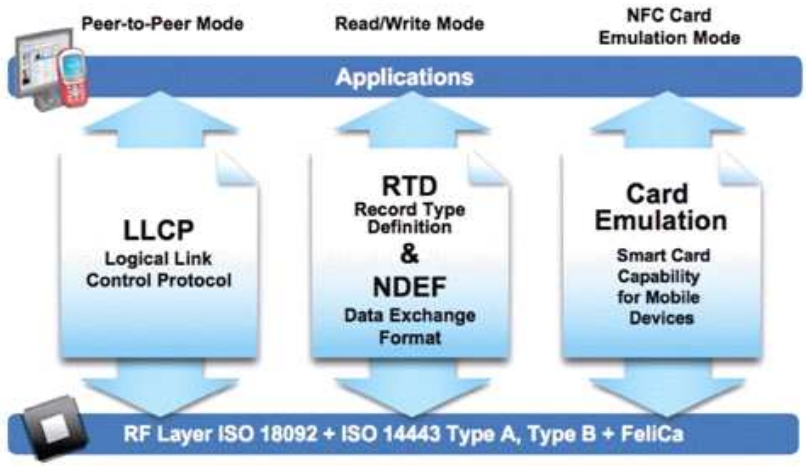

Fig. 2 NFC Communication Modes (Source: NFC Forum)

2.3. NFC features

Near Field Communication main features are the following [5]:

Fig. 3 Mobile-tag interaction

Short-range wireless communication: NFC works using magnetic induction: a reader emits a small electric current, which creates a magnetic field that in turn bridges the physical space between the devices. That field is received by a similar coil in the client device, where it is turned back into electrical impulses to communicate data such as identification number, status information, or any other information. So-called 'passive' NFC tags use the energy from the reader to encode their response, while 'active' or 'peer-to-peer' tags have their own power source and respond to the reader using their own electromagnetic fields.

The NFC electromagnetic field range is very short, being in practice of 4 centimeters or less, what means that the devices must be close enough to each other in order to establish the communication.

Short data transferences: The NFC communication is not oriented to accomplish massive data transactions. Furthermore the possible bit rates are in the order of hundreds of Kbit per second, what allows only the transference of small data quantities.

Tags typically store between 96 and 512 bytes of data and transfer data using at speeds of 106Kb/s, 212Kb/s, 424Kb/s or 848Kb/s – enough to move small pieces of information virtually instantaneously, as is essential in high-volume transport applications.

Operations in the ISM frequency: Like RFID, NFC works in the 13.56MHz radiofrequency spectrum, using less than 15mA of power to communicate data over distances that are usually far less than 20cm.

NFC works in the ISM (**I**ndustrial, **S**cientific and **M**edical) frequency band, what means that it is an unregulated band and has not licensing costs associated. Radio communication services operating within these bands must accept harmful interference, which may be caused by these applications [6].

2.4. NFC standard

The **N**ear **F**ield **C**ommunication **I**nterface and **P**rotocol (NFCIP) defines the physical and the transport layer of the technology. It is divided into two parts: NFCIP-1, which defines the NFC communication modes on the RF layer and other technical features, and NFCIP-2, which specifies the communication mode selection mechanism with the purpose of not disturbing any ongoing communication at $13{,}56$ MHz, for devices implementing ECMA-340 and the reader functionality for integrated circuit cards compliant to ISO/IEC 14443 or ISO/IEC 15693 [2][7].

The NFCIP-1 standard defines the two working roles that NFC devices can assume in a communication. These roles are both necessaries in a communication and are the following:

- **Initiator:** Device that initiates the communication.
- **Target:** Device that responds to the initiator.

Being the transmissions based in the question-answer principle, a device can only respond to another if the other one has initiated the communication with it. NFCIP-1 devices shall detect external RF fields at 13,56MHz with a value higher than a specific threshold while performing external RF field detection. The threshold value is $H_{threshold} = 0{,}1875\ A/m$.

This standard also defines two working modes:

- **Active Mode:** An Initiator and a Target shall alternately generate a RF field.
- **Passive Mode:** An Initiator shall produce a RF field to energize the target.

An active device acting as the initiator and a passive one as the target is one of the most utilized configurations. In this scenario, the active element generates a radiofrequency field which supplies electromagnetic energy to the passive device allowing it to send its information modulating the signal with a certain codification.

With the NFCIP-1 specifications, the peer-to-peer communication concept between two terminals is introduced. In this case, an active device A acts as initiator of the communications, sending a message to a device B. While it is waiting the answer, the device A deactivates its radiofrequency field and the device B activates it to send out the answer. Thus, during the communication the devices alternate the activation of their fields in order to send and receive the data.

In this last case is important not to confuse modes and roles. Both devices are actives, although its RF field is deactivated. Besides, although B is transmitting data, the initiator of the communication will continue being the device A until the communication is finished.

2.5. Mobile payments

A mobile payment can be defined as a transfer of funds in return for a good or service, where the mobile phone is involved in both the initiation and confirmation of the payment. The location of the payer and supporting infrastructure is not important: he may or may not be "mobile" or "on the move" or at a Point of Sale (PoS); the payment may be processed by credit cards or by a prepaid wallet.

2.5.1. Mobile payment categories

Payments can be done between consumers (P2P or C2C) or between consumers and companies (C2B). In addition, payments can either be executed in proximity, for example at the counter in a shop, or remotely, for example paying online via a mobile phone [8].

In the following figure, the different mobile payment categories are shown.

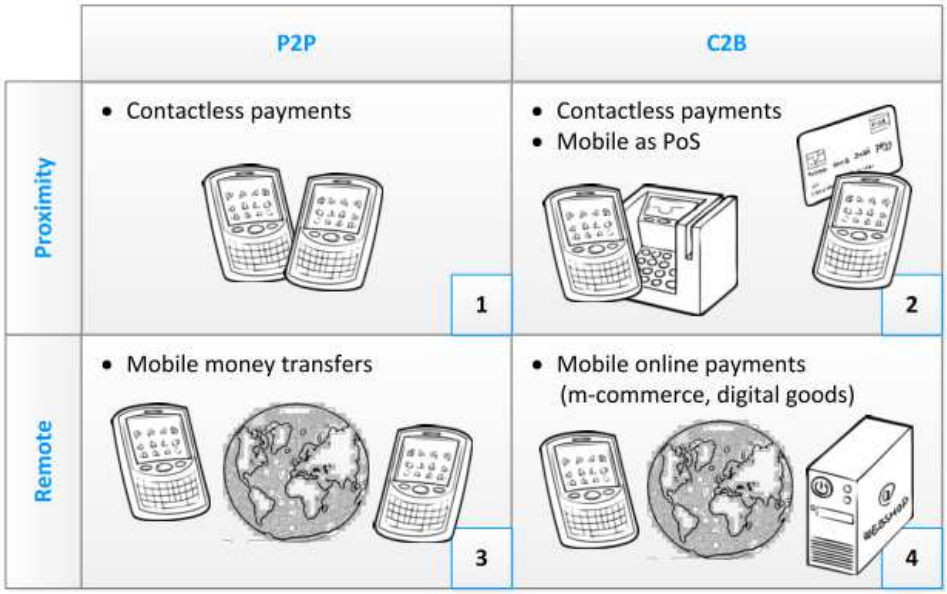

Fig. 4 Mobile payment categories. Source: [8]

Proximity payments

1. **Contactless payments**

Contactless payments are payments done in proximity without making contact. Some examples are paying at PoS by holding your mobile phone in proximity, transfer money to your friend by moving the phones towards each other or by paying your metro ride by holding your phone at the reader. As the examples show, contactless payments can be done both between consumers (P2P) and between consumers and merchants (C2B).

Contactless technology can be divided into two categories:

- Vicinity: this technology offers a maximum read distance of 1 to 1.5 meters.

- Proximity: this technology has a much smaller read distance, usually about 7.5 centimeters in most instances.

There are multiple ways to perform contactless communication. Nowadays NFC gets most of the attention in the field of contactless payments, and in general in the field of mobile payments. Still there are many other, similar methods to perform contactless payments; they can be done via the mobile internet, QR codes, Bluetooth or radio frequency waves of the mobile speakers.

QR Codes: For example, PayPal is using it for allowing payments. The Payment Code feature allow PayPal users to pay for things in physical retail stores by scanning a QR code generated in the app, or by using a one-time four digit code in stores that use PIN codes but don't have a scanner [9].

Bluetooth: PayPal also embraced Bluetooth Low Energy (BLE) technology for payments after writing off Near Field Communications (NFC) [10].

This payment system has been called Beacon, a payment service that employs the BLE wireless communications technology so that customers in a store can pay without reaching for their smartphone.

For the technology to work, retailers must plug in a USB dongle into a compatible point-of-sales system. When customers walk into the store with a smartphone and the PayPal app, they will be prompted to check in for hands-free payments. The PayPal app does not need to be open and users don't need to have a phone signal or GPS turned on.

Customers can elect inside the app to have their device always check into their favorite stores so they don't have to think about payment the next time they arrive. No information is exchanged between the user and the store if a customer ignores the prompt and does not check in.

Ultra sonic: Naratte, Inc. operates as a technology development company. It offers Zoosh, a software that enables users to securely exchange data between devices using the speakers and microphones [11].

2. Mobile devices used as POS

A recent phenomenon is the use of mobile devices as point of sale (PoS) to accept, typically, card payments. With the help of an extra device and an application for the hardware a mobile device can be used to accept card payments. The external card readers typically support payments between consumers and small enterprises. They are specifically targeted for enterprises not large enough for traditional PoS devices, thereby providing access to those otherwise excluded due to high sum investments [12][13][14].

The next images show different systems that use mobile devices as POS to accept credit and debit card payments.

Fig. 5 Square, Inc. payment hardware and software.

Fig. 7 Square, Inc. signature process.

Fig. 6 iZettle payment hardware and software.

Fig. 8 iZettle payment using a Tablet as POS.

These Mobile Points of Sale (mPOS) accept credit and debit cards, but do not accept contactless payments. Avance Pay's technology, on the other hand, offers the possibility of turning NFC-enabled smartphones into EMV compliant, contactless payment terminals with low investment costs.

The Avance Pay Mobile Payment Terminal is designed to accept all forms of contactless payments such as contactless credit and debit cards, and also NFC-based wallet applications. Therefore no additional readers are required; the merchant's NFC-enabled smartphone or tablet serves as an all-in-one solution.

Due to full end-to-end protection, the Mobile Payment Terminal delivers highly safeguarded transactions in both online and offline modes. Payments can be accepted securely, even when the Mobile Payment Terminal is running offline, without server connectivity. This allows for very flexible usage in different areas of application, where other devices would fail.

Remote payments

3. Mobile money transfers

A mobile money transfer is a transfer of funds from one consumer to another over long distance. There are two contexts in which mobile money transfers are executed.

- Payments between consumers within the same country: In developing countries this could be in between people in a major city and their relatives on the country side. However, also in a country such as the US there is a large market for P2P mobile money transfers, supported for instance by the PayPal services.

- Consumers sending money overseas: This area is dominated by remittances. A remittance is a transfer of funds from a foreign worker to his home country. This typically constitutes migrant workers from Africa or Asia Pacific that work in Europe and North America, but also happen within the same regions. Among the largest recipient countries are India, China, Mexico and the Philippines.

4. Mobile online payments

Mobile online payments are payments via the mobile browser or via an app on the mobile phone. There can be distinguished between two use cases of mobile online payments that both are executed in the B2C- environment: m-commerce and digital goods.

- M-commerce: online shopping for goods or services on the mobile phone. Online business models are incorporated into mobile devices in order to maximize revenue opportunities. Mobile devices are thereby becoming an ecosystem for m-commerce, allowing developers to build their own m-commerce applications on the mobile device.

- Digital goods: purchase of digital goods on the mobile phone through mobile platforms. Often ignored in the field of mobile payments, buying apps, games and music is the fastest growing area in mobile payments. Depending on the platform, mobile payments are typically done by either having stored card payment details linked to the user account on the mobile phone, or prepaid cards credited to the user account.

2.5.2. Key stakeholders

2.5.2.1. Providers of mobile payment services

Mobile network operators (MNO's)

For mobile network operators mobile payments are an attractive proposition for achieving a return on the investments made in infrastructure over the last two decades through reduction of churn, extra payment related revenues and through associated increases in air time and data use. For mobile network operators mobile payments also hold the possibilities of allowing for diversification into other areas of the consumer's needs and lifestyle.

Financial institutions

For financial institutions mobile payments are first and foremost a defensive play. From a retail banking point of view, financial institutions are primarily focused on protecting the current account and surrounding loan products. Retail payments including mobile payments are more often than not a loss leader for these more profitable products. From a wholesale banking point of view financial institutions have already been disintermediated to some degree from their wholesale customers by third parties in the area of online payments. Financial institutions are keen to avoid the further worsening of this situation through third party mobile payments. Mobile payments also hold the allure for financial institutions of assisting in the ongoing battle to reduce the use of cash and its associated costs. Furthermore in developing geographies mobile payments offer financial institutions the opportunity to cost-effectively capture and service unbanked and underbanked communities.

Handset manufacturers (OEMs)

Handset manufacturers (or **O**riginal **E**quipment **M**anufacturers, shortly OEMs) produce the mobile devices and thereby determine their capabilities and usability. The success of the use of the mobile device for payments has the potential for resulting in a substantial increase in both sales to new customers but also for the renewal of existing devices in the market to ones that are payment capable.

Technology providers

As with any technology led development, mobile payments hold the most promise for technology vendors and systems integrators: chip manufacturers create the smart card chip on which the mobile payment application or secure element can reside; the secure element issuer personalizes the chip with the secure element; the service provider offers specific services for end users such as authentication; while the Trusted Services Manager (TSM) enable the service provider to use the secure element.

All these organizations are positioning themselves to provide the infrastructure and messaging for mobile payments and in the process offering to act as a trusted intermediary between the banks and the mobile network operators.

It should be noted that in the market today both financial institutions and mobile network operators play the roles of Secure Element issuers, service providers and trusted services manager.

2.5.2.2. Demand side

Merchants

For merchants, Point of Sale mobile payments could provide faster throughput at the checkout and the ability to send real time marketing messages to the consumer. However, faster throughput could also be achieved through contactless cards and it is yet unclear whether consumers would actually want or appreciate real time marketing messages from the merchant on their phone. However un-manned or remote Point of Sale locations could benefit from mobile payment by allowing a reduction in servicing costs. Remote mobile payments provide another channel for merchants and as such are an attractive proposition if the use of the channel can gain wide scale adoption at lower costs than existing channels.

Consumers

From the perspective of the end consumer, the mobile phone has achieved 'permanent share of pocket', i.e. next to the wallet and keys it is the object that is most likely to be constantly with the consumer. Furthermore, consumers are increasingly more comfortable with the mobile phone fulfilling more than one function, with mobile devices slowly morphing into multi-media and multi-application devices. However, does this mean that end consumers are ready to abandon the wallet and rely primarily on the phone, which is more a lifestyle or leisure tool, for the important task of handling their payments?

2.6. NFC Payments

NFC Payments are defined as payments carried out leveraging the NFC technology, where there is an NFC-enabled device acting as the reader/cashier and a Contactless Smart Card or an NFC-enabled mobile device on Card Emulation Mode as the payer.

These payments can be carried out by a POS reader and a Contactless Smart Card, by a POS reader and an NFC-enabled mobile device leveraging the NFC Card Emulation Mode or between two NFC-enabled mobile devices, acting one of them as the cashier and the other as the payer.

The market developments have been quite uneven throughout the world. Some countries are much more advanced in terms of technology deployed and business cases implementation.

In Japan, South Korea, and other Asian countries, several successful mobile payment solutions have already been launched (e.g., Mobile Suica, Edy, Moneta, Octopus). Mobile phones are used for making purchases at convenient stores, transit fares, and many other goods. Governments and influential mobile network operators (MNOs) pushed to enhance

the development of mobile payment services, which could partially explain the greater success in Asia.

In Europe and North America, the development of mobile payments has not been as successful, with the exception of several countries including Austria, Spain, Croatia, and the Scandinavian countries

One major difference of the mobile payment services initiated in Asia, Europe, and the U.S. markets is the technology deployed. In fact, Japan and South Korea telcos distribute mobile phones ready for RFID technology (Radio Frequency Identification). This could be partially explained by the ubiquity of contactless cards (IC cards) for payment transactions.

In Europe and the U.S., mobile payment systems are still mostly based on SMS (Short Message Service), USSD (Unstructured Supplementary Service Data), WAP (Wireless Application Protocol), or IVR (Interactive Voice Response). This was done in order to facilitate the uptake of mobile payments by using the existing technologies installed in the current customer base. As these technologies were not making mobile payments very convenient and easy to use, companies are now testing new schemes based on NFC.

To a certain extent, NFC is the fusion of a contactless smartcard (RFID) and a mobile phone. Mobile phone can therefore be used like a contactless card. NFC has a shorter range than other wireless technologies embedded in a phone (e.g., Bluetooth, Wi-Fi). Another great feature of NFC is that mobile phones are capable to act as RFID tags or readers. This ability creates many opportunities for innovative services. There are several recurring examples used to illustrate the new possibilities offered by NFC. One good example is the smart poster. An RFID tag is embedded in a poster. By waving the mobile phone close to the poster, the user gets more information about the poster. Possible applications are ticketing or couponing. This could be seen as a new way to sell concert tickets through mobile phones.

The expected success of NFC is not only limited to the mobile payments capability. Many see that this technology will enable many innovative mobile services. Various applications in different industries (e.g., retail, logistics, transportation) could be developed to take advantage of the interaction between RFID tags and mobile phones. Access control scheme based on NFC also seems to be quite popular. This means that mobile phones could reinforce their position as a multi-function device. NFC also aims at facilitating communication between various devices (e.g., business card exchange, driver configuration), which could greatly contribute to the democratization of mobile computing.

2.6.1. Contactless Smart Cards

A contactless smart card is any pocket-sized card with embedded integrated circuits that can process and store data, and communicate with a terminal via radio waves. Contactless smart cards can be used for identification, authentication, and data storage. They also provide a means of effecting business transactions in a flexible, secure, standard way with minimal human intervention. A smart card has a microprocessor or memory chip embedded in it and that coupled with a reader it has the processing power to serve many different applications.

The following figure shows a separate contactless smart card with reader. In NFC card emulation mode, mobile phone will work just like this card with the additional benefits that we don't need to carry extra cards [15].

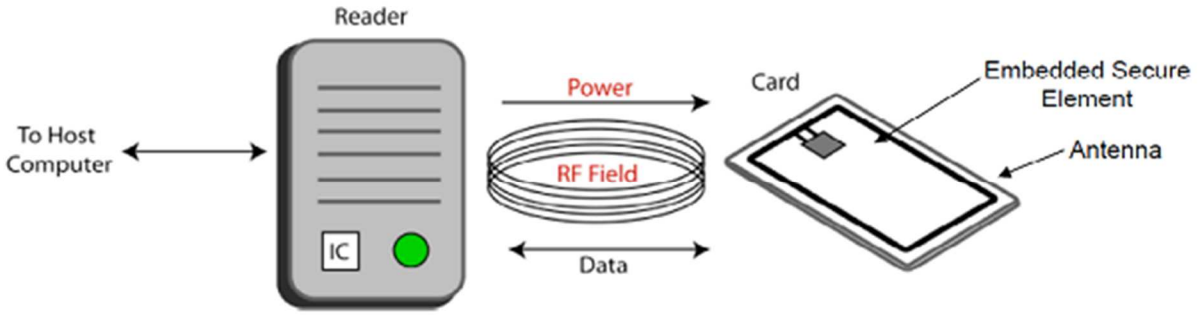

Fig. 9 Contactless smartcard operation scheme

2.6.2. Secure Element

In general, MNO (**M**obile **N**etwork **O**perator) issues SIM (**S**ubscriber **I**dentification **M**odule) cards to mobile users where personal identity and applications (such as contact application) are stored. Multiple contactless application can be stored in this card (updated from conventional SIM) for each use cases. This card works as secured element that stores and execute the contactless applications.

UICC (**U**niversal **I**ntegrated **C**ircuit **C**ard) cards are the evolution of SIM cards. Like SIMs, they go in our phone, have an application that stores our contacts, let our network identify who we are, and work with any network in the world. However UICC cards can run more than one application.

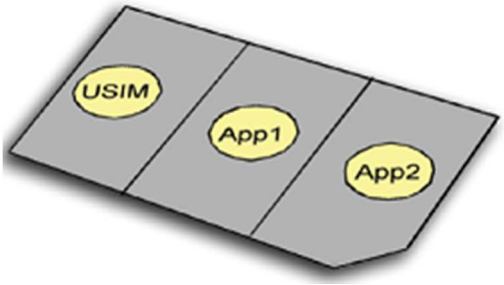

Fig. 10 UICC memory slots

The following diagram shows an NFC echo system diagram, showing those involved in NFC payments:

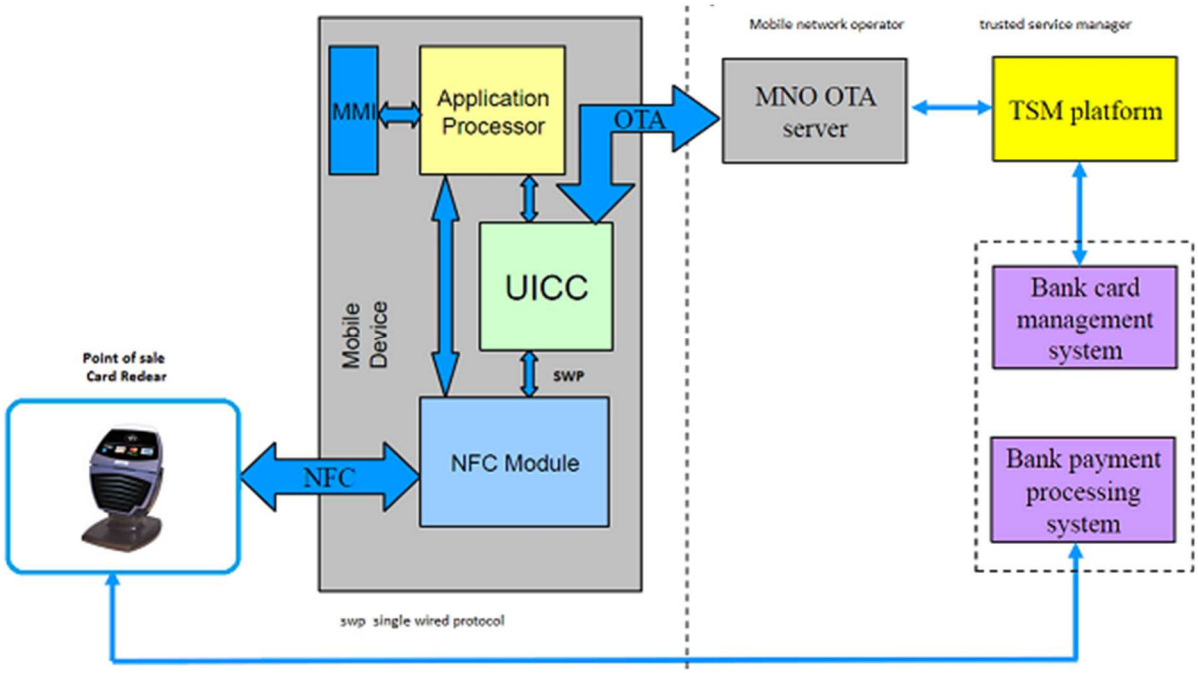

Fig. 11 End-to-end NFC Payment System scheme

More than one credit or debit card can be stored in the same UICC, increasing the convenience of PBM (**P**ay **B**uy **M**obile) to the consumer. In use, data transferred by NFC from the handset (actually from the UICC) to the reader is communicated to financial organizations using the same secure process as used for conventional credit or debit card transactions. One of the hottest topic with regard to NFC are SWP (**S**ingle **W**ire **P**rotocol) enabled UICCs. Whereas several handset manufacturers such as Nokia can equip its phones (some series 40, and some coming Symbian devices) those are capable to communicate with embedded secure element. The SWP connects the UICC to the NFC Modem through a single wire and thus adds contactless functionality to the UICC (SIM).

Security is a key consideration with NFC. Retail and transit payments with a mobile phone require wireless carriers, retailers, transport providers and banks to all work together. All of the transaction and payment card accounts information need to be kept secure and apart. For this reason, NFC requires the use of a "secure element." The UICC is the preferred technology for the secure element that stores subscriber details – such as credit card account numbers, transit accounts, and mobile phone details – and keeps these details separate and secure. To enable proximity payments, secure element capable of authenticating itself to a bank, and resistant to physical or logical attack. Contactless frontend interface standardized by ETSI to connect a UICC to a contactless frontend to pass data from UICC to card reader via NFC where the payment is processed by other echo system members. Data is communicated between UICC and NFC module with SWP which was developed to support the use of contactless payment applications running on UICCs – it enables the payment application on the UICC to communicate with contactless readers via the NFC interface of the handset. The standard is approved by ETSI November 2007.

Let’s take an example of micro payment application “NFC Wallet” consists of several parts. The first one - and the only one that is actually recognized by the user – is the applications running on the device. This part provides the user with a GUI (graphical user interface) and allows him/her to recharge the wallet over the air and view the amount of money available on the handset. To recharge the wallet, the application is able to establish a

GPRS connection to a transaction server located at the mobile network operator's supported bank for example. The transaction server accepts the incoming connection from the NFC handset. The authentication of the device is done thru a solution of the mobile network operator. The server checks the balance of the account of the customer and then sends the money (encrypted) back to the handset. There the money is directly stored in the secure element.

Another example could be using credit card application (see the previous figure). User select the application in device (may needs to give password), he/she touches the NFC reader (at point of sales), the reader gets the user's credit card details from the UICC via NFC and pass it to process with conventional ways and in return user get an electronic receipt that can be stored in device.

3. Introduction to drag & drop and WYSIWYG editors

3.1. Drag & drop editors

In computer graphical user interfaces, drag and drop is a pointing device gesture in which the user selects a virtual object by "grabbing" it and dragging it to a different location or onto another virtual object. In general, it can be used to invoke many kinds of actions, or create various types of associations between two abstract objects.

3.2. WYSIWYG programs

What-You-See-Is-What-You-Get (WYSIWYG) means that all elements of the page will be displayed on the exact same position (fixed layout) as in the designer; unlike fluid (dynamic) layouts (generated by traditional HTML editors) where the position of objects depends on the position and size of the objects surrounding it.

3.3. Android existing editors

There are a few frameworks existing to create Android applications, but not that many to edit the layouts which define the GUI of the application. Hereafter, some mobile development frameworks will be analyzed in order to present some features that might be interesting for Avance Pay's editor. DroidDraw is the only tool presented here that is simply intended to edit the android layouts.

3.3.1. App Inventor

App Inventor for Android is an application originally provided by Google, and now maintained by the Massachusetts Institute of Technology (MIT) [16].

Basically, App Inventor lets you develop applications for Android phones using a web browser and either a connected phone or emulator. The App Inventor servers store your work and help you keep track of your projects.

Once you are logged into your account you have access to the edition area. Here, a WYSIWYG editor is available to create the GUI and a blocks system to define the behavior of the application.

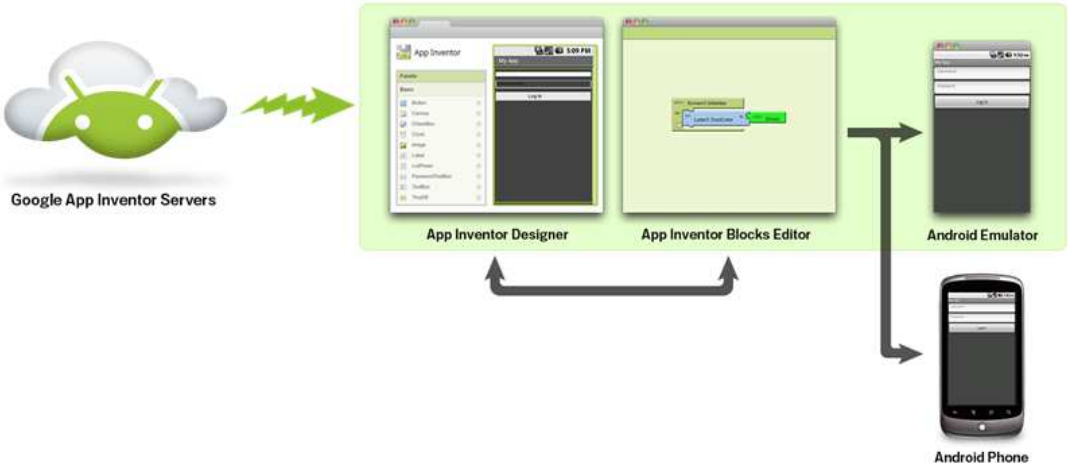

Fig. 12 App Inventor scheme

The GUI editor is accessed directly through the web browser, while the block editor is accessed via JavaWS (**Java W**eb **S**tart). As shown in the picture bellow, in the top right of App Inventor's edition web site there are two buttons; one for packaging the project into an .apk file and another for opening the Block Editor.

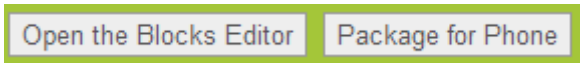

Fig. 13 Open Block and Package for Phone buttons

When the user clicks on the "Open the Blocks Editor" button a JNLP file is downloaded. This is a file created in the **J**ava **N**etwork **L**aunching **P**rotocol (JNLP) format; used for launching and managing Java programs over a network or on the Web; it can be double-clicked to run the program if the Java Runtime Environment (JRE) is installed.

JNLP files contain information such as the remote address for downloading the Java program (.JAR file) and the initial class to run. They are saved in an XML format and can be viewed or edited with a text editor.

However, in order to execute the blocks program App Inventor's Setup software package must be installed. This is really simple; the user only has to download the installer from App Inventor's web page.

As mentioned before, App Inventor consists of two main blocks: the Designer and the Blocks Editor.

Designer

On the left side of the editor, there is a palette where the user can pick the elements he wants to include in his layout. By dragging and dropping them into the area that represents the phone screen, the user is able to add all the elements needed to each of the screens he wants to create.

Create a new screen is as easy as clicking a button. This button is just above the viewer. Together to this button you can find buttons to save at any point your work, create a check point and add or remove a screen.

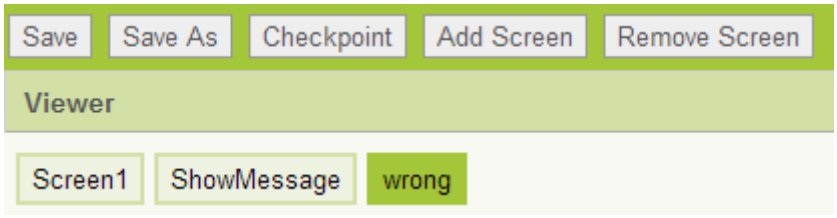

Fig. 14 App Inventor's buttons

On the right side of the designer there is a list of all the components used in your layout, named Components. Here you can find, as well, a Properties column, just to the right of the Components column. This properties column allows you to modify the properties of each one of the components included in the layout.

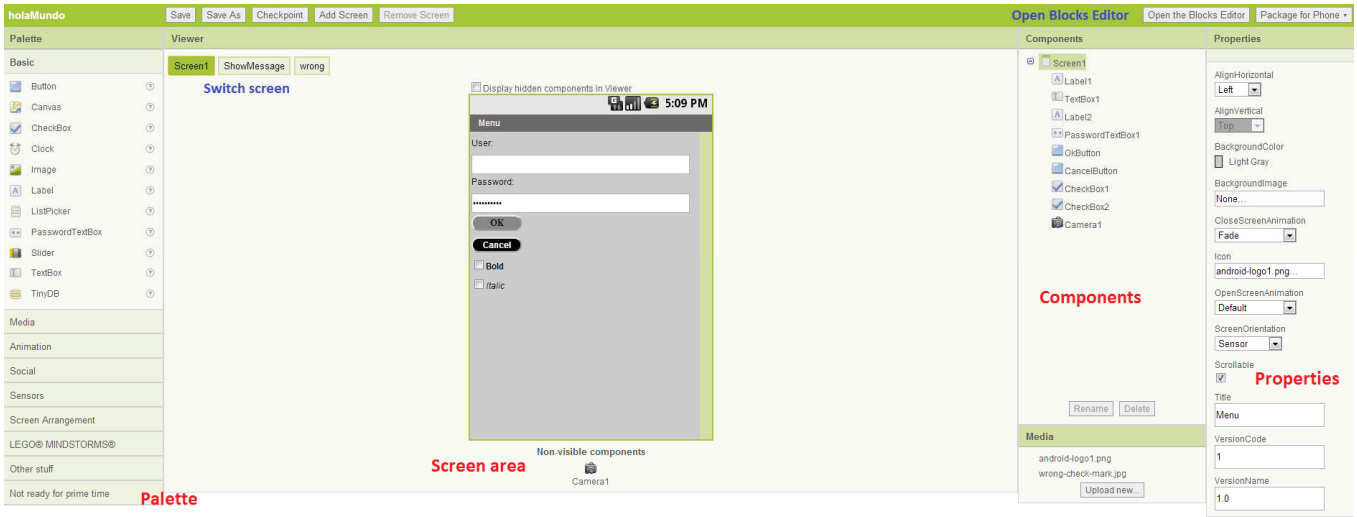

Fig. 15 App Inventor's Designer

Blocks Editor

The Blocks Editor allows the user to modify the behavior of the application in a high intuitive way. It consists of forming logical blocks to define the behavior of the different components which are part of each screen. In order to help the user to perform this task the program offers some feedback, by allowing the users to see their progress either in an Android-based mobile device or using an android emulator.

The mobile device can be connected in two different ways: connecting the device through the USB cable to the computer or via Wi-Fi. With both options the user can check any change he performs within the designer. The only action that is not supported is the change to another screen since the user must open the Block Editor for each screen independently.

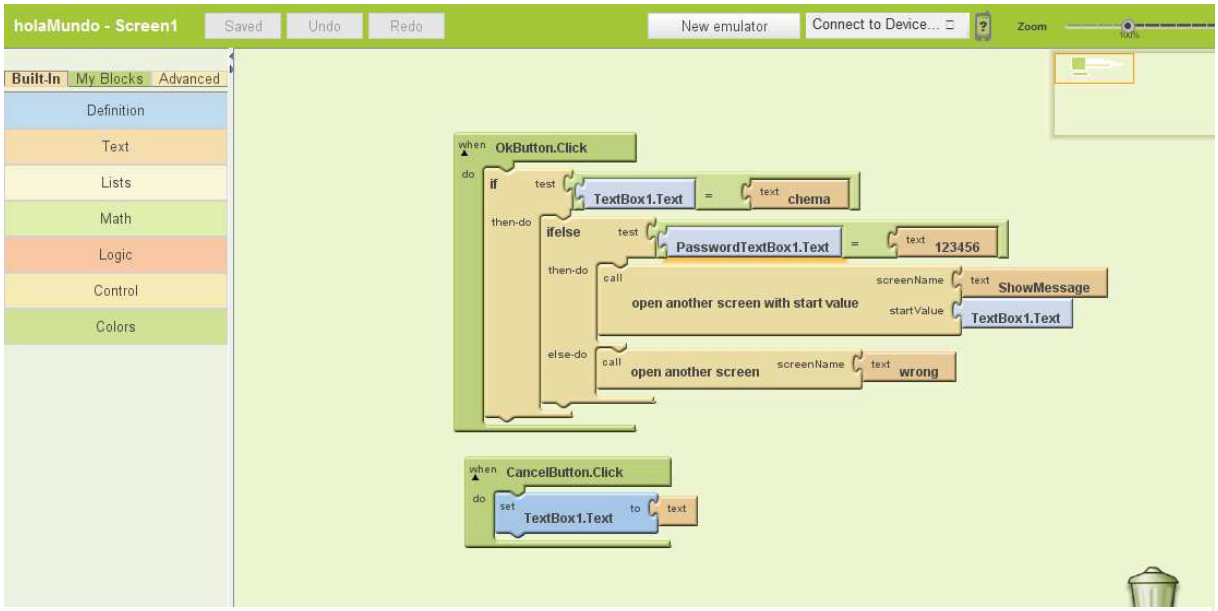

Fig. 16 App Inventor's Blocks Editor

On the left side of the editor window there is a menu where the user can pick the most suitable block to perform the desired task. This menu contains three different tabs:

- Built-in: Range of generic elements such as blocks to interact with texts, colors, lists, logic, controls, etc.
- My Blocks: In this tab are contained the blocks to interact with the components added in the GUI editor, such as buttons, TextBoxes, Checkboxes, etc.
- Advanced: Blocks to add generic functionality for the elements added.

Pros:

- The design is stored in the server, so the users can easily log into their accounts from a web browser to edit their app.

- Use of an android emulator to check the results or by connecting an android-based mobile device via a USB cable or via Wi-Fi.

- The use of the blocks to define the behavior of the app is intuitive and user-friendly.

- Large range of components available to add to the layout.

Cons:

- Only linear layout allowed in the Designer. This restricts the possibilities of the user to adapt the layout to the idea he may have in mind.

3.3.2. DroidDraw

DroidDraw is an open-source WYSIWYG program that allows the creation of Android XML layouts by dragging and dropping different android widgets.

Is it accessible via browser, accessing their web page which contains a java applet, or locally, if you download the java executable from their website.

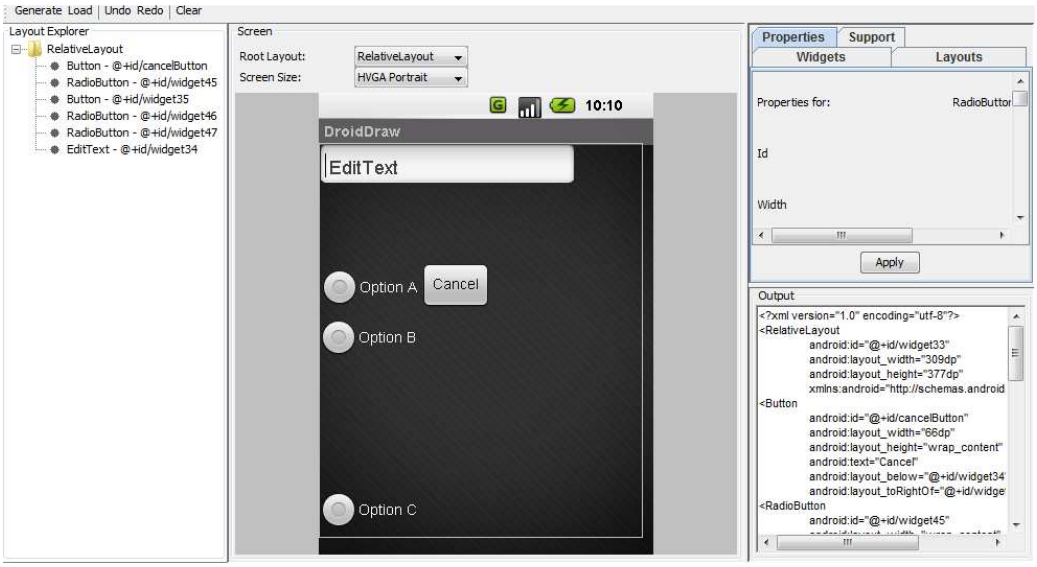

Fig. 17 DroidDaw application

The designer is really simple. It contains four frames. On the left side there is the Layout Explorer, where the user can see all the components added to the layout.

The central frame contains the Screen representation. Here the components can be dragged and dropped from the widgets list in order to edit the screen as desired. This is a WYSIWYG editor that allows the user to select the kind of layout he wants to use. The possible layouts are the following ones: Linear Layout, Relative Layout, Absolute Layout, Table Layout, Scroll View and Tab Host.

At the bottom right side is the output frame. Here is where the user can see the code generated based on the layout. The code generated is android-xml, so this tool is thought for programmers who want to avoid the programming of the layout. After editing it visually by means of DroidDraw they can pick the generated code and place it in the layout folder of their android project.

Just above the output frame is the fourth frame. This frame contains four tabs: Properties, Widgets, Layouts and Support. Within Widgets the user can find a range of android components to place in the layout. Layouts contain a sort of layout to place within the screen layout so that the user can create small areas

Pros:

- Generates the code in an output window, so it could also be edited manually.

- The layout type can be modified and it is even possible to create sub-areas within the root layout.

- It is possible to load an existing layout to edit it.

Cons:

- There are not a wide variety of widgets available.

- The program not always responses as it should do.

3.3.3. PhoneGap

PhoneGap is a web-based mobile development framework, based on the open-source Cordova project. PhoneGap allows you to use standard web technologies such as HTML5, CSS3, and JavaScript for cross-platform development, avoiding each mobile platform's native development language. Applications execute within wrappers targeted to each platform, and rely on standards-compliant API bindings to access each device's sensors, data, and network status.

The application is rendered using platform's web browser engine, not individual native UI objects. This entails that the look-and-feel is not as good as in a native application.

Furthermore, PhoneGap does not offer any drag-and-drop GUI to create the applications, but you can instead use any Web Editor to create your HTML/CSS code. Some of this Web Editors are for example DreamWeaver, Kompozer or CKEditor. The user may utilize one of these editors to create the different screens of their mobile app and then wrap all these files with PhoneGap in order to obtain the packages for each platform they want.

Although it may be a good approach in other cases, this is not what

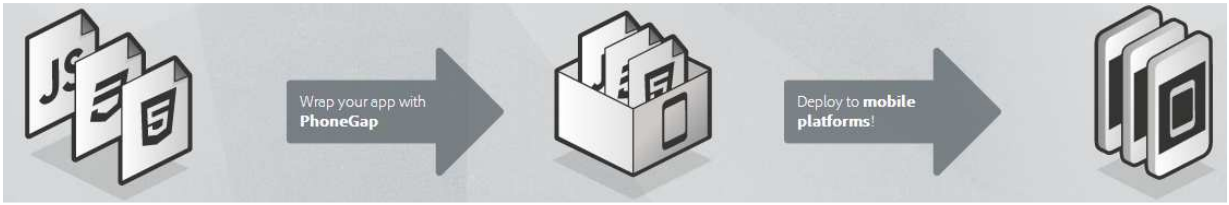

Fig. 18 PhoneGap's process

Pros:

- The possibility to deploy the code to many mobile platforms.
- Open source.

Cons:

- Applications are rendered using platform's web browser engine, not individual native UI objects. Consequently, the look-and-feel is not as responsive as in a native application.

3.3.4. AppyPie

Cloud based DIY Mobile App Builder that allows users with no programming skills, to create an app for Windows 8 Phone, Android and iPhone applications.

With AppyPie's App Maker, there is no need to install or download anything, you can just drag & drop app pages to create your own mobile app online. Once the App is build, you will receive an HTML5 based hybrid app that works with Android, iPhone, iPad, Windows Phone and Blackberry.

The fancy thing about AppyPie is that it is thought for people without any programming skill. Therefore, they have created a very simple designer that creates the applications based on predefined layouts for different topics.

Fig. 19 AppyPie's App Creator - Select App Type

First of all, once you entered the editor you have to choose the topic of your application.

Once you have chosen a topic you are requested to choose a name for the app, a logo, the background picture and the layout type.

Then you have the possibility to add new functionality or remove it, as well as edit some fields.

The possibilities here are not really wide, but the interesting thing about this mechanism is that you can offer the user a guided way to edit his apps, offering pre-programmed features. The user only has to decide whether he is interested in adding one feature or another, picking up those he may want to include.

Pros:
-Different edit options for different type of apps.
-Possibility to upload a logo and change the name of the app.

Cons:
-The possibilities of this editor are few. The user options are limited, but this may also be a good aspect if the users do not have any programming skill.

3.3.5. Comparison table

The following table presents a summarized comparison among the previously described editors:

	App Inventor	DroidDraw	PhoneGap	AppyPie
Mechanism	Graphical editor for the GUI of the app and a Block Editor for the behavior.	WYSIWYG editor for the GUI part of the app. This editor is only intended for this purpose.	Creates cross-platform apps from HTML, CSS and JavaScript files. The HTML/CSS files can be edited with a graphical Web Editor.	Guided app editing. The user chooses the name of the app, the topic, the logo and some features to include within the app.
Features	Offers the possibility to verify the modifications the user is carrying out by connecting an android device or using an android simulator.	• Output frame to show the generated code for further modifications. • Import a pre-designed layout to modify it.	The packaging of the apps can be performed from PhoneGap's cloud system. The user only has to upload a zip folder containing all the files.	User-friendly. Extremely intuitive.
Platforms supported	Android	Android	Android, iOS, Microsoft Mobile, BlackBerry, webOS, Symbian, Bada	Android, iOS
Licensing	Open Source	Open Source	Open Source	AppyPie account
Pros	• Designs saved in App Inventor servers. • The user can preview the result of their work. • Large range of components available to add to the layout.	• Generates the code in an output window • Is possible to create sub-layout areas within the root layout.	• Multi-platform	• Simple
Cons	• Only linear layout allowed in the Designer.	• It does not work properly every time.	• Rendering with web browser engine. The look-and-feel is not as good as	• Few editing options.

			in a native application.	
Server-based	YES	NO	NO	YES

Table 1 Android Editors comparison

4. Web programming

Programming can be briefly defined as the process of developing and implementing various sets of instructions to enable a computer to do a certain task. Essentially, you give the computer small steps of instructions, and the computer goes down the list, executing each one in order.

Web programming, however, refers to the writing, markup and coding involved in Web development, which includes Web content, Web client and server scripting and network security. Web programming is different from just programming, requiring interdisciplinary knowledge on the application area, client and server scripting, and database technology.

A typical web system is organized in three tiers, each running on a separate computer. Logic on the middle-tier server generates pages to send to a front-end browser and queries to send to a back-end database. In the next figure we can see graphically the three-tier model, with the common programming languages of each one in brackets.

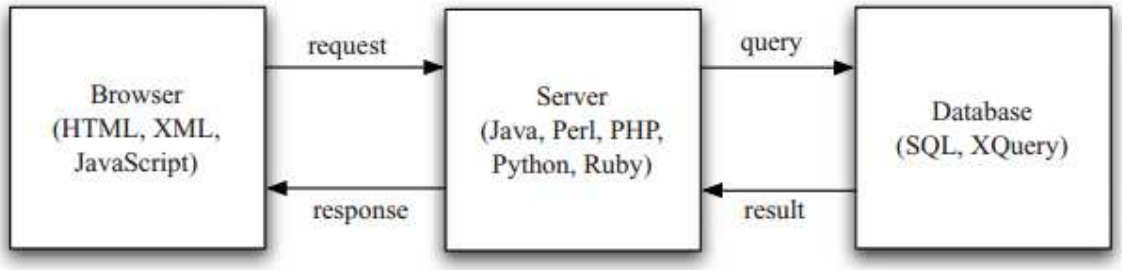

Fig. 20 Three-tier model.

The first tier is the client-side which is also called front-end. The other two tiers are formed by the server and the database, which are known as server-side or back-end.

There is a way to customize web pages and make them more interactive, and that is by mean of scripts. Scripts can be executed on the server side or in the client side. The two can be used together and, actually, this is often done because they do very different things.

Another way of customizing a web page is with AJAX. AJAX stands for **A**synchronous **J**avascript **a**nd **X**ML and is a group of interrelated web development techniques used on the client-side to create asynchronous web applications. With Ajax, web applications can send data to, and retrieve data from, a server asynchronously (in the background) without interfering with the display and behavior of the existing page.

The delivery of the web pages is possible through the **H**yper**t**ext **T**ransfer **P**rotocol (HTTP). The Hypertext Transfer Protocol is an application-level protocol for distributed, collaborative, hypermedia information systems. It is a generic, stateless, object-oriented protocol which can be used for many tasks, such as name servers and distributed object management systems, through extension of its request methods. A feature of HTTP is the typing and negotiation of data representation, allowing systems to be built independently of the data being transferred [17].

HTTP has been in use by the World-Wide Web global information initiative since 1990.

4.1. Client-side

The client is the side in which the web browser is running. Client-side scripts are interpreted by the browser, being JavaScript the main client-side scripting language. The web pages do also utilize a markup language, being HTML the most extended one, and usually a style sheet, being CSS the most widespread.
The steps followed by the browser to display a web-page are the following:

- The user requests a web page from the server.
- The server finds the page and sends it to the user.
- The page is displayed on the browser with any scripts running during or after display.

4.1.1. HTML

Hyper**T**ext **M**arkup **L**anguage (HTML) is the main markup language for creating web pages. The purpose of a web browser is to read HTML documents and compose them into visible or audible web pages. For this purpose the HTML language contains HTML elements consisting of tags, with come in angle brackets and normally in pairs. These tags are keywords which are not displayed by the browser, but offer information of how to display the content.

HTML allows images and objects to be embedded and can be used to create interactive forms. It provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. It can embed scripts written in languages such as JavaScript which affect the behavior of HTML web pages.

Below is a visualization of an HTML page structure:

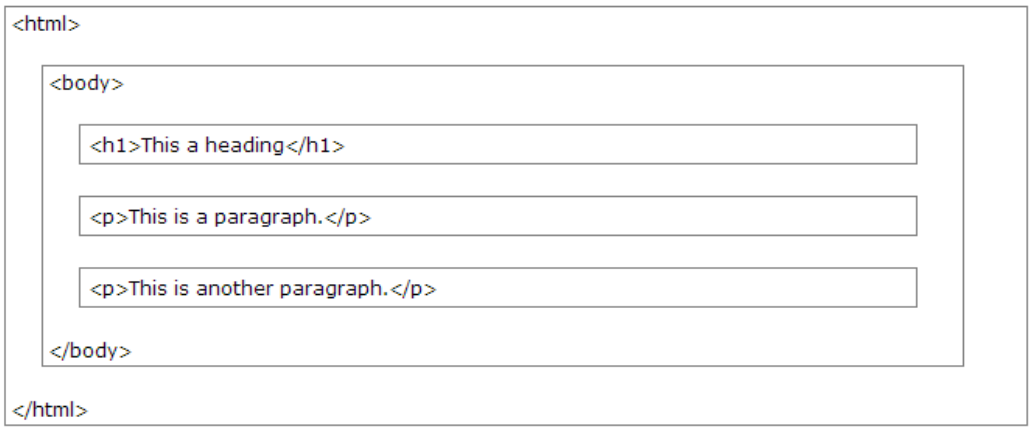

Fig. 21 HTML page structure example

HTML markup consists of several key components, including elements (and their attributes), character-based data types, character references and entity references. Another important component is the document type declaration, which triggers standards mode rendering. The <!DOCTYPE> declaration helps the browser to display a web page correctly. There are many different documents on the web, and a browser can only display an HTML page 100% correctly if it knows the HTML type and version used.

HTML elements, in their most general form have three components: a pair of tags, a start tag and an end tag; some attributes within the start tag; and finally, any textual and graphical content between the start and end tags, perhaps including other nested elements. The HTML element is everything between and including the start and end tags. Each tag is enclosed in angle brackets.

The general form of an HTML element is therefore: *<tag attribute1="value1" attribute2="value2">content</tag>*. Some HTML elements are defined as empty elements and take the form *<tag attribute1="value1" attribute2="value2" >*. Empty elements may enclose no content, for instance, the BR tag or the inline IMG tag. The name of an HTML element is the name used in the tags. Note that the end tag's name is preceded by a slash character, "/", and that in empty elements the end tag is neither required nor allowed. If attributes are not mentioned, default values are used in each case.

Versions

In the following table is shown a summary of the different versions of HTML and the year when they were launched.

Version	Year
HTML	1991
HTML+	1993
HTML 2.0	1995
HTML 3.2	1997
HTML 4.01	1999
XHTML 1.0	2000
HTML5	2012
XHTML5	2013

Fig. 22 HTML different versions

The first release of HTML was launched in 1991, when Tim Berners-Lee, a physicist working at CERN, proposed to create a global hypertext project, which later became known as the **W**orld **W**ide **W**eb (WWW).

HTML+ was published by the IETF as an Internet-Draft and was a competing proposal to become a standard to the HTML draft. Like HTML, it was based on the **S**tandard **G**eneralized **M**arkup **L**anguage (SGML). HTML+ added new features like figures, tables and forms. It also generalized the structures present in HTML to reflect practical experience with writing browsers and a desire to make it easier to convert between HTML+ and other formats [18].

Neither HTML nor HTML+ became an official standard. Was in 1995, when the IETF (**I**nternet **E**ngineering **T**ask **F**orce) published the HTML 2.0 which became an official standard.

Since 1996, the HTML standards are published by the W3C (**W**orld **W**ide **W**eb **C**onsortium). The HTML 3.2 standard, published in 1997, was the first standard published by this organism. This version incorporates the last advances in web pages developed until 1996, like Java applets and text which flows around the images.

HTML 4.0 was published in 1998 and represents a big leap from the previous versions. Among its most prominent novelties are the CSS style sheets, the possibility of including scripts in the web pages, improves the accessibility, complex tables and improvements in the forms. In 1999 an update of this version is launched within HTML 4.01.

Since the publication of HTML 4.01, the W3C focused in the development of XHTML, launching in the year 2000 the first release, XHTML 1.0. XHTML is an advanced version of HTML adapted to the XML language. It maintains almost all the HTML tags and features, but adds some restrictions and elements from XML.

Because of the lack of HTML standards from the W3C since HTML 4.01, in 2004 Apple, Mozilla and Opera created the WHATWG (**W**eb **H**ypertext **A**pplication **T**echnology

Working **G**roup), a group focused on the development of HTML 5.0, which was launched in 2012 and standardized by the W3C.

At the same time the W3C continued with the standardization of XHTML. In the 2013 has published the XHTML 5 version.

4.1.2. CSS

HTML was never intended to contain tags for formatting a document. When tags like <font>, and color attributes were added to the HTML 3.2 specification, it started a nightmare for web developers. Development of large web sites, where fonts and color information were added to every single page, became a long and expensive process.

To solve this problem, the World Wide Web Consortium (W3C) created CSS.

CSS is the language for describing the presentation of Web pages, including colors, layout, and fonts. It allows one to adapt the presentation to different types of devices, such as large screens, small screens, or printers. CSS is independent of HTML and can be used with any XML-based markup language. The separation of HTML from CSS makes it easier to maintain sites, share style sheets across pages, and tailor pages to different environments. This is referred to as the separation of structure (or content) from presentation [19].

CSS Syntax

A CSS rule has two main parts: a selector, and one or more declarations. In the figure you can see the structure:

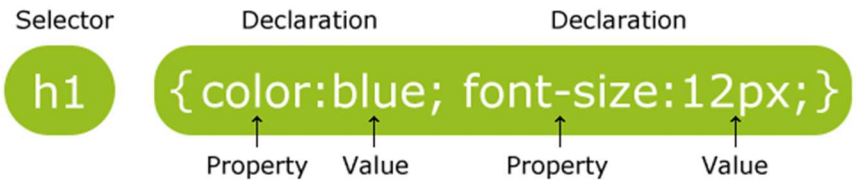

Fig. 23 CSS Syntax example

- Selector: The selector is normally the HTML element you want to style.
- Declaration: Each declaration consists of a property and a value.
- Property: The property is the style attribute you want to change. Each property has a value.

In addition to setting a style for a HTML element, CSS allows to specify selectors for elements with a concrete "id" or "class".

- The **id selector** is used to specify a style for a single, unique element. The id selector uses the id attribute of the HTML element, and is defined with a "#".
- The **class selector** is used to specify a style for a group of elements. Unlike the id selector, the class selector is most often used on several elements. This allows you to

set a particular style for many HTML elements with the same class. The class selector uses the HTML class attribute, and is defined with a ".".

Insertion of a CSS in an HTML document

There are three ways of inserting a style sheet:

- External style sheet: is ideal when the style is applied to many pages. With an external style sheet, you can change the look of an entire Web site by changing one file. Each page must link to the style sheet using the <link> tag. The <link> tag goes inside the head section of the HTML.
- Internal style sheet: An internal style sheet should be used when a single document has a unique style. You define internal styles in the head section of an HTML page, by using the <style> tag.
- Inline style: An inline style loses many of the advantages of style sheets by mixing content with presentation. To use inline styles you use the style attribute in the relevant tag. The style attribute can contain any CSS property.

Versions

Version	Year
CSS 1	1996
CSS2	1998
CSS 2.1	2011
CSS 3	2012

Table 2 CSS versions

4.1.3. Javascript

JavaScript (JS) was invented by Brendan Eich. It appeared in Netscape (a no longer existing browser) in 1995, and has been adopted by ECMA (a standard association) since 1997. ECMA-262 is the official name of the JavaScript standard.

It is an interpreted computer programming language. It was originally implemented as part of web browsers so that client-side scripts could interact with the user, control the browser, communicate asynchronously, and alter the document content that was displayed. More recently, however, it has become common in both game development and the creation of desktop applications.

Insertion of JS in an HTML document

JavaScript in HTML must be inserted between <script> and </script> tags and can be put in the <body> and/or in the <head> section of an HTML page.

Scripts can also be placed in external files. External files often contain code to be used by several different web pages. To use an external script, you must point to the .js file in the "src" attribute of the <script> tag: *<script src="myScript.js"></script>*.

4.2. Server-side

The server is where the Web page and other content lives. The server sends pages to the user/client on request. The process is:

1. the user requests a Web page from the server
2. the script in the page is interpreted by the server creating or changing the page content to suit the user and the occasion and/or passing data around
3. the page in its final form is sent to the user and then cannot be changed using server-side scripting

The use of HTML forms or clever links allow data to be sent to the server and processed. The results may come back as a second Web page.

Server-side scripting tends to be used for allowing users to have individual accounts and providing data from databases. It allows a level of privacy, personalization and provision of information that is very powerful. E-commerce, MMORPGs and social networking sites all rely heavily on server-side scripting.

PHP and ASP.net are the two main technologies for server-side scripting.
The script is interpreted by the server meaning that it will always work the same way. Server-side scripts are never seen by the user (so they can't copy your code). They run on the server and generate results which are sent to the user. Running all these scripts puts a lot of load onto a server but none on the user's system.

5. POS software design

5.1. Technology comparison

The main purpose of the technology comparison is to get a rough idea about the current state of the art concerning the POS (**P**oints **o**f **S**ale). Mobile points of sale (mPOS) are studied in detail as well. Especially, it will be focused on the GUI aspects of the market main players' applications.

Moreover, this report will help to determine which features are suitable for Avance Pay's app and will analyze those features that could be activated and deactivated with the editor to be designed throughout the present thesis.

5.1.1. Key Points of POS software design

Point of Sale software possesses the capability to streamline a retail business, since smooth operations lead to higher profit margins. However, in order to design appropriate POS software some key points should be taken into account [20].

Industry

POS software may be formatted for certain industries. Many solutions market themselves as specific to industries like Sports Equipment, Antiques, or Fashion. However, a significant portion are general solutions that offer all of the same capabilities. Creating a software package specific to an industry will guarantee offering all of the features that are needed.

Company Size

Different software solutions serve different sized companies. Size is an important factor when designing a retail point of sale solution. The growth a company may experience over the next few years should be considered. In this sense the following aspects should be taken into account:

- Should the software be scalable?
- Should it handle more stations and thereby more sales?
- Should it possess features to help the merchant handle multiple currencies?

Deployment Model

On Premise/Client-Server Software follows the Client-Server deployment model. The retailer purchases a physical disk and hosts it on their physical hard drive or server. Once installed, the applications are accessed locally or through the user's network.

Software as a Service follows the Online or SaaS deployment model. It is hosted on the provider's server, and the retailer employs the software through a network connection. In general, you must visit the provider's website for access.

Functionality

POS software contains features to help vendors conduct sales transactions. These features, or functionalities, have recently evolved to address more aspects of the retail industry. Consequently, the term "Retail Management Software" more adequately represents the functions included within POS software. The following are the features that extend the normal POS features:

- Inventory Management
- Customer Management
- Employee Management
- eCommerce
- Retail Accounting
- Reporting

Point of Sale Features

Apply to the actual sales transaction. These features help cashiers process transactions. Common features include receipt printing, gift card creation, and returns.

- Age Verification - Requires the customer's birthdate to approve transactions involving age-restricted items.
- Bar Code Scanning - Offers the ability to retrieve item pricing information using a barcode scanner.
- Buy X, Get Y Pricing - Allows organizations to offer special discounts using the X for Y pricing scheme (i.e. "Two for the price of One!" or "Buy 3, Get 1 Free!").
- Cancellation - Lets cashiers cancel payments after the transaction has occurred.
- Consignment - Offers the ability to manage consigned inventory and sales.
- Coupons - Allows organizations to assign coupons for discounts on certain items.
- Credit Card Processing - Allows credit card swipe as an optional payment method.
- Customer History - Keeps up to date records of customer sales transactions and other information.
- Customizable GUI - Provides the ability for managers to customize the interface to meet organization specific needs.
- Discounts - Allows cashiers to apply discounts to individual items or to the total transaction.
- Exchanges - Lets customers exchange items for different sizes or other in-store items.
- Gift Cards - Offers the ability to create gift cards/certificates which work like debit cards loaded with store credit.
- Items on Hold - Gives the ability to place items on hold and manage them thereafter.
- Layaways & Quotes - Capable of supporting layaway transactions and updating layaway debts.

- Mobile POS Capability - Includes the ability to perform point of sale tasks via a mobile platform.
- Multi-System Integration - Enables the interlinking of multiple stations in order to keep track of total sales and inventory.
- Multiple Currency - Accepts different types of currency at a single point of sale station.
- Multiple Payment Forms - Allows processing of individual transactions with multiple payment methods (i.e. multiple credit/debit cards or a cash and card split).
- Price Lookup - Allows the retrieval of pricing and other information at the point of sale station.
- Print Receipts - Capable of printing receipts containing important transaction information, including individual and total cost of goods, sales tax, payment method, and more.
- Receipt Notes - Allows cashiers to include transaction specific notes on the transaction receipt.
- Refunds - Lets customers receive their money back when they return a faulty or unwanted item.
- Rentals - Capable of processing rental transactions rather than standard sales processing.
- Returns - Allows customers to return items that they have purchased.
- Revenue Totals - Able to display the revenue totals for a single station or an entire store over a given period of time.
- Sales Commissions - Both calculates and updates commissions earned by sales representatives by item or altogether.
- Sell when Server/Network Down - executes transactions when the organization's network or server is down by storing transaction info and uploading it later when the connection is restored.
- Shipping/Delivery Setup - Lets customers and cashiers set up shipping for items that are too large to be carried out of the store.
- Special Orders - Accepts special orders when an item is temporarily unavailable or out of stock.
- Store Credit - Assigns store credit to a customer or card.
- Tax Exemption - Enables cashiers to dismiss tax charges when it is appropriate.
- Time-Limited Specials - Schedules price updates for limited-time sales and special discounts.
- Transaction Hold/Recall - Lets cashiers hold, or pause, transactions to perform other POS functions and then come back to finish them later.
- Void Transaction - Gives cashiers the ability to void whole or partial transactions.

5.1.2. mPOS in the market main features analysis.

First of all, some of the most important features present in several of the main mPOS applications will be explained:

- **Shopping cart:** List containing all the articles a client wants to acquire.
- **Tax rates:** The merchants have the possibility to add a tax rate to each product so that they can afterwards calculate sales tax for the transaction.
- **Tipping:** Merchants can configure the app to automatically prompt customers for a tip.
- **Discounts:** The merchants can apply a discount to any product.
- **Signature:** The clients can sign to verify the transaction.
- **E-mail receipts:** The merchants have the possibility to send a receipt to a client via e-mail.
- **SMS receipts:** The merchants have the possibility to send a receipt to a client via SMS.
- **Sales history:** The merchants can check the sales history in the app.
- **Reports:** The merchants can access reports.
- **Real-time sales volume:** The merchants can access sales reports in-the-app and at real time.
- **Inventory management:** The merchants can import their existing inventory database in a snap. Once the inventory has been uploaded, the system will automatically track how much they have in stock.
- **Save customers info:** Merchants can save customer information such as names and addresses. This may be helpful to identify sales trends and best customers, ideal for loyalty and other incentive programs.
- **Manage cash transactions:** Merchants can manage and record cash transactions within the phone application.
- **Refund:** Merchants can refund the clients.
- **Password protected:** The application is protected with a user and password screen for security purposes.

In the following table some of the major players' applications features are compared:

Feature	Square	iZettle	SumUp	Revel	AccuPOS
Shopping cart	✔	✔	✔	✔	✔
Tax rates	✔	✔	✔	✔	✔
Tipping	✔	✔	✔	✔	✔
Discounts	✔	✔	✔	✔	✔
Signature	✔	✔	✔	✔	✘
E-mail receipts	✔	✔	✔	✔	✘
SMS receipts	✔	✘	✔	✔	✘
Sales history	✔	✔	✔	✔	✔
Reports	✔	✔	✔	✔	✘

Real-time sales volume	✗	✗	✗	✓	✓
Inventory management	✗	✗	✗	✓	✓
Save customers info	✗	✗	✗	✓	✓
Manage cash transactions	✓	✓	✓	✓	✓
Refund	✗	✗	✗	✓	✓
Password protected	✓	✓	✓	✓	✓

Table 3 mPOS main features comparison

5.1.2.1. Square

Square brings mobile credit card processing to the masses with a miniscule square credit card reader that plugs into the audio jack of popular mobile devices. The app itself is free and there is no monthly service fee. You don't pay for a merchant service because the service works with your existing bank account.

Fig. 24 Square's app for iPhone

Square's application GUI is really simple. Its main screen consists of an "Insert amount" screen where you insert the amount you want to charge. Here you can add a description of the sale and even attach a picture. If you have the card inserted into your device you can directly swipe a credit card to charge the amount. If not, you can click the button that appears in the upper right corner to even choose to insert the credit card details manually or to register the sale as a cash payment.

You can always add more items, even by inserting them manually or by selecting them from the item library if you have previously added one.

Once the encash has been performed, a signature screen appears so that the client can sign the transaction. This screen is optional; this means that you can deactivate the signature process from the settings menu. The same thing happens with the tipping. If you have activated them the clients will be able to select the percentage they want to give as tip or even insert the amount they want to give manually.

Once the payment has been successfully performed, the user can select whether he wants to receive the receipt even to his email account or by SMS.

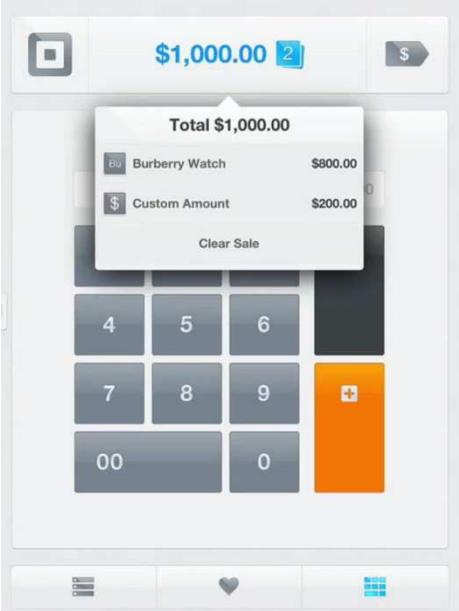

Fig. 25 Square's app shopping cart detail

As shown in the figure, the shopping cart for small devices appears in the top of the screen, as well as in the iPad vertical layout screen. The total amount is shown as well as the number of items selected, appearing within a little box. When clicked, a summary of the sales is displayed.

The shopping cart for the iPad landscape layout is displayed in detail in the right side of the screen.

The insertion of new items is really simple. In the "All items" screen there is an "Edit" button, which after being clicked allows the merchant to edit the items library. A "+" button appears where the Edit button was. After clicking it a "new item" pop-up screen pops up. In this pop-up the merchant can add a new product, selecting a name, a category, a price and it is even possible to add a photo or assign the product a color, a text, etc.

The merchants also have the possibility to add Reward and discount buttons. The procedure to create this discount buttons is very similar to the one of creating a new product.

The iPad version offers extra functionality that the android mobile phones and iPhones don't support. This extra functionality is for instance the possibility to have in-app reporting, print receipts or kitchen tickets, or have a cash drawer.

Fig. 26 iPad landscape layout

5.1.2.2. iZettle

iZettle has two clear target markets. The first is the small business provider who wants to begin offering card payment options to his/her customers. These users will apply for a 'Business Account'.

The other target market is for home users who want to be able to take card payments. These users will apply for a 'Personal Account'.

The difference between the two accounts is the amount you can take per transaction and total amount per day.

iZettle's app appearance is quite similar to Square's app. The main screen consists of a numeric keyboard where the merchant can type the amount to charge. It is possible to add a description of the sale as well as a picture.

Fig. 27 iZettle's app and reader

Once you have typed the amount, you can click on the pay button which is in the top right corner of the screen. You can choose among inserting the credit card details manually, entering a cash payment, to track all your incomes, or inserting a MasterCard credit card in the reader.

Although VISA payments are allowed, the payment procedure is different. If you insert a VISA card a SMS will be send to the owner of the card containing a URL. Once the buyer clicks on the URL link, the browser opens a web site where he can fill his VISA card details to proceed with the payment.

After completing this process the payment follows as in the MasterCard payment: a signature screen appears so that the client can sign. After the verification, there is a "send receipt" screen where the customer can introduce his email address in order to receive an electronic receipt.

There is also another way to perform a credit card payment and is with iZettle's PIN & CHIP device. This device is connected via Bluetooth with the device where the merchant has the app installed.

The customer inserts the card into the card slot. Then he only has to introduce his PIN and confirm the payment. If it results into a successful payment the confirmation will appear on the merchant's device.

Fig. 28 iZettle's PIN & CHIP

Fig. 29 iZettle's application

Furthermore, iZettle's app allows the merchant to create items and to access them from the library. It also allows vendors to organize the items by folders.

The method to insert new items is really similar to Square's one.

5.1.2.3. SumUp

SumUp's service, as Square's and iZettle's, includes a free card reader, phone app, and account setup as well as a no other fees in addition to a flat processing fee of 2.75% for all card types and transactions.

Fig. 30 SumUp's app and reader

The user interface is similar to Square's and iZettle's apps, but offers some differences. The main screen is again a numeric key board where the merchant can introduce the amount of the purchase.

After having inserted an amount, it can be charged by clicking the "Charge" button. The app will ask for the payment method to be used in the following screen. If a card payment is selected and the reader is connected to the device the app will ask the buyer for inserting the credit card. It doesn't recognize it directly, but the merchant must press the "read card" button.

Afterwards, a signature screen will appear so that the buyer can sign. He can even give a tip if desired.

Once the card has been successfully read and the transaction has been performed, a screen to send an email or SMS receipt will show up. The customer can insert his email address and he will receive the receipt.

On the other hand, the vendor can create his own items specifying the name, price, and category. He can also attach a picture to it and finally post it to the "shelf", where all the created items will appear.

Once he wants to sell some item from the library he goes to the shelf and selects as many items as he wants. A number will appear in the upper right part of the item image within a little colored square, indicating the number of times the item has been added to the shopping cart. When he has finishing adding products he can review the products added to the cart by clicking the blue button on the upper right part of the screen.

There is the possibility to activate and deactivate the tip and tax rates from the settings, as well as manage the products of the shelf. Moreover, the merchant has access to the sales history and to email and social network support.

5.1.2.4. Revel

The Revel POS works in a hotkey fashion, in which cashiers can tap on menu categories with drop-downs of items and modifiers to select. Or if you prefer, barcode numbers can be entered into the system for ringing of items with a Bluetooth wireless barcode scanner. Tax, discounts, service fees, and surcharges can be set to automatically add or manually add at the

time of sale. All payment tenders are acceptable: cash, check, gift certificate, credit card, debit card, and even customer rewards points. To pay with exact change requires hitting simply the "cash" button. Manual credit card entry is also a payment option. Once an item is added to a transaction it can be easily removed or deleted simply by tapping on it and tapping "delete." If an item has modifiers attached to it, the option for modifiers will automatically pop-up when the item is selected. In the "edit item" screen cashiers can set orders to-go, for delivery, repeat items, remove items, enter discounts, enter special requests, add quantities, or even pay.

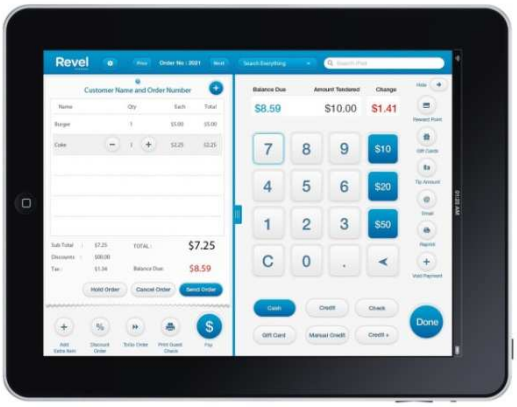

Fig. 31 Revel's POS app

Editing modifiers is as easy as un-selecting what you've selected and reselecting whatever you want. When completing an order cashiers can also add the customer's name to the order (for quick service environments). Of course the POS can also void payments, allow for tips, split bills, email receipts, print/reprint guest receipts, and enter rewards card numbers. Certain functions like discounting an order, opening the cash drawer, or voiding payments can have manager password requirements if you prefer. If an order has already been sent to the kitchen you even have the option of going back to it and adding additional items, sending then only the newly added items to the kitchen. Obviously items already sent cannot be edited or deleted, but they can be repeated, discounted, increased in quantity, or voided off the bill.

With Revel you can create multiple price levels, including time-based promotional prices (i.e. sales) or promotional campaigns like buy one get one free, mix and match, and 1/2 off.

Revel's POS supports lots of features, but could result complex.
In Europe, Revel is integrated with Sum Up, a mobile credit card processor.

5.1.2.5. AccuPOS


Fig. 32 AccuPOS' Android app

AccuPOS is a PC-based POS software, but this company offers also an Android-based version of their software.

This company focuses in offering a useful GUI and professional software more than in making it attractive for the sight. The layout of the application consists of buttons added by the merchant, grouped by colors. This way of showing the different items is the most pragmatic approach.

Then, the application of this company offers all kind of features, really valuable for merchants, but they will not be described because these features are mostly offered by their PC software. The only purpose of including this software to the review is to

show the appearance of their user interface, which, as mentioned before, focuses on the usefulness and not in the appearance.

5.1.3. mPOS features comparison table

The following table shows a summary of some mPOS features.

	Square	iZettle	SumUp	RevelSystems	AccuPOS
Features	• Receipts by SMS and email • Signature verification • Tips and taxes • Reporting • Add product • Discounts	• Receipts by email • Signature and PIN verification • Tips and taxes • Reporting • Add products • Discounts	• Receipts by email • Signature verification • Tips and taxes • Reporting • Add products • Discounts	• Only for tablets • Professional software	• Professional software • Arrange buttons by categories and color-based grouping
Type of device	Android devices, iPhone, iPad	Android devices, iPhone, iPad	Android devices, iPhone, iPad	iPad	PC, Android mobile devices
Type of business	Retailers	Retailers	Retailers	All	All
Pros	• Small card reader • Own wallet app	• CHIP&PIN device is really useful	• Easy to use	• Professional software • Own wallet app	• Professional software • Very complete • Multiple reports
Cons	• Reader only allows to swipe cards	• Payments with VISA cards are annoying. Card data must be inserted.	• Only for small merchants.	• May be difficult learn how to manage every feature the app offers.	• Not appropriate for small merchants for being too complex

Table 4 mPOS comparison

5.2. Definition of requirements

5.2.1. Customer Segmentation

First of all, is necessary to accomplish a customer segmentation in order to decide which customers we want to focus on and which features apply for which type of customer. For this purpose a workshop is required.

After studying the market possibilities these are the players that come up in the workshop meeting for defining the customer segments:

- **Small merchants:** Merchants with few sales a day and not a high volume of sales.
- **Event Merchants:** Merchants with are selling in events such as concerts, football matches, etc.
- **Retailers:** Merchant with a higher volume of sales than small merchants, requiring a more professional sales solution.
- **Workman:** Person who offers a service and works by his own.
- **Workman Company:** Company that counts with various workmen.
- **Tourism:** City tickets, normally to enter museums and access to special services.
- **Public transportation:** Taxis, buses, underground, tram, etc.

Use Case	Description	Used HW	Features / Keywords
Issuing Device	Issuing card	Smartphone / Tablet	• Load / activate • Different payment methods, incl. cash and online vouchers
Loading Device	Converting money	Smartphone	• Load / activate • Different payment methods, incl. cash and offline vouchers
CRM Device	Customer relationship management	Smartphone / Tablet	• Register cards • Complains • Refunding • Blocking
Small Merchants	Selling anything a few times a day	Smartphone Optional: • Cash Drawer • Receipt Printer • Customer Display	Gamification / Standalone • Hotkeys • Product groups • Specific amounts • Shopping cart • Discounts • Tipping • Different payment methods, incl. cash • (VAT) • Loyalty • (Coupons)
Event Merchants	Selling small assortment many times	Tablet	Fast & Simple GUI; full (integration with) cash register • Hotkeys • Product groups • Specific amounts

			• Shopping cart • (Discounts) • Tipping • Single payment method • Different payment methods, incl. cash & vouchers • (VAT)
Retail	Selling many things many times	Tablet	Professionals; full (integration with) cash register • Scanning • Article Numbers • Specific amounts • Shopping cart • Discounts • (Tipping) • Different payment methods, incl. cash & vouchers • Loyalty • Coupons • VAT
Workman	Runs his own business	Smartphone	3x Simple • Calculator • Tipping • VAT • Different payment methods, incl. cash
Workman company	Controlled by his boss	Tablet / smartphone	3x simple • Working report • Scanning • Tipping • Scheduling • Dispatching • Different payment methods, incl. cash • VAT • CRM integration
Tourism		Tablet	Standalone • Pass activation and verification –or– • Discount verification –or– • Payment / Points • Reporting

Taxi	POS for a taxi	Smartphone	3x Simple • Calculator • Tipping • VAT Different payment methods, incl. cash
Bus	Selling fixed prices tickets	Smartphone	3x Simple • One ride • Multiple rides (10x) Different payment methods, incl. cash

Table 5 Customer segmentation

Once the customer segments have been defined, a feature matrix containing the previously gathered mPOS features and the customer segments is created in order to define which features apply for which customer segment.

5.2.2. Features matrix

Feature	Affects GUI	Definition	Issuing device	Loading device	CRM device	Events merchants	Small merchants	Retail	Workman	Workman company	Tourism (City Ticket)	Taxi	Bus (Driver)	Bus (automated)
OFFERING FEATURES														
Age Verification	YES	Requires the customer's birthdate to approve transactions involving age-restricted items.	NO	NO	NO	OPTIONAL	OPTIONAL	OPTIONAL	NO	NO	OPTIONAL	NO	OPTIONAL	OPTIONAL
Bar Code Scanning	YES	Offers the ability to retrieve item pricing information using a barcode scanner.	NO	NO	NO	NO	NO	YES	NO	OPTIONAL	NO	NO	NO	NO
Price Lookup	YES	Allows the retrieval of pricing and other information at the point of sale station.	NO	NO	NO	NO	YES	YES	NO	OPTIONAL	NO	NO	NO	NO
Hotkeys	YES	Allow the merchant to create a library with the most sold products.	NO	NO	NO	OPTIONAL	YES	NO	OPTIONAL	OPTIONAL	NO	NO	NO	NO
Product groups	YES	The products can be grouped by category.	NO	NO	NO	OPTIONAL	YES	NO	NO	OPTIONAL	NO	NO	NO	NO
Specific amounts	YES	Offers the ability to configure buttons with the most useful default amounts payed by the client.	NO	NO	NO	YES	YES	YES	NO	NO	NO	NO	NO	NO
Shopping cart	YES	The selected items are added to a shopping list.	NO	NO	NO	YES	YES	YES	NO	OPTIONAL	NO	NO	NO	NO
Numeric keyboard	YES	Offers the possibility to insert an amount by typing the numbers.	NO	NO	NO	YES	YES	YES	OPTIONAL	OPTIONAL	NO	YES	NO	NO
Calculator	YES	Offers the possibility to perform numeric operations to get the final price to charge.	NO	NO	NO	NO	NO	NO	YES	NO	NO	NO	NO	NO
PAYMENT FEATURES														
Buy X, Get Y Pricing	NO	Allows organizations to offer special discounts using the X for Y pricing scheme (i.e. "Two for the price of One!" or "Buy 3, Get 1 Free!").	NO	NO	NO	OPTIONAL	OPTIONAL	OPTIONAL	NO	NO	NO	NO	NO	NO
Coupons	YES	Allows organizations to assign coupons for discounts on certain items or overall. [RESTRICTED] Bonus overall (e.g. 5 EUR off when purchase>20 EUR). [RESTRICTED] Bonus on one item (e.g. 5 EUR off when items acquired > 5).	NO	NO	NO	OPTIONAL	OPTIONAL	YES	OPTIONAL	OPTIONAL	YES	OPTIONAL	NO	NO
Discounts	YES	Allows cashiers to apply discounts to individual items or to the total transaction.	NO	NO	NO	OPTIONAL	YES	YES	OPTIONAL	OPTIONAL	NO	NO	NO	NO
Time-Limited Specials	NO	Schedules price updates for limited-time sales and special discounts.	NO	NO	NO	NO	OPTIONAL	OPTIONAL	NO	NO	NO	NO	NO	NO
Loyalty	YES	Loyalty programs.	NO	NO	NO	NO	YES	YES	OPTIONAL	NO	NO	NO	NO	NO
Multiple Currency	YES	Accepts different types of currency at a single point of sale station.	NO	NO	NO	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL
Tips	YES	Allows the customer to give a tip if desired.	NO	NO	NO	YES	YES	OPTIONAL	YES	YES	NO	YES	NO	NO
Multiple Payment Forms	YES	Allows processing of individual transactions with multiple payment methods (i.e. multiple credit/debit cards or a cash and card split).	YES	YES	NO	NO	YES	YES	YES	YES	NO	YES	OPTIONAL	NO
TRANSACTION FEATURES														
Cancellation / Reversal	YES	Lets cashiers cancel payments after the transaction has occurred.	NO	NO	NO	YES	YES	YES	YES	YES	YES	YES	YES	NO
Exchanges	YES	Lets customers exchange items for different sizes or other in-store items.	NO	NO	NO	NO	OPTIONAL	YES	NO	NO	NO	NO	NO	NO
Refunds	YES	Lets customers receive their money back when they return a faulty or unwanted item.	NO	NO	YES	NO	OPTIONAL	YES	NO	NO	OPTIONAL	NO	NO	NO
Rentals	YES	Capable of processing rental transactions rather than standard sales processing.	NO	NO	NO	NO	OPTIONAL	OPTIONAL	NO	NO	NO	NO	NO	NO
Returns	YES	Allows customers to return items that they have purchased.	NO	NO	NO	NO	NO	OPTIONAL	NO	NO	NO	NO	NO	NO
Load	YES	Allows customers to load value to their cards.	YES	YES	NO	NO	OPTIONAL	OPTIONAL	NO	NO	NO	NO	NO	NO
Activate	YES	Allows the activation of a card.	YES	YES	NO	NO	OPTIONAL	OPTIONAL	NO	NO	YES	NO	NO	NO
Register cards	YES	Registers a card to the system.	NO	NO	YES	NO	NO	NO	NO	NO	NO	NO	NO	NO
Blocking	YES	Allows the merchant to block a card.	NO	NO	YES	NO	NO	NO	NO	NO	NO	NO	NO	NO
EXTRA FEATURES														
Print Receipts	YES	Capable of printing receipts containing important transaction information, including individual and total cost of goods, sales tax, payment method, and more.	NO	NO	NO	NO	YES	YES	OPTIONAL	OPTIONAL	NO	YES	OPTIONAL	OPTIONAL
Receipt Notes	YES	Allows cashiers to include transaction specific notes on the transaction receipt.	NO	NO	NO	NO	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	NO	OPTIONAL	OPTIONAL	OPTIONAL
Multi-System Integration	NO	Enables the interlinking of multiple stations in order to keep track of total sales and inventory.	YES	YES	YES	YES	OPTIONAL	YES	NO	YES	OPTIONAL	OPTIONAL	YES	YES
Customer History	ONLINE SERVICE	Keeps up to date records of customer sales transactions and other information.	NO	NO	NO	OPTIONAL	YES	YES	NO	NO	NO	NO	NO	NO
Revenue Totals	ONLINE SERVICE	Able to display the revenue totals for a single station or an entire store over a given period of time.	NO	NO	NO	YES	YES	YES	NO	OPTIONAL	NO	OPTIONAL	OPTIONAL	OPTIONAL
Shipping/Delivery Setup	YES	Lets customers and cashiers set up shipping for items that are too large to be carried out of the store.	NO	NO	NO	NO	NO	OPTIONAL	NO	NO	NO	NO	NO	NO
Sell when Server/Network Down	NO	Executes transactions when the organization's network or server is down by storing transaction info and uploading it later when the connection is restored.	YES	YES	YES	YES	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	YES	YES	YES
Transaction Hold/Recall	YES	Lets cashiers hold, or pause, transactions to perform other POS functions and then come back to finish them later.	NO	NO	NO	NO	OPTIONAL	OPTIONAL	NO	NO	NO	NO	NO	NO
Special Orders	YES	Accepts special orders when an item is temporarily unavailable or out of stock	NO	NO	NO	NO	OPTIONAL	OPTIONAL	NO	NO	NO	NO	NO	NO

Table 6 Features matrix

After defining the features for each customer segment, some GUIs are proposed. The classical lifecycle for small merchant mPOS is the following:

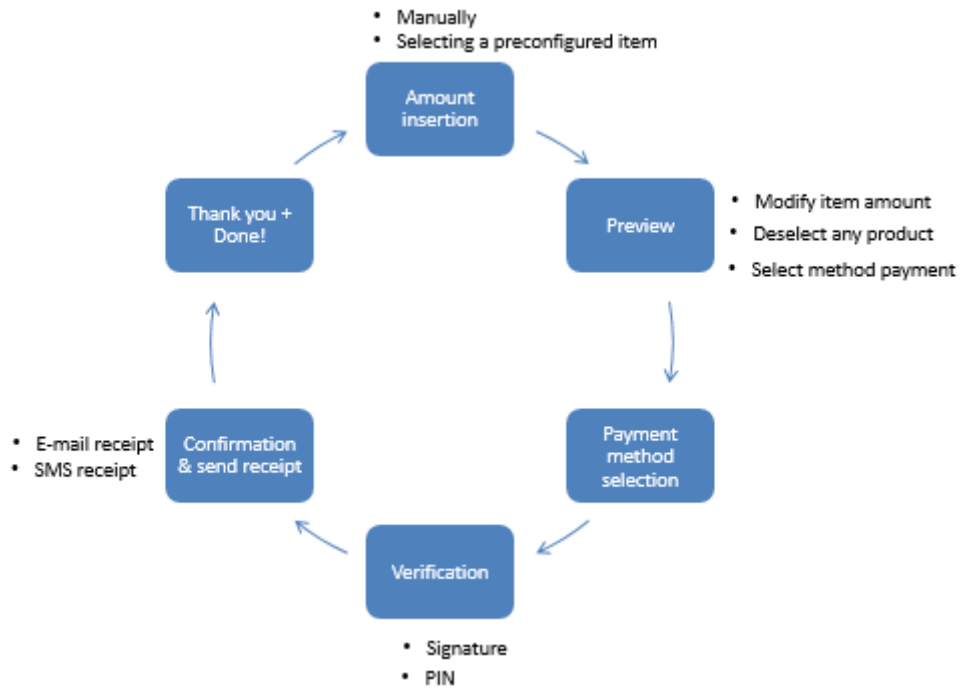

Fig. 33 mPOS screens flow

Based on this schema the small merchants GUI proposal design for the mPOS application is created. The following are some examples of the designed screens:

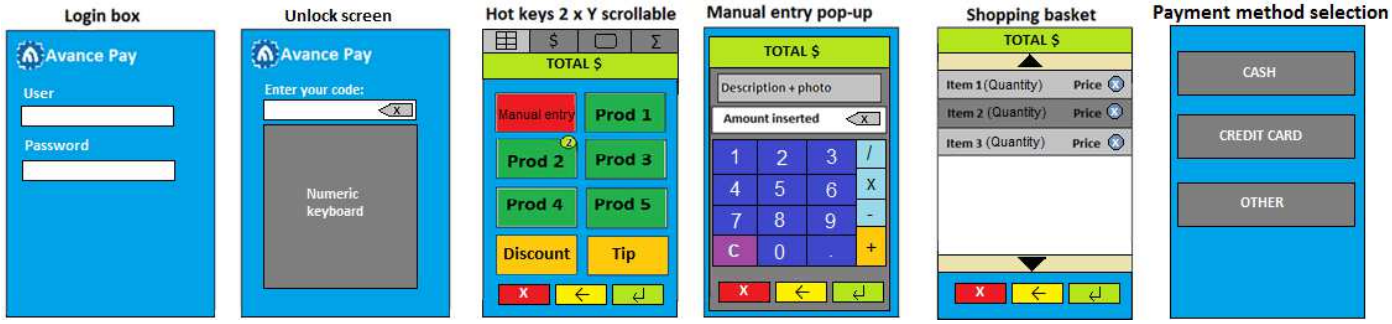

Fig. 34 mPOS layouts proposal

Some designs were proposed to create an mPOS application for various customer segments. However, we realized that the better approach would be to focus on one of those. This decision was taken considering market preferences of the company. Therefore, since this point we focus on Event Merchant mPOS.

5.2.3. Event Merchant mPOS design

Avance Pay's **Point of Sales** for event Merchants is intended to be an application for Android Tablets that allows Event Merchants to manage their sales.

The design of the GUI consists of a grid-based layout. The grid utilized will have a dimension of 7 (vertical) by 9 (horizontal). In its turn each grid square will contain an internal grid of 3 (vertical) by 4 (horizontal).

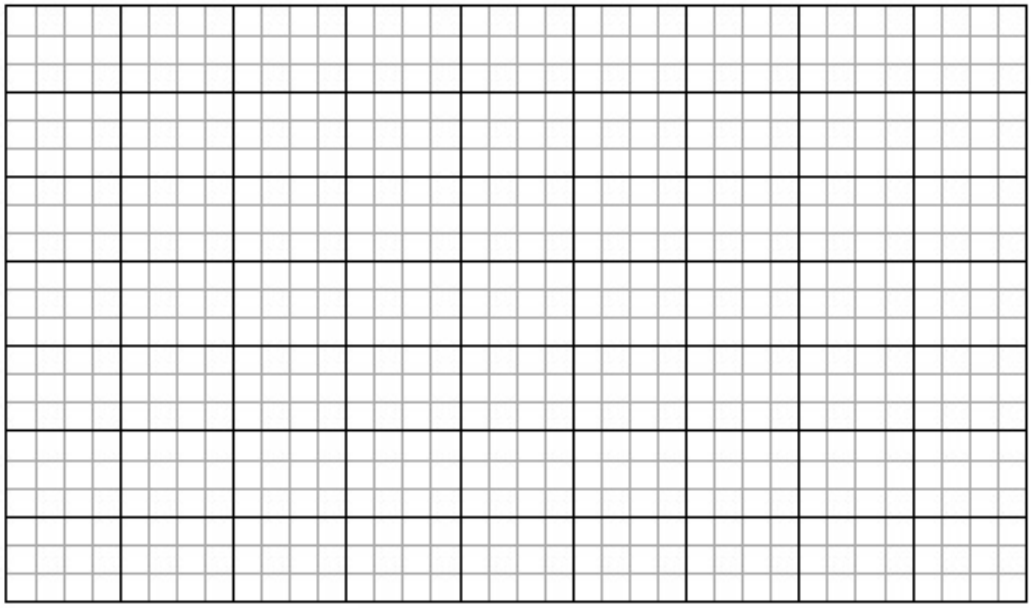

Fig. 35 Tablet grid

All the screens of the applications have been designed using the above grid. Hereafter there is an example of how the login screen has been designed:

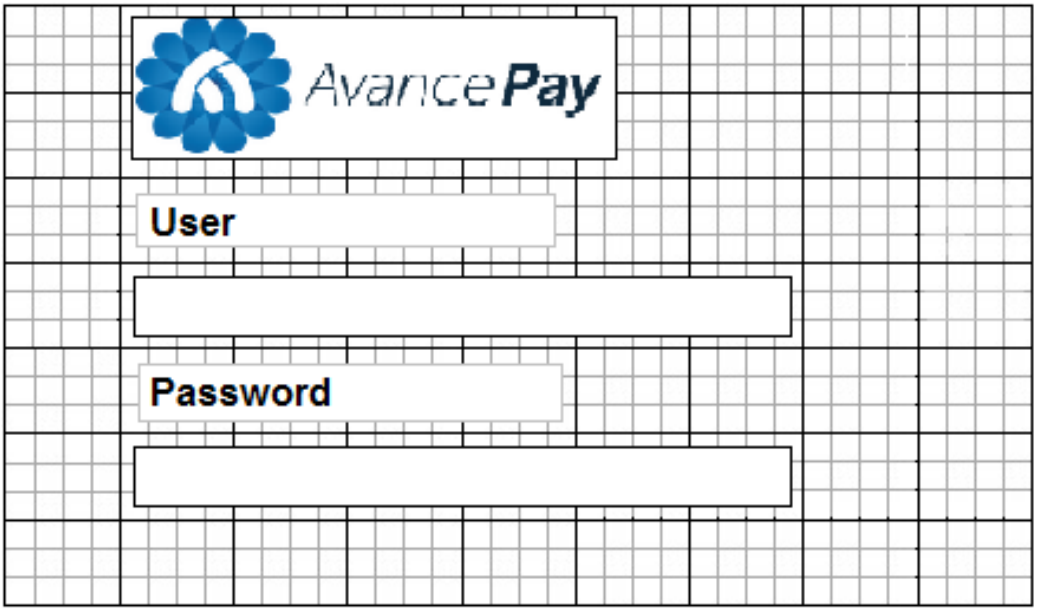

Fig. 36 Tablet mPOS login screen design

The following image shows the expected final result:

Fig. 37 mPOS login screen

The main screen of the application will be the sales screen, where the merchant can manage his sales operations. This will the first screen to be opened after doing the log in.

In the left side will be a tab control, where the user can select the screen to be displayed. The tab with the shopping cart opens the main screen, which is the sales screen. As you can appreciate, each element of the design has a defined size; easily assigned using the grid and the same applies for the position of each element.

Sales Screen

The following picture illustrates the five main areas in which this screen is divided:

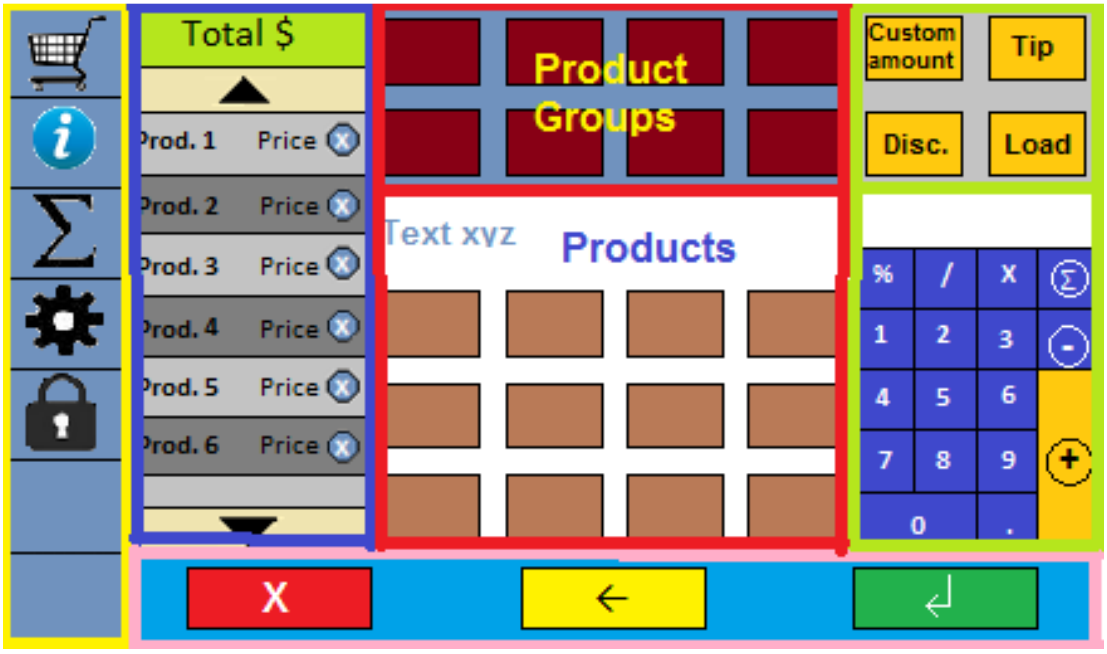

Fig. 38 mPOS sales screen

The areas will be measured by squares of the grid taken up. Nomenclature: Y (vertical) by X (horizontal) number of squares. **Y x Z**.

1. **Tab control:** The tab control will allow the users to control the screen to be displayed. This area will take up the whole first row, being each tab one square of the grid. Therefore, there will be a maximum of seven tabs. Dimension: **7 x 1**.
2. **Shopping cart:** the shopping cart will consist of a list of the selected products and a textbox showing the total amount to be paid. In the image it is highlighted by the blue square. Dimension: **2 x 6**.
3. **Products:** The products screen is marked in the picture with red. It consists of two sub-sections: the "Product Groups" and the "Products". Once the merchant selects a category; the corresponding products belonging to that category will appear in the "Products" area.
   Dimension: **6 x 4**.
   -Product Groups: **2 x 4**.
   -Products: **4 x 4**.
4. **Calculator:** the calculator function will be controlled by 4 radio buttons. The 4 buttons will allow the user among inserting a manual entry, entering a tip or a discount or loading a card. It is highlighted in the picture in green in the right part of the layout. Dimensions: **6 x 2**.
5. **Control buttons:** The control buttons are three buttons at the bottom of the screen, which allow the user to cancel a transaction (red button), undo the last operation (yellow button) or go to the next screen (green button). Dimension: **1 x 8**.

Information screen

This screen will contain Avance Pay's contact information and support, so it will be probably based on a web page screen. It will not be editable.

Transactions screen

This screen is out of scope for the present thesis while it is a not editable screen. It will show the transactions realized in the terminal.

Settings

This screen is out of scope for the present thesis while it is a not editable screen. It will show a screen with some setting options.

Lock screen

The lock screen will consist of a numeric keyboard, a text box and Avance Pay's logo. As shown in the following picture, the same grid is utilized to determine the position of the elements.

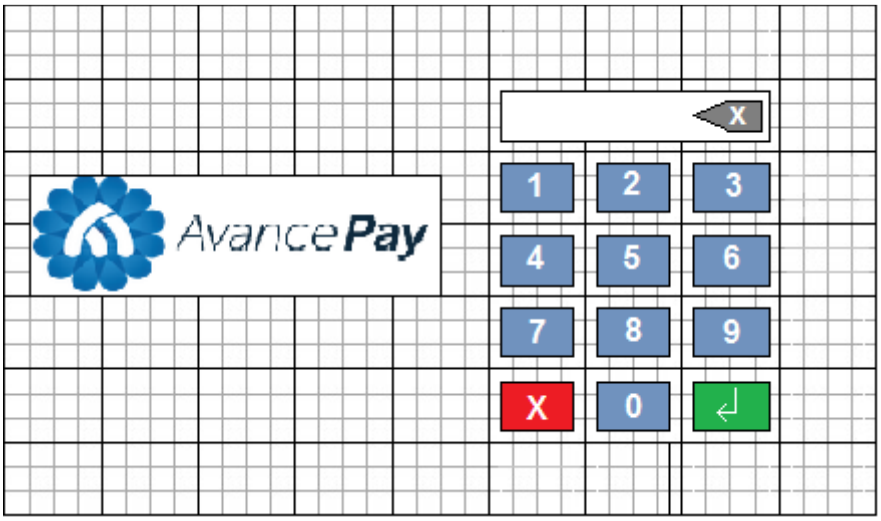

Fig. 39 mPOS lock screen design

The final screen will look like the following picture:

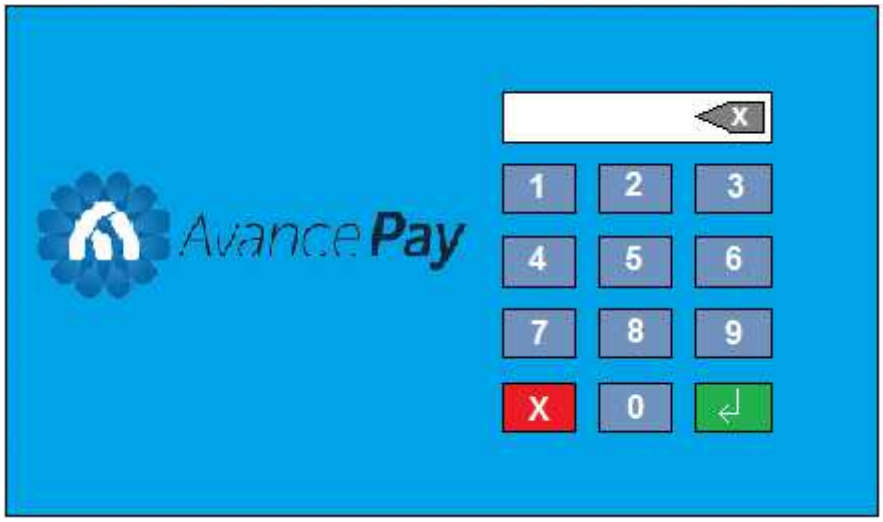

Fig. 40 mPOS lock screen

The numeric keyboard will allow users to insert the unlock code PIN. The red button resets the textbox and the green button confirms the inserted number.

5.3. Prototype creation

Once the GUI design is clearly defined, the idea is to create a functional prototype of the application. The technology to be used to create it is JavaFX, which is a cross platform GUI toolkit for Java. The functionality of the application will be programmed in Java.

The layout of the screens have been created with JavaFX Scene Builder. It is a visual layout tool that lets users quickly design JavaFX application user interfaces, without coding. Users can drag and drop UI components to a work area, modify their properties, apply style sheets, and the FXML code for the layout that they are creating is automatically generated in

the background. The result is an FXML file that can then be combined with a Java project by binding the UI to the application's logic.

The bellow image shows how this editor looks like:

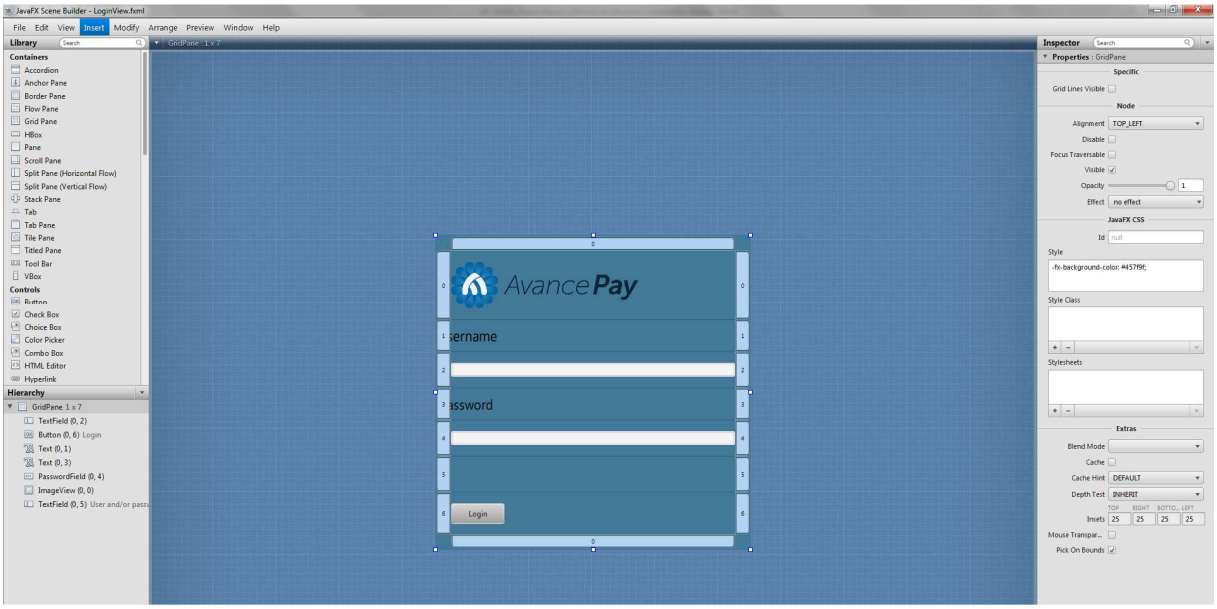

Fig. 41 JavaFX Scene Builder

In the left side of the program you can find the

The result for the login screen is the following:

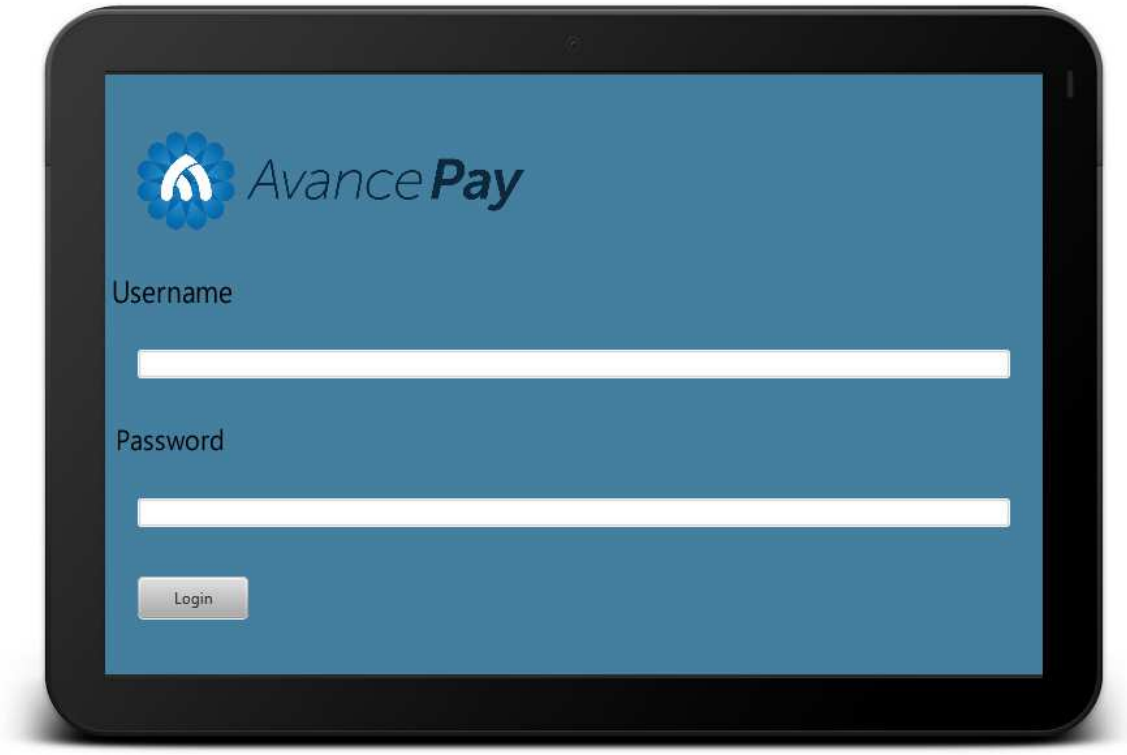

Fig. 42 Login screen - Prototype

Then, the Sales Screen has been created. The following picture shows how it looks like. It respects the previously designed layout.

Every single button has been implemented, carrying out their tasks. Furthermore, the category buttons and their corresponding products are loaded by reading a configuration xml file, as the real app will do it.

Fig. 43 Sales screen - Prototype

The following shows how the lock screen looks like.

Fig. 44 Lock Screen - Prototype

6. Design and development of the Drag & Drop Editor

In this section the design and development of the drag and drop application are explained. The requirements are firstly explained as well as the proposed solutions.

6.1. Requirements

Some of the requirements were stipulated from the outset: It had to work for web browsers, as a standalone application and for android devices.

Moreover, in order to reuse code, a common programming language had to be chosen. For this reason the use of standardized web APIs came into the spotlight.

This would be possible if the android application and the standalone application somehow could display the web content. For that reason, the idea was to use a similar approach to the PhoneGap's one, and use android's web browser engine to render the drag & drop application. In the same way the standalone application had to render the web content using a web engine.

The chosen technology to create the standalone application was JavaFX, because it had been used beforehand and was, therefore, known. Furthermore, its WebView could be used to display the content as for the android app.

WebView is a Node that manages a web engine and displays its content. The associated WebEngine is created automatically at construction time and cannot be changed afterwards. WebView handles mouse and some keyboard events, and manages scrolling automatically, so there is no need to put it into a ScrollPane [21].

The WebView enables the display of the frontend content. However, the intention was to use the whole application without requiring an internet connection. For that reason, the solution proposed was to insert the open-sourced and commercially usable web server, Jetty, into the standalone application; and i-Jetty into the android application.

Jetty provides a Web server and javax.servlet container. Moreover, these components are open source and available for commercial use and distribution, converting Jetty in the ideal choice. Other features of Jetty are: small footprint, enterprise scalable, embeddable, full featured and standard based [22].

Hereafter there is a summary of the technology used in the different platforms:

- **Web Browser:** Use of HTML, CSS and JavaScript for the front-end. Two JavaScript-based frameworks have been used: KineticJS and JQuery.
  For the backend Java was the choice.
- **Standalone application:** JavaFX with the WebView node to display the frontend content and integrated with Jetty to manage the servlets.

- **Android application:** Android app using android WebView for the front-end and i-Jetty for the backend servlets.

6.1.1. Web application solution

The web application is intended to be an online platform where Avance Pay's users can access their accounts in order to edit their POS applications. In this section a solution to address this project is proposed.

The schema in mind consists of a front-end side, a back-end side and a terminal with a POS application. The users might connect via web-browser to Avance Pay's platform. From the platform they may have access to the current configuration of their terminals.

They could make the changes they want and once they have saved the new configuration file it could be updated to their android devices via FTP.

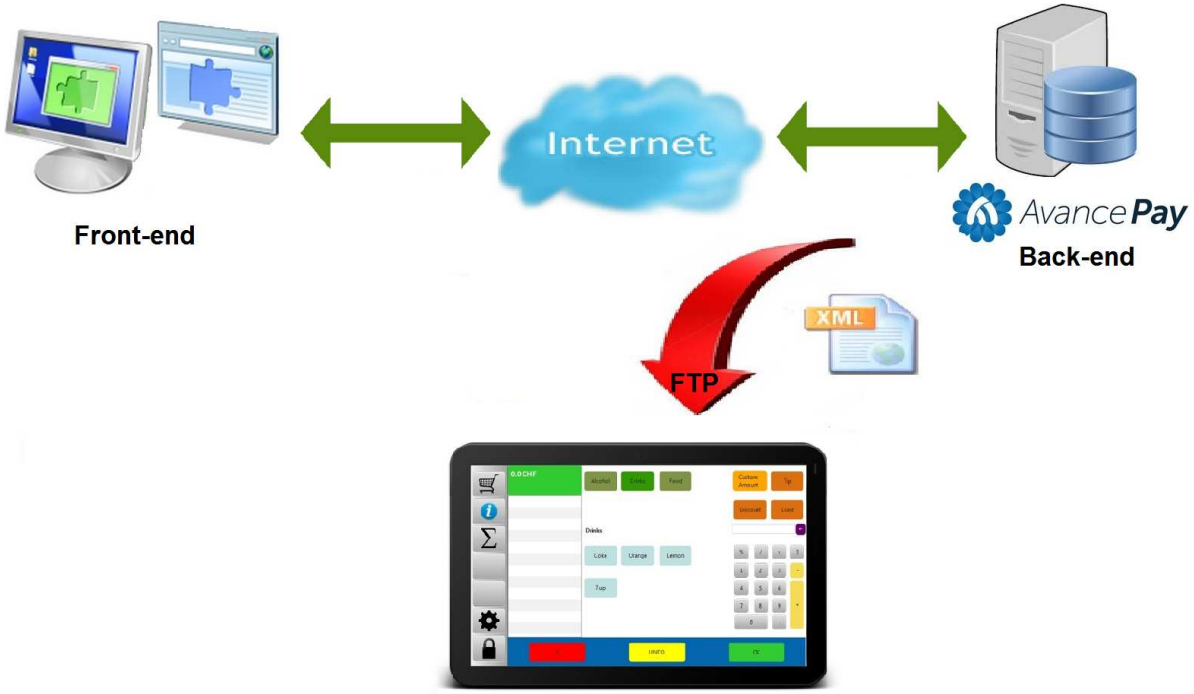

Fig. 45 Solution for the web application

In the image above the proposal is explained graphically. The front-end solution will be the same as for the standalone and android applications, HTML5, CSS3 and JavaScript. The technology to handle the creation of such configuration files will be Java.

6.1.2. Standalone application solution

As mentioned before, the standalone had to work as a complete solution. This means, it had to work in offline mode, permitting the users to load a pre-configured file or to create a new configuration.

In the following figure the proposed solution is illustrated to make it easier to understand:

Fig. 46 Solution for the standalone application

The idea was to integrate the front-end and the back-end within a Java application. The GUI part would be managed using JavaFX making use of its WebService engine to display the front-end content. The back-end code would be handled by Jetty, where the XML configuration file is generated.

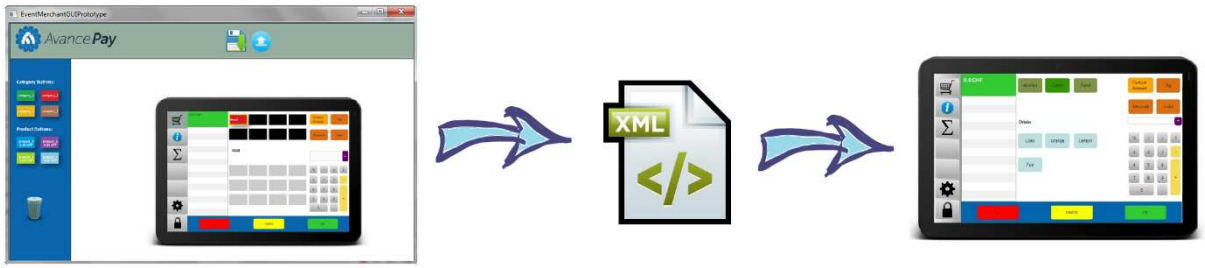

Fig. 47 Standalone application logic

The previous image explains that the configuration files are generated by the standalone application. These files must be afterwards stored in the appropriate folder in the tablet containing Avance Pay's POS application.

The load of a pre-configured file is accomplished by choosing the file from the system files. Then the file must be stored in the tablet. This method is optimal if the configuration is done by an employee of Avance Pay with the knowledge of where to store the file afterwards. A better approach to make it usable to users might be to connect an android device to the computer and detect such configuration file automatically. Once the user has saved the new configuration it remains in the same folder inside the tablet, thus avoiding the need of placing the file inside the appropriate place.

6.1.3. Android application solution

The Android solution has a slightly difference with respect to the standalone one. It is the use of i-Jetty, the android version of Jetty. Here you can see this solution graphically:

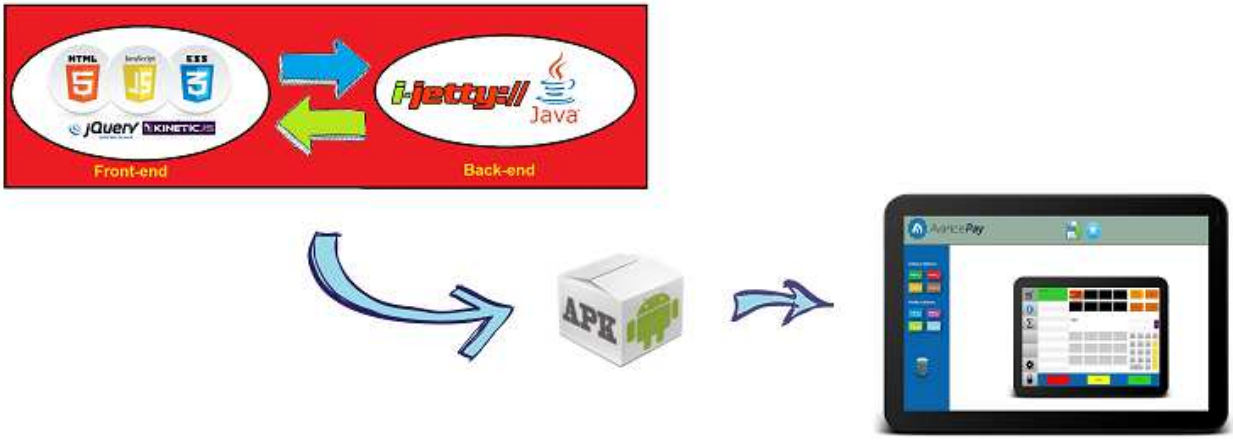

Fig. 48 Solution for Android application

An .apk file is generated, containing a WebView to display the front-end content. As in the previous solution the management of the servlets is done with i-Jetty.

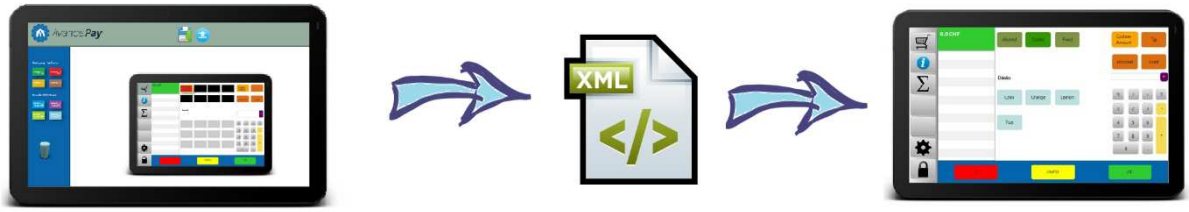

Fig. 49 Android Drag & Drop Editor logic

Currently, a configuration file must be chosen in order to edit it. For the future might be considered as an option to directly look if the tablet running the editor contains Avance Pay's POS application and in that case load automatically the current configuration if desired. Also, it would directly loaded when the users save it.

6.2. Design

Finally, after having decided how the mPOS app would look like and how it would work, the Drag & Drop Editor design could be accomplished.

Firstly the editable elements of the mPOS application had to be defined in order to establish the requirements of the editor. At an early stage the idea was to permit the editing

of various elements, but later was decided that would be a much better approach to offer only the possibility of editing few things.

These few editable elements would be the category or group buttons and the product buttons belonging the different categories. Moreover, the user might be able to choose among different style buttons.

Furthermore, the application had to offer the possibility of eliminating any inserted button, so a bin would be useful for that purpose. Also a Save Button and a Load Button were defined as compulsory elements of the Drag & Drop Editor. Their function is described in the next section of this chapter.

The editing tool had to follow the drag & drop principle to facilitate the use of the application to the users. The idea was that they could see their Sales Screen in the middle of the editor and they could be able to drag and drop the categories and products buttons from a tool pane at the left of the screen.

The following image shows the final design of the Drag & Drop Editor. It counts with all the elements.

- **Category buttons:** Four different style buttons that the user can drag and drop to a category button outline. The category buttons outlines appear in black within the tablet image.
- **Product buttons:** If a category button is selected, the product buttons appear. To select a category, the category button must be placed in the tablet and then it has to be clicked. There are also four styles of product buttons.
- **Bin:** Placed right below the product buttons. When a button placed in the tablet is dragged and dropped onto the bin the button is removed. If it is a category button containing product buttons, also these are eliminated.
- **Save button:** When clicked the current configuration is clicked, generating a configuration file with the xml structure.
- **Load button:** Allows to select a configuration file to load its configuration to be edited.
- **Tablet with button outlines:** The tablet with the buttons outlines where the buttons can be placed.

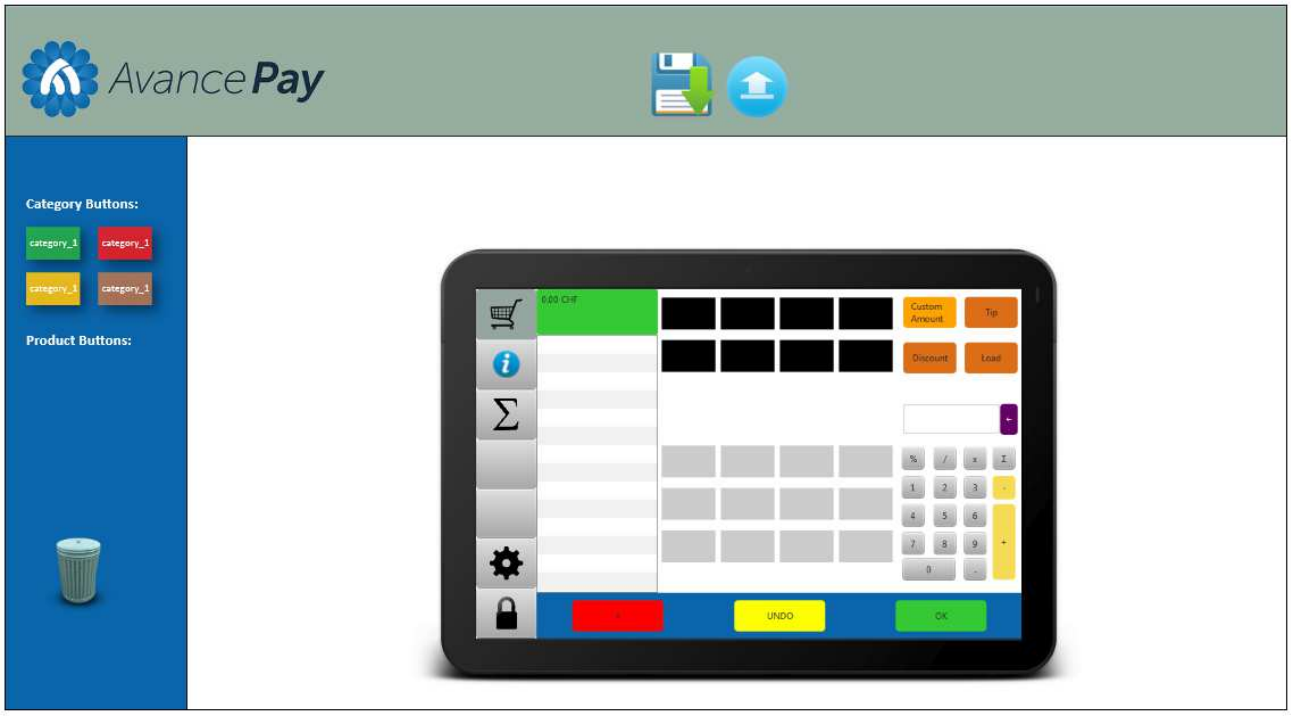

Fig. 50 Drag & Drop Editor initial state

When a category button is placed in a category outline and clicked, the product buttons appear.

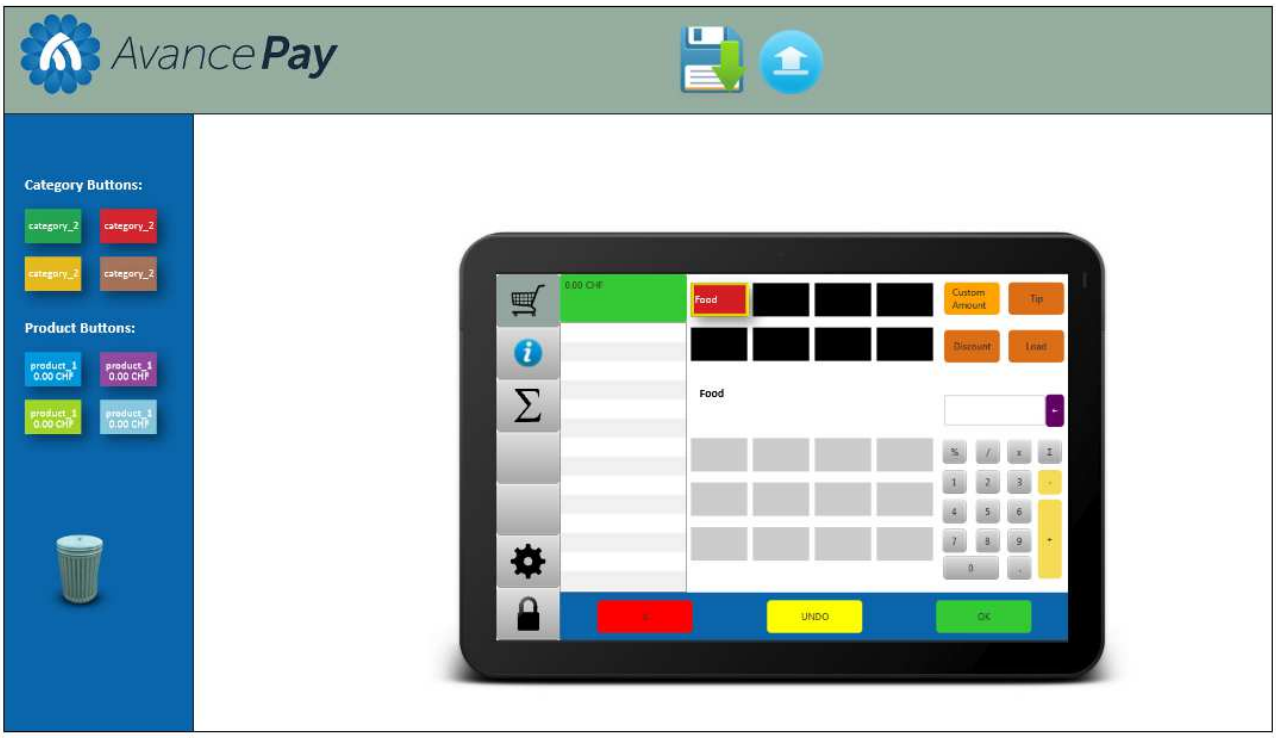

Fig. 51 Drag & Drop Editor category selected

An automatic dialog shows up when a new category button is placed in a category outline, where the user can introduce the category names.

Fig. 52 Drag & Drop Editor Category dialog

While the same happens for the product button but in this case the dialog has more fields to fill out.

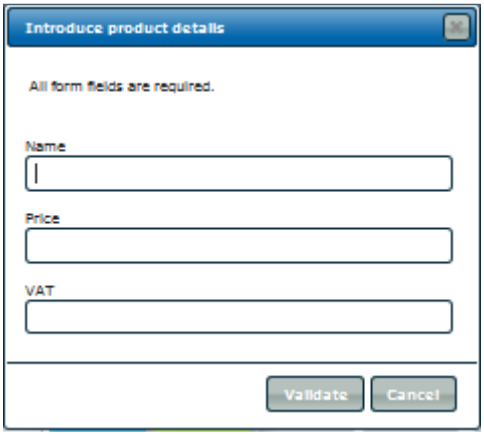

Fig. 53 Drag & Drop Editor - Product dialog

Once the editing is finished the editor shall produce an XML file for the categories and products, specifying the products belonging to each category, the name, the price and the VAT value. The following code shows an example of such configuration file.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<configuration>
	<categories>
```

```
        <category id="category_1">
            <name>Food</name>
            <btn-type>1</btn-type>
            <position>1</position>
            <products>
                <product id="product_5">
                    <name>Pizza</name>
                    <btn-type>1</btn-type>
                    <position>5</position>
                    <price>8.5</price>
                    <vat>21.0</vat>
                </product>
                <product id="product_6">
                    <name>Spaghetti</name>
                    <btn-type>1</btn-type>
                    <position>6</position>
                    <price>10.0</price>
                    <vat>21.0</vat>
                </product>
            </products>
        </category>
        <category id="category_6">
            <name>Drinks</name>
            <btn-type>2</btn-type>
            <position>6</position>
            <products>
                <product id="product_2">
                    <name>Coke</name>
                    <btn-type>1</btn-type>
                    <position>2</position>
                    <price>2.5</price>
                    <vat>21.0</vat>
                </product>
            </products>
        </category>
    </categories>
</configuration>
```

Accordingly, the load button had to allow the user to choose a configuration file in order to modify it. Its content is loaded and it is overwritten with the new configuration once the user saves his modifications.

When a button placed in a button outline within the tablet is dragged the bin had to show the user that it can be eliminated, changing its image to an open bin.

Fig. 54 Opened bin (left) and closed bin (right)

6.3. Development

The development explanation is split in two sub-sections: front-end and back-end programming.

6.3.1. Front-end programming

On the one hand, KineticJS' API was the main choice to animate the editor web application (front-end programming). Most of the interactive actions are supported by KineticJS, like the drag and drop actions, the change of images, the highlights, etc.

On the other hand, JQuery was also used, for example to show the dialogs to enter the category and product details.

In this section the programming logic will be briefly explained.

Basically, the index.htm document imports all the javascript files, which are the ones in charge of loading all the images and managing the logic of the application.

KineticJS allows you to add objects to canvas which will respond to events like mouse click and drag. The <canvas> element is only a container for graphics. You must use a script to actually draw the graphics. For that purpose we use KineticJS.

First of all we have to create a container in our HTML5 index page:

```
<div id="container"></div>
```

Then, we can import all the scripts:

```
<script>
        $.getScript('js/variables.js', function() {
        });
        $.getScript('js/main.js', function() {
        });
        $.getScript('js/productsManager.js', function() {
        });
        $.getScript('js/categoriesManager.js', function() {
        });
        $.getScript('js/productDialog.js', function() {
        });
        $.getScript('js/categoryDialog.js', function() {
        });
        $.getScript('js/loadButton.js', function() {
        });
        $.getScript('js/saveButton.js', function() {
```

```
        });
</script>
```

As shown in the code these scripts are imported using jQuery's getScript() function. After this, the main.js script loads the images and initiates the KineticJS' stage. The stage in KineticJS is used to contain multiple layers and is instantiated this way:

```
var stage = new Kinetic.Stage({
        container : 'container',
        width : window.innerWidth,
        height : window.innerHeight
});
```

The container refers to the div container created in the html index page. It is the canvas container and will be used to display all the elements created with KineticJS.

Then we have to add the layers. Two different layers are used:

- **Background layer:** contains the background image.
- **Elements layer:** contains the elements that integrate the editor (buttons, bin, logo, etc.).

Within the main method the buttons outlines are also created. This outlines are images placed inside the tablet image, where the buttons can be dropped. Finally the category buttons are created. To manage the creation of category buttons there is a javascript file called categoriesManager.js containing all the needed functions.

Inside the categoriesManager.js file can be found the logic to decide if more category buttons should be created as well as when to call the method createProdBtns() from the productsManager.js file. These decisions are made based on events related to the buttons.

To manage an event on a button is managed in the following way:

```
button.on('dragend', function() {…});
```

The first parameter of the method is a string indicating the action. The allowed action strings are the following: mouseover, mousemove, mouseout, mouseenter, mouseleave, mousedown, mouseup, click, dblclick, touchstart, touchmove, touchend, tap, dbltap, dragstart, dragmove, and dragend. So many are the possibilities.

To make sure that a button was dropped on top of the corresponding outline or the bin there is a method that verifies this. This method also admits a certain distance error, thus is, it verifies whether a button was dropped nearby its corresponding outlines or the bin.

While the users are configuring their app buttons, the mapping configuration is stored in a javascript object, so that when a category button is clicked its pre-configured product buttons are shown and the rest hidden.

Fig. 55 Drag & Drop Editor - Changing category

In the images it can be appreciated that the products of each category are loaded when the category button is clicked.

The dialogs to insert the product and category buttons details are implemented within two respective javascript files, containing functions that open the dialogs and save the inserted details in the javascript configuration map object.

6.3.2. Back-end programming

For the back-end Java was the choice. The two classes implemented were two servlets, one called when the save button is clicked and the other when the load button is clicked.

When the save button is clicked, the saveServlet.java is called using Ajax. In concrete the method used is a jQuery method. It sends the map variable containing the configuration map, but firstly this map is converted to a JSON string.

In the save button servlet the JSON string is received and parsed to java objects, where the information is stored. Then a class to create an XML file is called and the generated file saved. An OK message is returned to the front-end, where a successful message is shown to the user.

In the other case, when the load button is clicked the loadServlet.java is called asking the user to select an xml file containing a previous configuration. If the file is correct the xml file is converted to a JSON string and passed to the front-end, where it is read and stored in the configuration map variable. Once it is saved in the javascript object the new configuration is displayed.

7. Conclusions and future work

7.1. Conclusions

Throughout the present Thesis some Android Editors were analyzed in order to determine the best approach for Avance Pay's solution. After this analysis is concluded that a simple and intuitive editor would be the best choice. Moreover, these editors did not fulfill the requirements, i.e., to work for web browsers, as a standalone application and as an Android application. Hence a proprietary solution is proposed to address these requirements. However, some good ideas from these editors were appreciated and should be taken into account for the future work, such as storing the configurations in our server and updating the applications GUIs remotely.

The intention of doing it simple relies on the profile of the users, who are vendors, not necessarily with informatics skills. For that reason, it may be a good idea to introduce some tips to guide the users through their editing tasks.

Then, different mPOS were as well analyzed in order to determine the best design for our POS application. Many features from these mPOS were gathered and was found that the focus needs to be on one customer segment, due to the diversity of these and the differences among the applications they need.

To finalize, the final application was developed using web standardized APIs. Even though this application works fine for Android, the feeling and velocity are worse, so it is recommended to use native code in order to avoid this limitations.

7.2. Future work

The current application constitutes a basis application. In it the vendors can edit their POS applications from scratch or even load a preconfigured file to edit it. Future work may cover the access to an online platform, using a user and a password, where users may have access to different services. From their accounts they may see their current POS configuration installed on their devices. The idea may be that once they save a configuration it is updated remotely via an update manager.

Moreover, some other services may be interesting to offer from an online platform, such as reports, customer service, online shop to enhance the POS capabilities, etc.

Nonetheless, the application could also support offline editing by detecting a device with Avance Pay's POS application installed on it attached to the computer via USB; or if it is installed in the same device when running the editor in android. It may access the configuration file and load the new configuration when saved.

The current design suits for Event Merchant points of sale, so other customer segment applications can be designed and adapt the drag and drop editor so it allows the editing of the different applications.

Bibliography

[1] H. Stockman, “Communication by Means of Reflected Power.” 1948.

[2] ECMA, *Ecma-340: Near Field Communication Interface and Protocol-1*, no. December. 2004.

[3] ISO, *International Standard ISO / IEC: Near Field Communication*, vol. 18092. ISO / IEC, 2004.

[4] N. Forum, “About NFC Forum.” [Online]. Available: http://www.nfc-forum.org/aboutus/.

[5] INTECO, “La tecnología NFC: Aplicaciones y gestión de seguridad.” pp. 1–21, 1948.

[6] International Telecommunication Union (ITU), *Radio Regulations (Volume 1) - Article 5: Frequency allocations, RR 5.150*. .

[7] ETSI, “Near Field Communication Interface and Protocol-2 (NFCIP-2),” vol. 1. pp. 1–9, 2004.

[8] Innopay, *Mobile payments 2012*. 2012, pp. 1–100.

[9] “PayPal to enable in-store payments simply by scanning a QR code.”

[10] J. J. Echevarria, J. Ruiz-de-Garibay, J. Legarda, M. Alvarez, A. Ayerbe, and J. I. Vazquez, “WebTag: Web browsing into sensor tags over NFC.,” *Sensors (Basel).*, vol. 12, no. 7, pp. 8675–90, Jan. 2012.

[11] “Ultrasonic Payments: Naratte.”

[12] “Square Up.” [Online]. Available: https://squareup.com/.

[13] “i-Zettle.” [Online]. Available: https://www.izettle.com/.

[14] “SumUp.” [Online]. Available: https://sumup.co.uk/.

[15] NOKIA, “Inside NFC: secure payment technology.”

[16] M. Group, “App Inventor.”

[17] W3C, “HTTP Protocol (RFC 2616).”

[18] David Raggett, “A Review of the HTML+ Document Format.”

[19] W3C, “Cascading Style Sheets.”

[20] Findthebest.com, “Point of Sale (POS) Software.”

[21] “JavaFX WebView.” [Online]. Available: http://docs.oracle.com/javafx/2/api/javafx/scene/web/WebView.html.

[22] “Jetty Project.” [Online]. Available: http://www.eclipse.org/jetty/.

Annex A
Introducción

Motivación

Los pagos NFC ya están aquí. Muchas son las ventajas de tener todo integrado en el teléfono móvil. Es por este motivo que Avance Pay está desarrollando terminales de pago integrados en teléfonos móviles. El sistema de Avance Pay pretende ser rápido, seguro, económico, fiable y extensible. Presenta muchas ventajas tanto para los clientes como para los vendedores.

El desarrollo de una interfaz drag-and-drop tiene como objetivo facilitar a aquellas personas sin conocimientos de programación, la personalización de sus aplicaciones de venta. Esta herramienta conseguirá que el usuario, de una forma altamente intuitiva, pueda diseñar y hacer cambios en su propia aplicación.

Del mismo modo se pretende dotar al sistema de pago de Avance Pay de un elemento diferenciador respecto a otros sistemas de pago, y para ello se pretende proveer a los clientes de la empresa de una herramienta que les permita personalizar las aplicaciones de venta de sus negocios.

Objetivos

El objetivo de este Proyecto Fin de Carrera es diseñar y crear una interfaz gráfica de usuario (GUI) tanto para terminales móviles, con sistema operativo Android, como para clientes web. La idea es que los comerciantes que estén usando el sistema de Avance Pay, puedan editar sus propias aplicaciones de venta: cambiar la vista de la interfaz, cambiar precios, añadir nuevos productos, nuevas opciones, etc.

El sistema en mente consistirá en un editor drag-and-drop para entornos Android y otro para navegadores web y será diseñado y creado a lo largo del presente Proyecto Fin de Carrera.

Para desarrollar la aplicación en mente se analizarán distintos editores Android así como distintos Terminales Puntos de Venta móviles (mPOS por sus iniciales en inglés). La idea es entender como están implementados estos programas.

Metodología y plan de trabajo

Con el fin de lograr los objetivos de este Proyecto Fin de Carrera la siguiente metodología será seguida de manera estricta. En primer lugar se revisará literatura moderna sobre la tecnología NFC, de tal manera que se pueda presentar una visión general sobre la misma en este trabajo. Además, también serán revisados artículos sobre editores Drag and Drop y sobre código XML, entre otros.

Una vez se haya revisado la literatura moderna referente al tema, se haya obtenido una idea sobre el estado del arte y el estudiante se haya familiarizado con los editores existentes, serán definidos los requerimientos del sistema de Avance Pay, así como un plan de desarrollo del mismo.

Después del análisis funcional se procederá con la definición de los requerimientos del fichero intermedio (XML) y con la programación de las interfaces drag-and-drop. Una vez se tenga el código base, se procederá a la evaluación del mismo. Se corregirán los eventuales errores que pueda contener y se añadirán las posibles mejoras que se consideren oportunas.

El plan de trabajo constará de los siguientes pasos:

1. Revisión de la literatura moderna sobre la tecnología NFC y los editores Drag and Drop.
2. Familiarización con los editores existentes.
3. Definición de los requerimientos del sistema.
4. Definición de un plan de desarrollo.
5. Implementación de la primera “release” (prueba de concepto).
6. Evaluación y discusión de los resultados.
7. Finalización de la memoria.

Estructura de la memoria

Capítulo 1. Introducción: Introducción de la memoria, donde la motivación y objetivos del PFC son expuestos.

Capítulo 2. Introducción a la tecnología NFC: La tecnología NFC es presentada. Los modos, características y el estándar son explicados, así como una introducción a los pagos móviles, centrada en especial en los pagos NFC.

Capítulo 3. Introducción de editores drag & drop y WYSIWYG: El significado de estos términos y una comparación entre algunos editores se presentan en este capítulo.

Capítulo 4. Programación Web: Conceptos básicos de la programación web son presentados. Los principios cliente-servidor son explicados. Los principales estándares web son explicados.

Capítulo 5. Diseño de software para TPV: En primer lugar, las características y elementos de la interfaz gráfica de las diferentes aplicaciones TPV son analizadas y comparadas. En segundo lugar, el diseño de una aplicación TPV es explicado y finalmente se expone el proceso del desarrollo de una aplicación prototipo.

Capítulo 6. Diseño y desarrollo del editor drag-and-drop: En primer lugar, los requerimientos del Editor Drag-and-drop son expuestos así como la solución propuesta. En Segundo lugar, se muestra el proceso de diseño y desarrollo de la aplicación.

Capítulo 7. Conclusiones y trabajo futuro: Para terminar las conclusiones del proyecto y posibles líneas futuras son expuestas.

Annex B

Conclusiones y trabajo futuro

Conclusiones

A lo largo del presente Proyecto Fin de Carrera, algunos editores Android fueron analizados para poder determinar la posible mejor solución para el editor de Avance Pay. Después de este análisis se vio que un editor intuitivo y simple sería la mejor elección. También se vio que ninguno de estos editores cumplían los requerimientos, es decir, funcionar para Android, navegadores web y como aplicación independiente para Windows. Por lo tanto, una solución propia es propuesta para cumplir dichos requerimientos. De todos modos, algunas ideas de estos editores pueden tenerse en consideración para líneas futuras de este producto, tales como almacenar las configuraciones en nuestro servidor y ser capaces de actualizar la configuración de las distintas aplicaciones remotamente.

La intención de hacer una aplicación simple tiene que ver con el perfil del usuario, que son comerciantes sin conocimientos informáticos amplios, por lo general. Por esta razón, podría ser una buena idea introducir algunos consejos y ayudas para guiar al usuario en el proceso de edición de su aplicación.

Más adelante, diferentes aplicaciones TPV fueron analizadas y comparadas de manera que se pudiese determinar el mejor diseño para nuestra aplicación TPV. Muchas características de estos TPV fueron reunidas y se descubrió que debíamos centrarnos en un segmento de clientes, debido a la diversidad de estos segmentos y las distintas necesidades que estos presentan.

Para finalizar, la aplicación final fue desarrollada usando APIs de estándares web. A pesar de que esta aplicación funciona adecuadamente para Android, la percepción de velocidad y rendimiento no son como en una aplicación nativa, por lo que si esto se quiere evitar se recomienda desarrollar la aplicación en código nativo Android.

Trabajo futuro

La aplicación desarrollada a lo largo de este Proyecto Fin de Carrera constituye una aplicación básica. En ella los comerciantes pueden editar sus aplicaciones TPV desde cero o incluso cargar una configuración previa para modificarla. Líneas futuras podrían cubrir el acceso a una plataforma online, usando un usuario y contraseña, donde los usuarios podrían tener acceso a distintos servicios. Desde sus cuentas podrían ver la configuración actual cargada en sus terminales. La idea podría consistir en que una vez que una vez que el usuario guarde una nueva configuración online, la aplicación se actualice automáticamente con ese mismo diseño a través de un Gestor de Actualizaciones.

Además, otros servicios podrían ser de interés tales como informes de ventas, servicio de atención al cliente, tienda online con utilidades para aumentar las capacidades de las aplicaciones TPV, etc.

De cualquier modo, la aplicación también podría soportar edición offline detectando un terminal con la aplicación TPV de Avance Pay instalada en el mismo que haya sido conectado al ordenador mediante un cable USB; o si se trata del editor para Android que detectase la aplicación TPV en ese mismo terminal. Podría acceder los archivos de configuración y cargar la nueva configuración una vez que se guarde en el editor.

El diseño actual está pensado para comerciantes del mundo de los eventos, por lo que otros diseños para otros tipos de clientes podrían ser diseñados para aumentar la oferta de Avance Pay.

Annex C

1) Ejecución Material
   - Compra de ordenador personal (Software incluido) ..... 2.000 €
   - Material de oficina ..... 150 €
   - Total de ejecución material ..... 2.400 €

2) Gastos generales
   - 16 % sobre Ejecución Material ..... 384 €

3) Beneficio Industrial
   - 6 % sobre Ejecución Material ..... 144 €

4) Honorarios Proyecto
   - 1800 horas a 15 €/ hora ..... 27000 €

5) Material fungible
   - Gastos de impresión ..... 280 €
   - Encuadernación ..... 200 €

6) Subtotal del presupuesto
   - Subtotal Presupuesto ..... 32.558 €

7) I.V.A. aplicable
   - 21 % Subtotal Presupuesto ..... 6.837,18 €

8) Total presupuesto
   - Total Presupuesto ..... 39.395,18 €

Madrid, Abril de 2014

El Ingeniero Jefe de Proyecto

Fdo.: José María Angulo Pinedo

Ingeniero Superior de Telecomunicación

Annex D

Este documento contiene las condiciones legales que guiarán la realización, en este proyecto, de creación de un editor Drag-and-Drop que permita editar aplicaciones TPV para entornos Android. En lo que sigue, se supondrá que el proyecto ha sido encargado por una empresa cliente a una empresa consultora con la finalidad de realizar dicho sistema. Dicha empresa ha debido desarrollar una línea de investigación con objeto de elaborar el proyecto. Esta línea de investigación, junto con el posterior desarrollo de los programas está amparada por las condiciones particulares del siguiente pliego.

Supuesto que la utilización industrial de los métodos recogidos en el presente proyecto ha sido decidida por parte de la empresa cliente o de otras, la obra a realizar se regulará por las siguientes:

Condiciones generales

1. La modalidad de contratación será el concurso. La adjudicación se hará, por tanto, a la proposición más favorable sin atender exclusivamente al valor económico, dependiendo de las mayores garantías ofrecidas. La empresa que somete el proyecto a concurso se reserva el derecho a declararlo desierto.

2. El montaje y mecanización completa de los equipos que intervengan será realizado totalmente por la empresa licitadora.

3. En la oferta, se hará constar el precio total por el que se compromete a realizar la obra y el tanto por ciento de baja que supone este precio en relación con un importe límite si este se hubiera fijado.

4. La obra se realizará bajo la dirección técnica de un Ingeniero Superior de Telecomunicación, auxiliado por el número de Ingenieros Técnicos y Programadores que se estime preciso para el desarrollo de la misma.

5. Aparte del Ingeniero Director, el contratista tendrá derecho a contratar al resto del personal, pudiendo ceder esta prerrogativa a favor del Ingeniero Director, quien no estará obligado a aceptarla.

6. El contratista tiene derecho a sacar copias a su costa de los planos, pliego de condiciones y presupuestos. El Ingeniero autor del proyecto autorizará con su firma las copias solicitadas por el contratista después de confrontarlas.

7. Se abonará al contratista la obra que realmente ejecute con sujeción al proyecto que sirvió de base para la contratación, a las modificaciones autorizadas por la superioridad o a las órdenes que con arreglo a sus facultades le hayan comunicado por escrito al Ingeniero Director de obras siempre que dicha obra se haya ajustado a los preceptos de los pliegos de condiciones, con

arreglo a los cuales, se harán las modificaciones y la valoración de las diversas unidades sin que el importe total pueda exceder de los presupuestos aprobados. Por consiguiente, el número de unidades que se consignan en el proyecto o en el presupuesto, no podrá servirle de fundamento para entablar reclamaciones de ninguna clase, salvo en los casos de rescisión.

8. Tanto en las certificaciones de obras como en la liquidación final, se abonarán los trabajos realizados por el contratista a los precios de ejecución material que figuran en el presupuesto para cada unidad de la obra.

9. Si excepcionalmente se hubiera ejecutado algún trabajo que no se ajustase a las condiciones de la contrata pero que sin embargo es admisible a juicio del Ingeniero Director de obras, se dará conocimiento a la Dirección, proponiendo a la vez la rebaja de precios que el Ingeniero estime justa y si la Dirección resolviera aceptar la obra, quedará el contratista obligado a conformarse con la rebaja acordada.

10. Cuando se juzgue necesario emplear materiales o ejecutar obras que no figuren en el presupuesto de la contrata, se evaluará su importe a los precios asignados a otras obras o materiales análogos si los hubiere y cuando no, se discutirán entre el Ingeniero Director y el contratista, sometiéndolos a la aprobación de la Dirección. Los nuevos precios convenidos por uno u otro procedimiento, se sujetarán siempre al establecido en el punto anterior.

11. Cuando el contratista, con autorización del Ingeniero Director de obras, emplee materiales de calidad más elevada o de mayores dimensiones de lo estipulado en el proyecto, o sustituya una clase de fabricación por otra que tenga asignado mayor precio o ejecute con mayores dimensiones cualquier otra parte de las obras, o en general, introduzca en ellas cualquier modificación que sea beneficiosa a juicio del Ingeniero Director de obras, no tendrá derecho sin embargo, sino a lo que le correspondería si hubiera realizado la obra con estricta sujeción a lo proyectado y contratado.

12. Las cantidades calculadas para obras accesorias, aunque figuren por partida alzada en el presupuesto final (general), no serán abonadas sino a los precios de la contrata, según las condiciones de la misma y los proyectos particulares que para ellas se formen, o en su defecto, por lo que resulte de su medición final.

13. El contratista queda obligado a abonar al Ingeniero autor del proyecto y director de obras así como a los Ingenieros Técnicos, el importe de sus respectivos honorarios facultativos por formación del proyecto, dirección técnica y administración en su caso, con arreglo a las tarifas y honorarios vigentes.

14. Concluida la ejecución de la obra, será reconocida por el Ingeniero Director que a tal efecto designe la empresa.

15. La garantía definitiva será del 4% del presupuesto y la provisional del 2%.

16. La forma de pago será por certificaciones mensuales de la obra ejecutada, de acuerdo con los precios del presupuesto, deducida la baja si la hubiera.

17. La fecha de comienzo de las obras será a partir de los 15 días naturales del replanteo oficial de las mismas y la definitiva, al año de haber ejecutado la provisional, procediéndose si no existe reclamación alguna, a la reclamación de la fianza.

18. Si el contratista al efectuar el replanteo, observase algún error en el proyecto, deberá comunicarlo en el plazo de quince días al Ingeniero Director de obras, pues transcurrido ese plazo será responsable de la exactitud del proyecto.

19. El contratista está obligado a designar una persona responsable que se entenderá con el Ingeniero Director de obras, o con el delegado que éste designe, para todo relacionado con ella. Al ser el Ingeniero Director de obras el que interpreta el proyecto, el contratista deberá consultarle cualquier duda que surja en su realización.

20. Durante la realización de la obra, se girarán visitas de inspección por personal facultativo de la empresa cliente, para hacer las comprobaciones que se crean oportunas. Es obligación del contratista, la conservación de la obra ya ejecutada hasta la recepción de la misma, por lo que el deterioro parcial o total de ella, aunque sea por agentes atmosféricos u otras causas, deberá ser reparado o reconstruido por su cuenta.

21. El contratista, deberá realizar la obra en el plazo mencionado a partir de la fecha del contrato, incurriendo en multa, por retraso de la ejecución siempre que éste no sea debido a causas de fuerza mayor. A la terminación de la obra, se hará una recepción provisional previo reconocimiento y examen por la dirección técnica, el depositario de efectos, el interventor y el jefe de servicio o un representante, estampando su conformidad el contratista.

22. Hecha la recepción provisional, se certificará al contratista el resto de la obra, reservándose la administración el importe de los gastos de conservación de la misma hasta su recepción definitiva y la fianza durante el tiempo señalado como plazo de garantía. La recepción definitiva se hará en las mismas condiciones que la provisional, extendiéndose el acta correspondiente. El Director Técnico propondrá a la Junta Económica la devolución de la fianza al contratista de acuerdo con las condiciones económicas legales establecidas.

23. Las tarifas para la determinación de honorarios, reguladas por orden de la Presidencia del Gobierno el 19 de Octubre de 1961, se aplicarán sobre el denominado en la actualidad “Presupuesto de Ejecución de Contrata” y anteriormente llamado ”Presupuesto de Ejecución Material” que hoy designa otro concepto.

Condiciones particulares

La empresa consultora, que ha desarrollado el presente proyecto, lo entregará a la empresa cliente bajo las condiciones generales ya formuladas, debiendo añadirse las siguientes condiciones particulares:

1. La propiedad intelectual de los procesos descritos y analizados en el presente trabajo, pertenece por entero a la empresa consultora representada por el Ingeniero Director del Proyecto.

2. La empresa consultora se reserva el derecho a la utilización total o parcial de los resultados de la investigación realizada para desarrollar el siguiente proyecto, bien para su publicación o bien para su uso en trabajos o proyectos posteriores, para la misma empresa cliente o para otra.

3. Cualquier tipo de reproducción aparte de las reseñadas en las condiciones generales, bien sea para uso particular de la empresa cliente, o para cualquier otra aplicación, contará con autorización expresa y por escrito del Ingeniero Director del Proyecto, que actuará en representación de la empresa consultora.

4. En la autorización se ha de hacer constar la aplicación a que se destinan sus reproducciones así como su cantidad.

5. En todas las reproducciones se indicará su procedencia, explicitando el nombre del proyecto, nombre del Ingeniero Director y de la empresa consultora.

6. Si el proyecto pasa la etapa de desarrollo, cualquier modificación que se realice sobre él, deberá ser notificada al Ingeniero Director del Proyecto y a criterio de éste, la empresa consultora decidirá aceptar o no la modificación propuesta.

7. Si la modificación se acepta, la empresa consultora se hará responsable al mismo nivel que el proyecto inicial del que resulta el añadirla.

8. Si la modificación no es aceptada, por el contrario, la empresa consultora declinará toda responsabilidad que se derive de la aplicación o influencia de la misma.

9. Si la empresa cliente decide desarrollar industrialmente uno o varios productos en los que resulte parcial o totalmente aplicable el estudio de este proyecto, deberá comunicarlo a la empresa consultora.

10. La empresa consultora no se responsabiliza de los efectos laterales que se puedan producir en el momento en que se utilice la herramienta objeto del presente proyecto para la realización de otras aplicaciones.

11. La empresa consultora tendrá prioridad respecto a otras en la elaboración de los proyectos auxiliares que fuese necesario desarrollar para dicha aplicación industrial,

siempre que no haga explícita renuncia a este hecho. En este caso, deberá autorizar expresamente los proyectos presentados por otros.

12. El Ingeniero Director del presente proyecto, será el responsable de la dirección de la aplicación industrial siempre que la empresa consultora lo estime oportuno. En caso contrario, la persona designada deberá contar con la autorización del mismo, quien delegará en él las responsabilidades que ostente.