ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"DESARROLLO DE UN SISTEMA PARA EL AUTOCOMPLETADO DINÁMICO DE INFORMACIÓN DE USUARIOS EN PUNTOS DE VENTA"

INFORME DE MATERIA INTEGRADORA

Previo a la obtención del Título de:

INGENIERO EN CIENCIAS COMPUTACIONALES

KEVIN RICARDO SILVA CHÁVEZ

ESTEBAN DANIEL MUÑOZ GUEVARA

GUAYAQUIL – ECUADOR

AÑO: 2016

AGRADECIMIENTOS

Agradezco en primer lugar a Dios por haberme permitido llegar hasta este momento y por darme la dicha de contar con mis padres y demás personas muy cercanas a mí, las cuales que en todo momento me han brindado su apoyo incondicional y el ánimo cuando lo he necesitado. También a cada uno de mis profesores que he tenido a lo largo de mi carrera en la universidad, especialmente a la PhD. Lorena Carlo que me ha sabido guiar a lo largo del desarrollo de este proyecto y a su vez al PhD Daniel Ochoa por tener la confianza, paciencia y dedicación con nuestro proyecto. Por último deseo agradecer a la Escuela Superior Politécnica del Litoral por acogerme todos estos años y darme las herramientas intelectuales para desenvolverme en el ámbito profesional.

Kevin Ricardo Silva Chávez

AGRADECIMIENTOS

Agradezco a Dios por la voluntad y fuerza que me ha dado para mantenerme constante en la lucha hacia mi título universitario, a mis padres quienes siempre estuvieron ahí para darme la mano en cualquier problema que se me presentaba, a mis familiares más cercanos que de una u otra manera me dieron la mano para seguir adelante, a mis profesores con los cuales he aprendido mucho y siempre me han ayudado cuando lo he necesitado, gracias a la Dra. Lorena Carlo directora del proyecto y el Dr. Daniel Ochoa tutor del mismo quienes nos dirigieron en el transcurso del proyecto dándonos ideas y fuerzas para seguir adelante. Para finalizar deseo agradecer a la ESPOL por ser la institución que me brindó los conocimientos que me servirán a lo largo de mi carrera como profesional.

Esteban Daniel Muñoz Guevara

DEDICATORIA

Dedico este trabajo a Dios ya que sin El nada hubiera sido posible, también a mi familia porque con su esfuerzo y apoyo siempre me mantuvieron constante en esta etapa de mi vida estudiantil y en general a todas las personas que me han ayudado en cada momento en que lo he necesitado.

Kevin Ricardo Silva Chávez

DEDICATORIA

Le dedico este logro primero a Dios ya que El me dió las fuerzas necesarias para llegar hasta aquí y me mantuvo firme en el camino, a mis padres Esteban Calixto Muñoz Sellan y Lorena Narcisa Guevara Medina quienes han influido en mi para ser un hombre de bien, lleno de valores, a mis hermanos, a mi novia Andrea Riera, y a mis profesores quienes con su vocación y conocimiento han formado a un profesional.

Esteban Daniel Muñoz Guevara

TRIBUNAL DE EVALUACIÓN

Ph.D. Lorena Carlo Unda

PROFESOR EVALUADOR

Ph.D. Daniel Ochoa Donoso

PROFESOR EVALUADOR

DECLARACIÓN EXPRESA

"La responsabilidad y la autoría del contenido de este Trabajo de Titulación, nos corresponde exclusivamente; y damos nuestro consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual"

Kevin Ricardo Silva Chávez

Esteban Daniel Muñoz Guevara

RESUMEN

Las personas en el contexto de su día viven un estilo de vida acelerado por lo cual cualquier actividad que le tenga que tomar mucho tiempo incomoda y aburre en ciertas ocasiones, el proceso de facturación en Ecuador es una de estas actividades que todavía se realizan en varios puntos de venta preguntando uno a uno cada dato que precisa la factura, lo cual resulta demasiado lento e incómodo para el cliente. Por esto, generalmente lo que hacen los clientes es comunicarle al cajero encargado del punto de venta que le emita dicha factura a nombre de *Consumidor Final*. Al hacer esto se perjudica el cliente ya que en el momento de realizar la correspondiente declaración de impuestos no se podrá considerar estas facturas de *Consumidor Final*, solo las que están a su nombre.

QRInvoice es un prototipo de sistema que ofrece agilizar de manera significativa este proceso. Cuenta con una aplicación móvil que genera un código QR que contiene los datos del cliente y con una aplicación de escritorio que lee dicho código QR por medio de la webcam de la computadora, extrae los datos y luego los inserta en la factura, todo esto de manera automática. Tiene el potencial de evitar molestias al cliente y aumentar el volumen de ventas de una empresa a causa de la mayor rapidez con la que se atiende a cada cliente.

Se utilizó la metodología ágil llamada SCRUM por las exigencias mismas de presentar avances funcionales en un corto tiempo y porque el usuario debía estar inmerso contantemente en el proceso de desarrollo. En la fase de análisis se conceptualizó las necesidades que tenían los clientes que iban a realizar una compra y tenían que emitir la factura a su nombre y se estudió el alcance que debía tener el sistema. En la fase de diseño se determinaron las herramientas adecuadas para el desarrollo, además de establecer los conceptos de usabilidad, rapidez y eficiencia como características fundamentales del sistema. En la fase de implementación se desarrolló de manera iterativa cada módulo funcional. Como resultados se obtuvo que el proyecto posee un buen nivel de usabilidad y sencillez de uso para el cliente, también que por tener una base de datos interna el nivel de

rapidez de la aplicación es muy satisfactorio. Todo esto indica que el uso del sistema QRInvoice ayuda a mejorar esta actividad tan básica en el proceso contable de una empresa y a su vez mantiene un adecuado nivel de satisfacción de los clientes.

ÍNDICE GENERAL

CAPÍTULO 1

1. INTRODUCCIÓN

En la sociedad se vive atado a derechos que se deben hacer cumplir y obligaciones que como ciudadanos se tienen que acatar para el desarrollo del bien común de la sociedad y del país en general. La facturación es una de estas obligaciones que las empresas y las personas naturales deben realizar con el fin de sustentar sus actividades económicas.

Una persona natural si posee RUC debe facturar con el fin de sustentar los gastos que proceden de su actividad comercial detallada en su RUC. Las personas que no poseen RUC pero que en su trabajo perciben un ingreso mayor a 900 dólares deben realizar su declaración de gastos personales para la cual se detallan los gastos que se ha tenido que provienen de las facturas en las cuales se encuentra el nombre de la persona en cuestión en el período de un año.

Una empresa debe de pedir facturas para actividades que correspondan a *"Obtener, mantener y mejorar"* los modelos de negocios y procesos estratégicos correspondientes a la empresa, para la compra de suministros y demás gastos de gestión [1].

A causa de todo lo detallado en los párrafos anteriores existe una necesidad de que las facturas de gastos contengan nuestros datos por la exigencia de emitir declaraciones de impuesto que nos pide el SRI.

El mundo actual vive de manera acelerada, desde que empieza el día hasta que este acaba. El tiempo llega a ser considerado como uno de los bienes más preciados e insustituibles en la vida, incluso más que el dinero o cualquier bien [2], por lo cual ocurre muy a menudo que en el momento de realizar una compra la persona pida al cajero que la factura vaya a nombre de *Consumidor final* para evitar pasar tiempo entregando sus datos a la persona encargada de caja, lo cual a la empresa que emite la factura no le afecta pero el cliente si se ve afectado en cuanto a que las facturas a nombre de consumidor final no sirven para declarar sus gastos

y además no se podrá sustentar la actividad comercial [3], además que al emitir las facturas con datos específicos va a ayudar al usuario a aumentar su crédito tributario y así al momento de hacer su declaración de impuestos el valor que se deberá pagar será menor a causa del crédito tributario que procede de las facturas de compras relacionadas a la actividad comercial que ha emitido a su nombre [4].

La solución que se plantea corresponde en desarrollar un sistema dividido en dos plataformas, una móvil encargada de encapsular dentro de un código QR los datos personales del cliente y una aplicación de escritorio que se encarga de insertar los datos del cliente en la interfaz gráfica de la factura, todo esto independiente del lenguaje de programación en el que el punto de venta esté desarrollado y en qué sistema operativo se ejecute el software (Windows o Linux)

2.1 Trabajos relacionados

2.1.1 Informe de Investigación (Centro de Visión y Robótica)(CVR)

En el informe “Estableciendo un Hook” [5] presentado al CVR se implementa una plataforma en Windows para envío de mensajes entre el sistema operativo y los procesos. Este software utiliza 2 maneras para poder monitorear dichos mensajes.

- Windows Hooks.- Función callback que permite monitorear mensajes entre el sistema operativo y los procesos.
- SDK Windows - Función SendMessage().- Permite enviar un mensaje a una ventana, la cual lo recibe mediante un procedimiento por defecto de la ventana llamado *window procedure.*

En el reporte del CVR se describen ciertas consideraciones que se deben de tomar en cuenta al trabajar con los objetos gráficos del sistema operativo:

- La versión del sistema operativo (32 o 64bits) ya que el SDK de Windows cambia para cada versión.

- El tipo de lenguaje en el cual está hecha la aplicación del punto de venta porque de acuerdo a esto se verifica si se debe ingresar al kernel del sistema operativo (el caso de aplicaciones implementadas en c#) o si se debe ingresar a la máquina virtual de java (el caso de aplicaciones java).
- El sistema operativo en cuestión (Windows o Linux) ya que cada sistema operativo maneja de manera distinta la interfaz gráfica.

Este trabajo se enfoca en la primera parte a investigar a fondo acerca de las distintas maneras detalladas de obtener los identificadores de cada ventana de manera individual con el mouse para así proceder a la siguiente parte del proyecto que fue realizar las conexiones de la computadora con la cámara y luego establecer la información del código QR en los controles que corresponden a la factura del punto de venta.

CAPÍTULO 2

2. ANALISIS Y DESARROLLO DEL SISTEMA

2.1 Análisis de la solución

QRInvoice es un sistema que ayuda al usuario que está realizando una compra en algún local y desea naturalmente que su factura se encuentre a su nombre, que dicho proceso se agilite simplemente mostrando el código QR generado en la aplicación móvil correspondiente a sus datos, de esta manera la aplicación de escritorio se encargará de forma automática de leer el código, desencriptar los datos contenidos en él, e insertarlos en la vista de la factura del punto de venta.

El proceso que se sigue hoy en día es demasiado lento puede resultar incluso molestoso en algunas situaciones ya que el cajero del punto de venta debe empezar a preguntar uno por uno (en el peor de los casos) cada campo que debe llenar en la factura para luego recién proceder a registrar dicho documento. Este proceso se pretende agilizar de manera que lo único que el cliente debe hacer es mostrar el correspondiente código QR y al pasarlo por la cámara, los datos se llenarán de manera dinámica en su factura.

Se puede asegurar que las ventajas son para las dos partes involucradas en la venta ya que, por un lado, como ya se ha mencionado el cliente podrá tener su factura con datos de manera muy rápida, sencilla y cómoda para él. Por otro lado la empresa que vende el producto estará en capacidad de generar más rápido sus facturas y el cliente estará satisfecho con la rapidez con la que se le atenderá.

El sistema para poder trabajar requiere tener por parte del cliente un Smartphone y por parte del local de ventas una cámara web. En el caso del teléfono celular, la mayoría de personas tienen uno, y en cuanto a la cámara no se exige demasiado ya que la aplicación trabaja con una resolución estándar de cualquier cámara sencilla.

En la Tabla 1 se visualiza las ventajas y desventajas que ofrece el sistema QRInvoice.

Ventajas	Desventajas
✓ Agilita el proceso de venta y facturación. ✓ Permite generar facturas con datos de múltiples personas. ✓ Genera satisfacción en el cliente por la atención eficiente al momento de generar su factura.	× Se necesita que el cliente posea un teléfono celular inteligente. × Cada punto de venta debe tener una cámara web estándar. × Actualmente funciona solamente para sistemas operativos Windows y Linux. × El prototipo aún no ha sido integrado con un sistema de punto de venta real.

Tabla 1: Ventajas y desventajas del uso del sistema QRInvoice

2.1.1 Terminología

Para poder entender los textos posteriores donde se explica a detalle cada módulo se describe en la Tabla 2 los términos utilizados en QRInvoice y qué significan en el contexto del sistema.

Término	Significado
Usuario	Corresponde a la cuenta que el cliente crea al iniciar la aplicación móvil. En pocas palabras, el dueño del celular.
Persona	Describe los datos personales del individuo que servirán para llenar la factura mediante el código QR generado a partir de dichos datos.

Tabla 2: Terminología usada en el contexto del sistema QRInvoice

2.1.2 Aplicación de escritorio (Windows y Linux)

Se cuenta con una aplicación de escritorio apta para correr en la plataforma Windows y otra que se puede ejecutar en una plataforma Linux. El objetivo es que el sistema sea totalmente flexible y sea capaz de instalarse en cualquiera de estos sistemas operativos mencionados. Se debe mencionar que todavía no se ha hecho la integración ni las pruebas en un punto de venta real.

La aplicación de escritorio QRInvoice actualmente cuenta con 2 módulos:

- **Módulo de Configuración**

  Este módulo es el encargado de dar la respectiva información y guiar paso a paso a la persona en el proceso de configuración de la aplicación. Se debe seleccionar uno a uno cada campo hasta completar todos los campos disponibles.

  La información que se debe configurar será: Cédula, Nombre, Apellido, Dirección, Email y Teléfono.

- **Módulo de Iniciar captura**

  Este módulo es el encargado de capturar la imagen de la cámara web, detectar el código QR, extraer los datos contenidos en él y, de haber configurado correctamente la aplicación al inicio, inserta dichos datos en la vista de la factura.

2.1.3 Aplicación móvil (Android)

El aplicativo móvil QRInvoice se encuentra disponible actualmente para la plataforma Android y cuenta con los siguientes módulos:

- **Módulo Agregar persona**

  Este módulo se encarga de pedir que la información necesaria para generar el código QR. Valida la información y luego de

esto, se generará el respectivo código QR para la persona recién creada.

- **Módulo Ver persona**

  Modulo encargado de mostrar en primer lugar la información correspondiente a esa persona, y luego el código QR generado con esos datos.

- **Módulo Editar persona**

  Se encarga de mostrar los datos de dicha persona para así verificar cual deseo cambiar, y si es así, al momento de guardar la persona con los campos modificados se genera nuevamente un código QR correspondiente a dicha persona con los nuevos datos ingresados.

- **Módulo Eliminar persona**

  Este módulo se encarga de eliminar totalmente los datos de esa persona, incluyendo aquí el código QR correspondiente a la persona a eliminar.

- **Módulo Iniciar/cerrar sesión**

  Este módulo se encarga de autenticar al usuario, una vez iniciado sesión se cargará en la base de datos interna toda la información relacionada al usuario, esto es: Información del usuario, todas las personas pertenecientes a dicho usuario y los respectivos códigos QR asociados a estas personas.

- **Módulo Registrar/Crear usuario**

  Se encarga de generar un nuevo usuario a fin de que el individuo que acaba de instalar a aplicación en su teléfono pueda iniciar sesión para poder empezar a usar el sistema QRInvoice.

- **Módulo Editar usuario**

  Este módulo es encargado de mostrar la información correspondiente al usuario y si es necesario cambiar alguno de estos datos.

- **Módulo Lista Personas**

  Este módulo se encarga de mostrar en forma de lista cada persona asociada al usuario activo en dicha sesión, además de mostrar información básica de la persona como el nombre y la cédula. Desde este módulo se podrá acceder a todas las opciones disponibles de la aplicación.

2.1.4 Esquema general

A continuación en la Figura 2.1 se presenta un esquema paso a paso del funcionamiento del sistema QRInvoice tomando en cuenta que la aplicación ya está instalada en el celular del usuario y en el punto de venta y en la Figura 2.2 se muestra un esquema resumido del objetivo del sistema.

Figura 2.1: Esquema paso a paso del funcionamiento del sistema QRInvoice

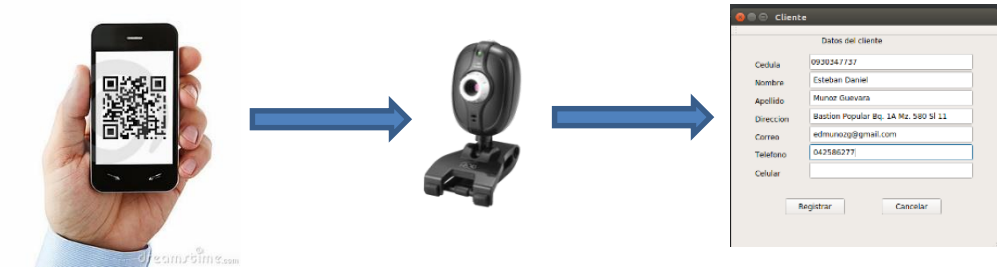


Figura 2.2: Esquema resumido del funcionamiento del sistema

2.2 Metodología de desarrollo ágil

La metodología ágil tiene como objetivo llevar el correcto manejo del software de una manera iterativa e incremental, las cuales son realizadas de la siguiente manera [6]:

- Planificación
- Análisis de requisitos
- Diseño
- Codificación
- Pruebas y Documentación

2.2.1 Fase de análisis

En esta fase se pudo aprender acerca de las necesidades de las personas con este tema de la facturación por medio de preguntas a diferentes profesionales y además por medio de las distintas reuniones que se tuvo con el PhD Daniel Ochoa (Director del Centro de Visión y Robótica de la ESPOL), el cual hizo notar la problemática principal que conlleva a desarrollar el sistema QRInvoice como propuesta dirigida a mejorar la atención al cliente en cuanto a tiempo de emisión de su factura, también en cuanto a la incomodidad de dar datos personales, y además en cuanto a la flexibilidad de poder tener a la mano datos de diferentes personas, en el caso que no se desee pedir alguna factura a nombre del cliente, sino que se desee pedir a nombre de algún tercero.

También se pudo conceptualizar el tema de la facturación y todo lo que conlleva a obligaciones tributarias con la ingeniera comercial Ginger Chávez, lo cual nos ayudó a definir los requerimientos del sistema.

2.2.2 Fase de diseño

Se consideró en la fase de diseño que las aplicaciones desarrolladas cumplan con las necesidades del cliente y que sean fáciles de utilizar por los mismos.

Se utilizó la herramienta de control de versiones *Git* [7], la cual permite trabajar en equipo teniendo el código fuente centralizado en un repositorio llamado *Bitbucket* [8]. También se utilizó la herramienta para el control de avances y gestión de proyecto llamada *Taiga.io*, la cual permite controlar el proyecto utilizando la metodología ágil SCRUM [9] donde se pueden generar historias de usuarios y estas asignárselas a un sprint con el fin de crear un paquete funcional para los clientes.

Las herramientas utilizadas para el desarrollo de las aplicaciones se muestran en la tabla 3. Para el desarrollo móvil se trabajó con Android Studio [10], para la versión de escritorio en Windows (Visual Studio [11]) y Linux (PyCharm [12] y NetBeans [13])

Git	Software para control de versiones de código fuente [7].
Bitbucket	Repositorio centralizado donde varios desarrolladores pueden tener acceso al código fuente para poder realizar cambios en el mismo [8].
Taiga.io	Plataforma web open source para el control y planificación de proyectos utilizando la metodología ágil SCRUM [9].
Visual Studio 2012	Entorno de desarrollo para aplicaciones que corren sobre Windows [11].
PyCharm	Entorno de desarrollo para aplicaciones utilizando el lenguaje de programación PYTHON [12] .

NetBeans	Entorno de desarrollo para aplicaciones utilizando el lenguaje de programación JAVA [13].
QtCreator	Entorno de desarrollo para aplicaciones utilizando el lenguaje de programación C++ [14].

Tabla 3: Herramientas utilizadas en el desarrollo

2.2.3 Fase de implementación

Para el desarrollo del sistema se utilizó la metodología de desarrollo ágil SCRUM ya que este método proporciona una retroalimentación constante con el usuario, permitiendo así que cada cambio que dicho usuario plantee se lo pueda realizar a tiempo. De esta manera se garantizaba que el sistema vaya siempre de acuerdo a las necesidades del usuario.

La metodología SCRUM también garantizó trabajar de manera colaborativa y eficiente. El problema se lo podía dividir en pequeñas tareas y al tener una comunicación constante en el equipo de desarrollo las responsabilidades se podían repartir y discutir.

La necesidad del sistema de ser presentado en un corto tiempo impulsó al equipo a utilizar esta metodología. También por la naturaleza cambiante de los requerimientos de los usuarios.

El proceso de implementación utilizando SCRUM establece que se debe dividir el producto en entregables más pequeños (prototipos) los cuales se desarrollaron en diferentes iteraciones. Luego de cada iteración se procedió a mostrar dicho avance al usuario para identificar cualquier error y a su vez para reafirmar que el desarrollo iba por el camino correcto.

QRInvoice se desarrolló en un conjunto de 4 sprints o iteraciones los cuales se presentaban cada 4 semanas (tiempo que duraba cada iteración).

También se realizó una reunión semanal con el profesor encargado cada viernes para verificar avances, identificar dificultades y recibir

retroalimentación acerca de lo desarrollado hasta ese punto. Se utilizó una herramienta web open source llamada Taiga para administrar el proyecto utilizando la metodología SCRUM.

La Figura 2.3 muestra las iteraciones que se establecieron en el proyecto QRInvoice.

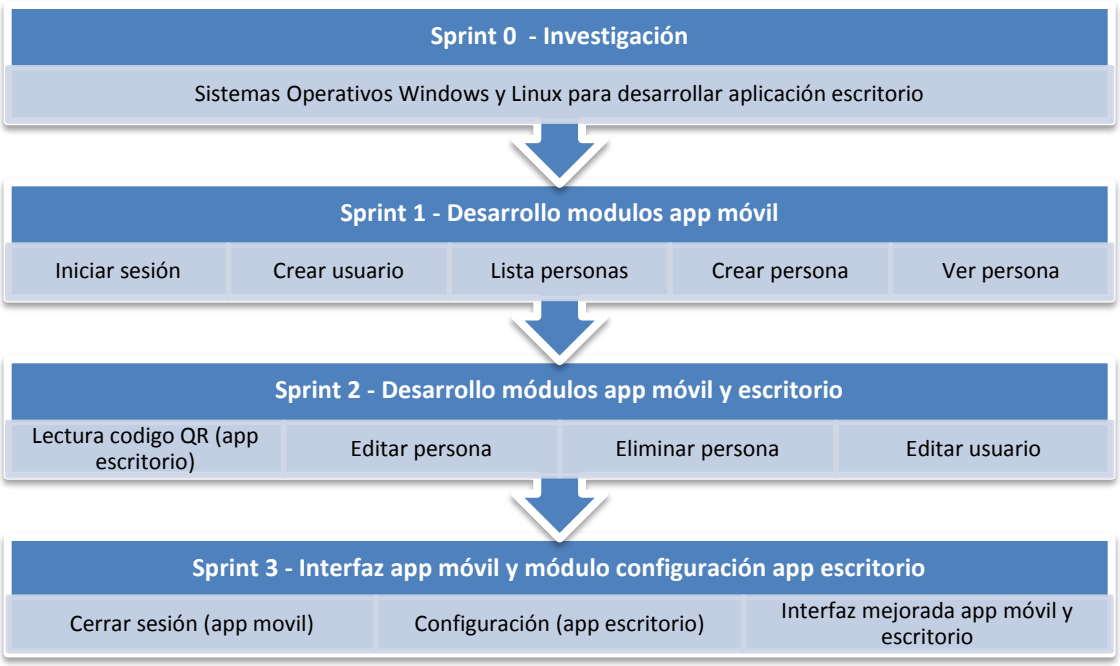


Figura 2.3: Detalle de iteraciones en el desarrollo del sistema

Sprint 0: Investigación acerca del funcionamiento del manejo de las ventanas de cada sistema Operativo (Windows y Linux)

En el primer sprint se realizó una investigación en cada sistema operativo de cómo realizan cada uno el manejo de las ventanas. Además también se estudió la manera en que cada sistema realiza el mapeo de los controles correspondientes a cada venta para así encontrar la forma de mantener algún dato que haga referencia a ese control. Después de tener almacenado este identificador -que debía ser único- se podía acceder libremente a dicho control.

Al realizar la investigación del sistema operativo Windows se pudo encontrar que el manejador de ventanas se encontraba a nivel de

Kernel y que la manera para poder acceder a los datos de algún control era capturando los mensajes que se transmiten desde y/o hasta las ventanas. La herramienta que permite realizar la respectiva captura de dichos mensajes es llamada *Windows Hooks* [15], el cual es una función callback que se encarga de monitorear los mensajes entre el sistema operativo y cada uno de los procesos que corren en memoria (entre ellos las ventanas). Una vez utilizando esta función callback en una interfaz sencilla (ver Figura 2.4) se pudo acceder a cada proceso y subproceso en ejecución y así se obtuvo los identificadores requeridos como lo muestra la Figura 2.5

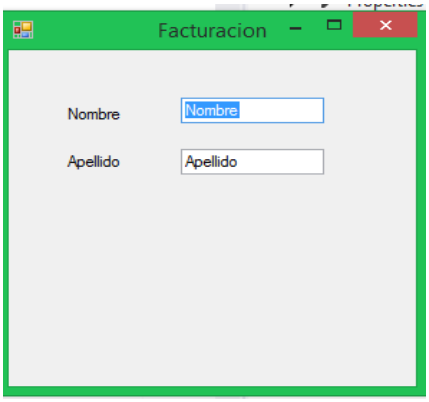


Figura 2.4: Interfaz demo inicial de prueba

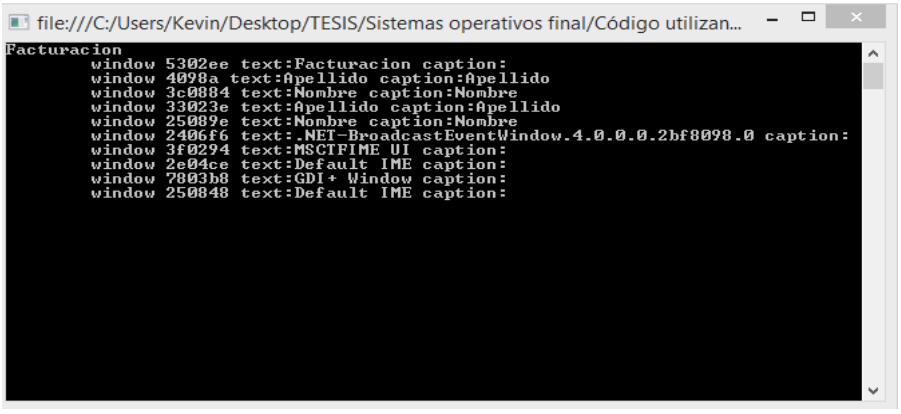


Figura 2.5: Consola donde se muestra los resultados de usar Windows Hooks en el demo de prueba

En cuanto a Linux la manera en la que se lleva a cabo la comunicación con los objetos gráficos del sistema operativo es a

través de un servicio llamado x-server, el cual consiste en una capa que permite interactuar al usuario con los objetos gráficos mostrados en la pantalla.

Existe una librería que provee la funcionalidad de comunicarse entre el servidor de pantallas (x-server) y la interfaz gráfica llamada xdotool [16], el cual provee una abstracción al sub sistema de capas de Linux.

Algunas de las funcionalidades principales que se utilizan en el sistema que ofrece esta librería son:

- ***mousemove***

  Permite mover el mouse a una determinada posición (X, Y) de la pantalla.

- ***xdotool search --name "NOMBRE_VENTANA"***

  Permite buscar una ventana por el nombre de la misma.

- ***windowactivate***

  Permite pasar una ventana al primer plano de la pantalla.

- ***xdotool type "TEXTO"***

  Permite la inserción de texto en el lugar donde se encuentre el focus.

Luego se procedió a crear el primer demo en un archivo "*bash scripting"* (Ver Figura 2.6) que se ejecuta a través de líneas de comando en Linux, el cual se encarga de enviar datos a la ventana de facturación. En la Figura 2.7 se muestra el comando que ejecuta el archivo bash llamado "send_data" en Linux.

```
#!/bin/sh

CEDULA=$1
NOMBRE=$2
APELLIDO=$3
DIRECCION=$4
CORREO=$5
TELEFONO=$6
CELULAR=$7

xdotool search --name "MainWindow" windowactivate
xdotool windowfocus [window=%1]

xdotool type $CEDULA
xdotool key Tab
sleep 0.2

xdotool type $NOMBRE
xdotool key Tab
sleep 0.2

xdotool type $APELLIDO
xdotool key Tab
sleep 0.2


xdotool type $DIRECCION
xdotool key Tab
sleep 0.2


xdotool type $CORREO
xdotool key Tab
sleep 0.2


xdotool type $TELEFONO
xdotool key Tab
sleep 0.2


xdotool type $CELULAR
xdotool key Tab
sleep 0.2
```

Figura 2.6: Script “send_data” utilizado en el primer demo de la aplicación de la plataforma Linux

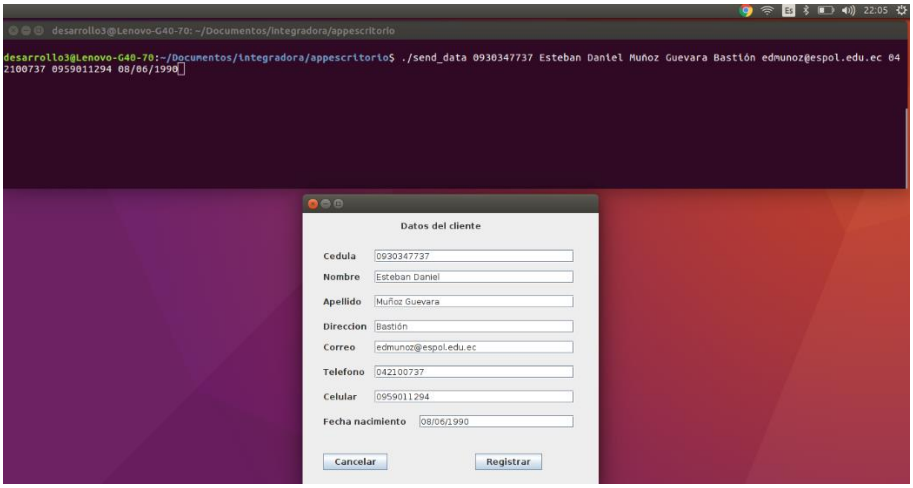


Figura 2.7: Ejecución del script "send_data"

Utilizando xdotool se levanta la ventana de nombre Cliente, luego se coloca el focus en el primer campo de la ventana y se ejecuta el script el cual consiste en pasar los datos a las cajas de texto. Una vez hecho esto se ejecuta el comando para hacer presionar "Tab" el cual se dirige al siguiente campo, por último, se transfiere e inserta los datos en cada campo. Sin embargo al probarlo más exhaustivamente ese método no resultó ser el más adecuado ya que en un punto de venta no siempre se va a encontrar los campos en el orden que se tiene programado. Por esta razón se procedió a buscar otra manera de realizar el trabajo -la cual se detalla más adelante- y así no depender de los "Tab" que se estaban realizando hasta la fecha.

SPRINT 1: Desarrollo módulos aplicación móvil

En el Sprint 1 se desarrolló la interfaz del inicio de sesión, en su primera versión se manejaba como respuesta del servidor web si era correcta o no la combinación de usuario y contraseña. Más adelante por tema de flexibilidad y rapidez del funcionamiento de la aplicación se incluyó una base de datos interna SQLite, esta base permite almacenar en la memoria del celular los datos correspondientes a las personas y QRs de cada una, así la aplicación podía funcionar sin problemas sin internet. Una vez iniciada la sesión correctamente se

despliega la interfaz de “Lista de personas”, la cual es la pantalla principal de la aplicación móvil porque conecta todos los demás módulos. En la Figura 2.8 se muestran las interfaces de “Inicio de sesión” y “Lista de personas” en su primera versión.

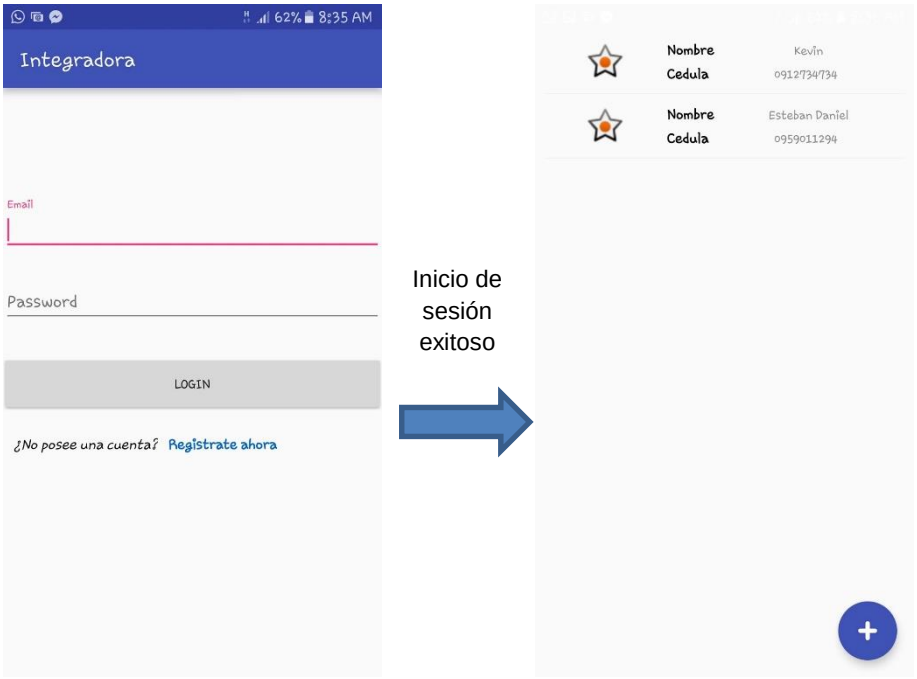
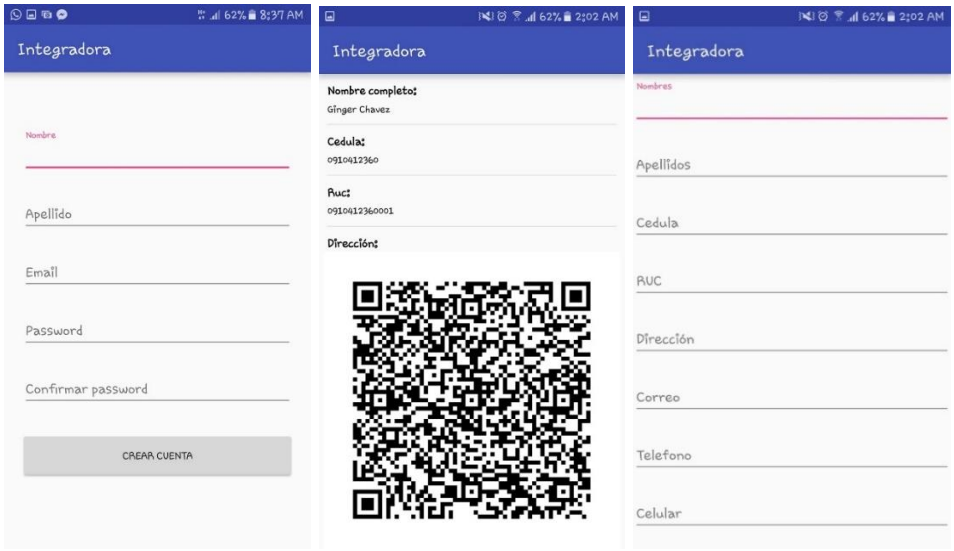


Figura 2.8: Primera versión de interfaces Iniciar sesión y Lista de personas

También se desarrollaron las interfaces de crear usuario para los individuos que utilizan por primera vez la aplicación y no poseen una cuenta, la interfaz de “Crear Persona” y la de “Ver Persona” en donde se realizan las validaciones respectivas para cada campo, además de verificar si la persona a crear es natural o jurídica.

En la pantalla de crear persona se realiza el proceso de generación del código QR una vez que se haya creado la persona. El código QR es inicialmente un bitmap pero se guarda en la base de datos como una cadena de String codificada en base 64. En la Figura 2.9 se muestran las interfaces de “Crear Usuario”, “Crear Persona” y “Ver Persona” en su primera versión.



(a) (b) (c)

Figura 2.9: Interfaces de Crear Usuario (a), Ver Persona (b) y Crear Persona (c) en su primera versión

Sprint 2: Desarrollo de módulos aplicación móvil y aplicación de escritorio.

En el sprint 2 se empezó a trabajar en la aplicación de escritorio, tanto en Windows como en Linux, se desarrollaron primeras versiones que permitían realizar la conexión con la cámara webcam, detectar el código QR y decodificarlo para obtener los datos contenidos en él. La versión de Windows fue desarrollada en C# y la versión de Linux fue desarrollada en Python.

Cabe destacar que se utilizó una librería externa open source llamada ZXing [17] tanto para la generación del código QR (en la aplicación móvil) como para la lectura del mismo (en la aplicación de escritorio Windows y Linux). En la Figura 2.10 se muestra la primera versión de la aplicación de escritorio QRInvoice.

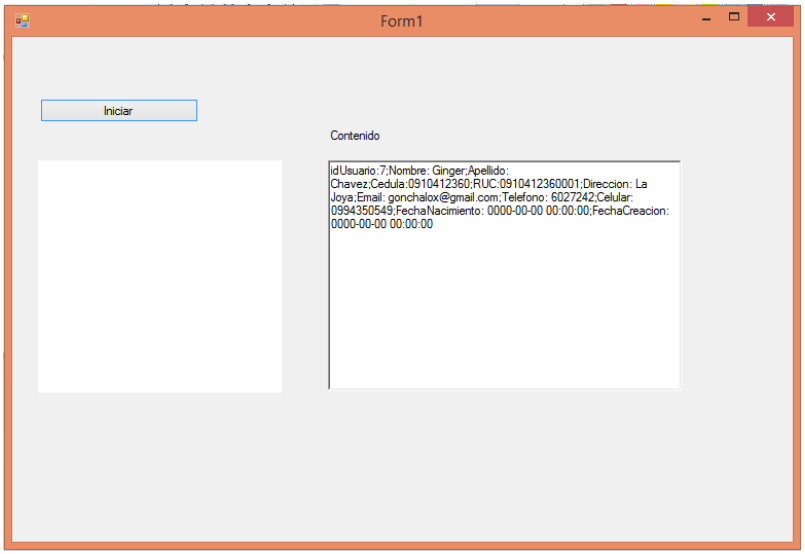
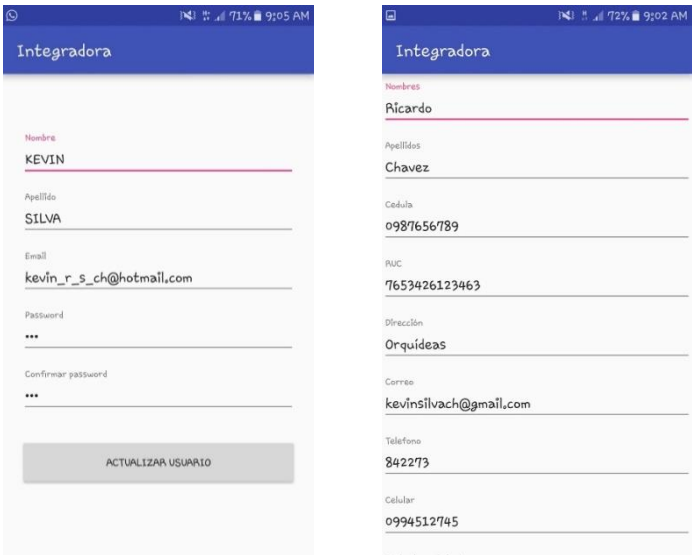


Figura 2.10: Primera versión modulo lectura QR de la aplicación de escritorio

También se desarrolló la interfaz de “Editar Usuario” que permite modificar los datos del usuario inclusive la contraseña. Se desarrollaron las interfaces de “Editar Persona” y “Eliminar Persona” con el objetivo de cambiar información personal de la persona o en caso de querer eliminar una persona. En la Figura 2.11 se muestran las interfaces de “Editar Usuario” y “Editar Persona”.



(a) (b)

Figura 2.11: Interfaces Editar usuario (a) y Editar Persona (b)

Sprint 3: Mejora de Interfaz y desarrollo de modulo configuración en la aplicación de escritorio.

En el Sprint 3 se desarrolló el módulo de configuración de la aplicación de escritorio (Windows y Linux) para que el cajero del punto de venta pueda configurar campo a campo toda la información que se necesita en el proceso de facturación. En la Figura 2.12 se muestra la interfaz de la ventana de configuración.

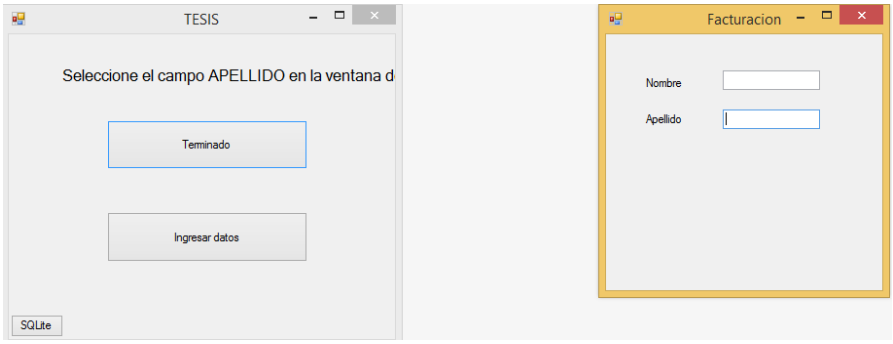


Figura 2.12: Ventana de configuracion de aplicación de escritorio.

También se procedió a realizar mejoras a la interfaz de todo el sistema (móvil y escritorio) con el fin de que QRInvoice se maneje con una paleta estándar de colores, que las interfaces sean parecidas y que sean amigables para el usuario. En las Figuras 2.13 y 2.14 se muestran las diversas pantallas de la aplicación en su versión final.

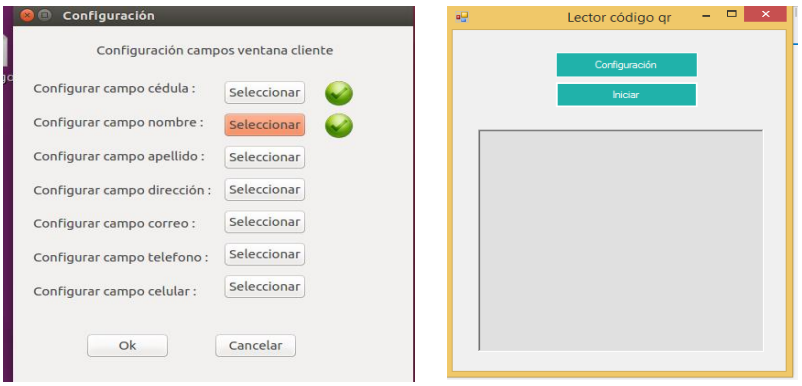


Figura 2.13: Versión final de la aplicación de escritorio

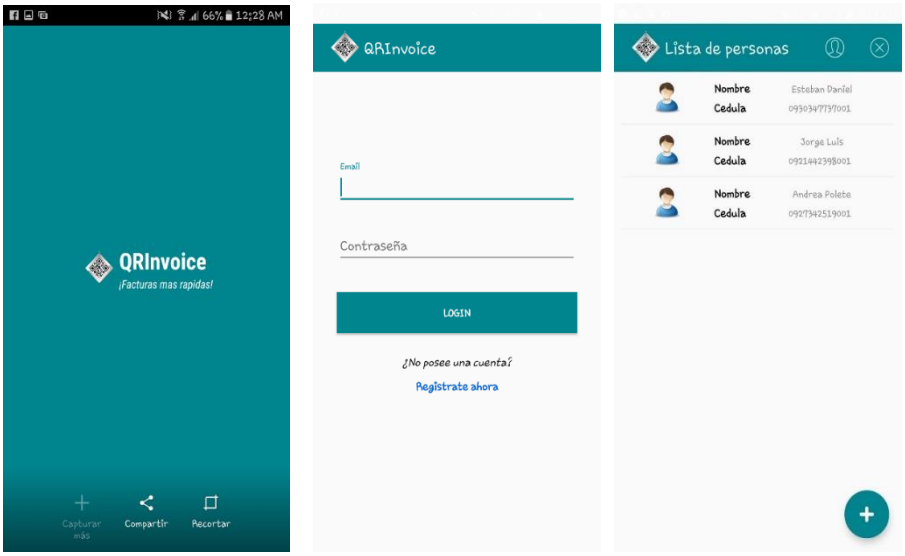


Figura 2.14: Apariencia final de la aplicación móvil

2.2.4 Fase de test

En esta fase se utilizó una serie de aplicaciones demo para emular una ventana de registro de factura en un punto de venta con el fin de saber si el sistema realizaba la lectura e inserción de los datos en la ventana de factura. Las aplicaciones demo fueron creadas en diferentes lenguajes de programación (C#, QT y Java) para observar que el comportamiento del sistema sea el mismo sin importar el lenguaje en el que esté escrito el software de punto de venta. También se procedió a mostrar la aplicación móvil a 5 personas con el objetivo de recibir una retroalimentación en cuanto a la interfaz y terminar de probar la funcionalidad del aplicativo móvil resultando algunos fallos los que se pudieron resolver a tiempo.

CAPÍTULO 3

3. OBTENCION DE REFERENCIA A OBJETOS GRAFICOS DE LA VENTANA EN WINDOWS Y LINUX

Una de las maneras que permitió saber la ubicación exacta de un campo en la ventana del registro de cliente es utilizando un juego de coordenadas relativas y absolutas. En la Figura 3.1 se muestra la ventana absoluta que es prácticamente la que se puede ver en la pantalla de la computadora, y la ventana relativa es tomada con respecto a la ventana del registro de cliente.

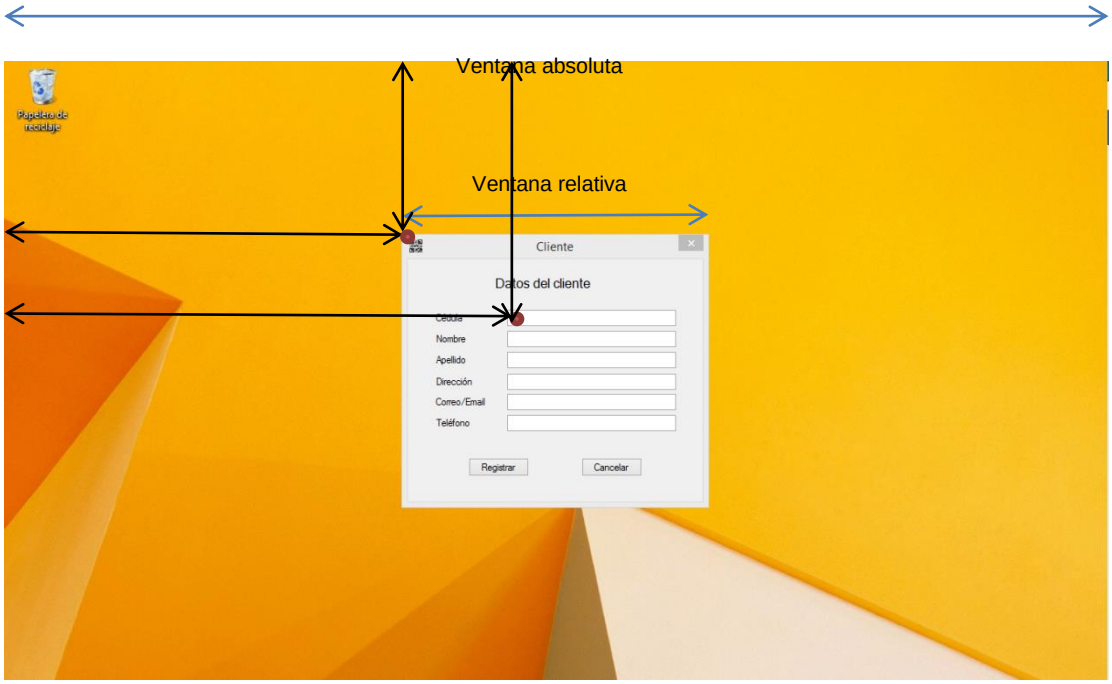


Figura 3.1: Ventana absoluta y relativa

En la Figura 3.2 se muestra de donde salen los componentes de la ecuación para encontrar las coordenadas relativas del control.

Figura 3.2: Cálculo de posiciones en ventana de registro de cliente

$$deltaY = Yr - Ya \qquad (3.1)$$

$$deltaX = Xr - Xa \qquad (3.2)$$

Donde las ecuaciones 3.1 y 3.2 permiten saber la posición del campo con referencia al origen de la ventana. Esto se lo hace una sola vez cuando el usuario debe de configurar la ventana de registro de cliente.

Para saber la posición actual del campo dentro de la ventana de facturación (ver las ecuaciones 3.3 y 3.4) se le suma a la posición actual de dicha ventana el respectivo *deltaX* y *deltaY*. Los resultados se almacenan en un archivo de texto plano que se genera al momento de la configuración.

Para saber la posición y el nombre del control en donde se desea insertar el texto (por ejemplo: nombre), se debe leer desde el archivo donde se almacenó el *deltaX* y el *deltaY* respectivos y realizar el siguiente calculo:

Posición campo nombre:

$$X = deltaX + (\text{posición actual de ventana}) \qquad (3.3)$$

$$Y = deltaY + (\text{posición actual de ventana}) \qquad (3.4)$$

En donde *X* y *Y* (descritas en las ecuaciones 3.3 y 3.4) corresponden a la posición del campo donde se va a insertar el texto.

El archivo de texto se genera al momento de la configuración, el cual contiene *deltaX* y *deltaY* por cada campo en la ventana de facturación. Este archivo maneja

el siguiente formato: [CAMPO],[DELTA_X],[DELTA_Y]. En la Figura 3.4 se puede ver un ejemplo de un archivo de configuración generado en la aplicación de Linux.

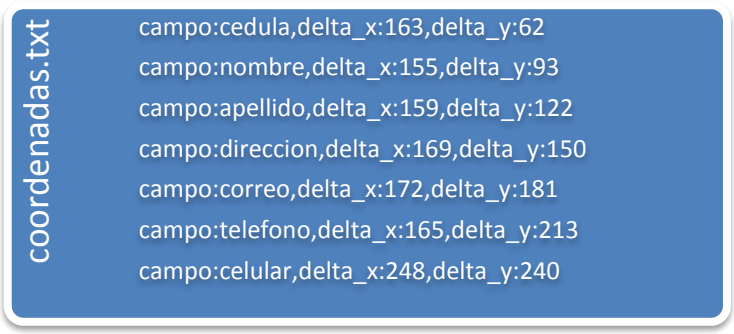


Figura 3.4: Ejemplo del formato del archivo generado en la configuración

De esta manera se puede insertar, por medio de Windows Hooks [15] o xdotool [16] en Linux, los datos respectivos teniendo las coordenadas donde se encuentra cada control y teniendo el texto que debe ir en cada control por medio de la lectura del código QR.

CONCLUSIONES Y RECOMENDACIONES

Tomando en cuenta el análisis hecho del contexto actual de la sociedad y de la manera en la que hoy en día se lleva a cabo el proceso de facturar en un punto de venta, además de las dificultades e incomodidades que produce esto en los clientes, se puede concluir que evidentemente la solución propuesta genera una mayor eficiencia en comparación con el método actual, siendo mucho más rápida y cómoda para las dos partes involucradas en el proceso de venta.

La solución desarrollada llamada QRInvoice de acuerdo al análisis previo hecho, y a las pruebas que se realizaron, es útil para aquellas personas que deben realizar sus compras a nombre propio o a nombre de terceros.

Se deben tomar en cuenta las condiciones de luz en el ambiente al querer realizar la detección del código QR por medio de la cámara con la aplicación de escritorio.

Se pueden realizar a futuro mejoras en la interacción de la aplicación con el usuario como por ejemplo: agregar un avatar personalizado dentro de la aplicación. Todo esto para que el usuario se identifique más y personalice a su conveniencia su aplicación a fin de mejorar la experiencia al usar el aplicativo móvil de QRInvoice.

En un trabajo futuro se puede generar dentro de la aplicación tarjetas imprimibles que contengan el código QR de alguna persona que se desea, de tal manera que en caso de no poder llevar el celular a algún lugar, o quizás en caso de que se acabe la batería del dispositivo y este se apague, se pueda tener a mano el código de la persona en la tarjeta y simplemente mostrarlo.

Se podría hacer rentable el proyecto añadiendo fechas de expiración a los códigos QR, (por ejemplo: un año), y cobrar por la generación de dicho código un valor mínimo tal que al pagar este valor se tenga disponible el código durante el tiempo que se adquirió la licencia de uso.

En un futuro el sistema se podría extender a ser capaz de generar automáticamente la declaración de impuestos de dicha empresa ya que se cuenta con todos los datos

de la factura y se pueden almacenar para usarlos en la generación de la declaración.

BIBLIOGRAFÍA

[1] SRI, «Servicio de Rentas Internas: Declaración de Impuestos,» 2016. [En línea]. Available: http://www.sri.gob.ec/web/guest/declaraciones-de-impuestos.

[2] J. M. Opi, Estructuración del tiempo, Barcelona: Editorial Amat, 2009.

[3] SRI, «Servicio de Rentas Internas: Facturación,» 2016. [En línea]. Available: http://www.sri.gob.ec/web/guest/facturacion.

[4] SRI, «Servicio de Rentas Internas: Guía Tributaria,» Julio 2016. [En línea]. Available: Disponible en: http://multimedia.sri.gob.ec/flipbook/guia-2.html.

[5] CVR, «Estableciendo un Hook,» Guayaquil, 2014.

[6] The Blokehead, Scrum - ¡Guía definitiva de prácticas ágiles esenciales de Scrum!, Babelcube, 2016.

[7] Software Freedom Conservancy, «GIT,» 2016. [En línea]. Available: https://git-scm.com/ .

[8] Atlassian, «Bitbucket,» 2016. [En línea]. Available: https://bitbucket.org/.

[9] Taiga Agile, «Taiga,» LLC, 2016. [En línea]. Available: https://taiga.io/.

[10] Google, «Android developers,» 2016. [En línea]. Available: https://developer.android.com/studio/intro/index.html.

[11] Microsoft, «Visual Studio,» 2016. [En línea]. Available: https://www.visualstudio.com/es/ .

[12] IntelliJ IDEA, «PyCharm,» JetBrains, 2016. [En línea]. Available: https://www.jetbrains.com/pycharm/.

[13] Oracle, «Netbeans,» 2016. [En línea]. Available: https://netbeans.org/.

[14] The Qt Company, «QT Creator IDE,» 2016. [En línea]. Available: https://www.qt.io/ide/.

[15] Microsoft, «Windows Dev Center: Hooks,» 2016. [En línea]. Available: https://msdn.microsoft.com/en-us/library/windows/desktop/ms632589(v=vs.85).aspx.

[16] J. Sissel, «Xdotool - command-line X11 automation tool,» 2016. [En línea]. Available: https://www.semicomplete.com/projects/xdotool/xdotool.xhtml.

[17] GitHub, «Repositorio Zxing,» 2016. [En línea]. Available: https://github.com/zxing/zxing.