UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Aplicación de pago mediante tarjeta regalo

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Jaime Andrés Server Tomás

Tutor: Emilio Vivancos Rubio

2014-2015

Resumen

En el presente trabajo de final de grado voy a diseñar e implementar un sistema de gestión de pagos en comercios mediante tarjetas regalo. La aplicación debe de ser lo suficientemente sencilla para que el pequeño comercio pueda utilizarla fácilmente, con usuarios con pocos conocimientos informáticos y en equipos informáticos con poco poder computacional. La lectura de la tarjeta regalo se realizará mediante códigos QR generados por la aplicación. Para ello, se creará una aplicación capaz de crear y leer los códigos. Desde la misma aplicación se podrá dar de alta una tarjeta regalo generando un código QR y asignando un saldo inicial. Al crear una nueva tarjeta regalo, ésta se registrará en la base de datos. Una vez creada, se podrán realizar pagos. Los movimientos del saldo de la tarjeta quedarán registrados en la base de datos (ya sean cobros, devoluciones o nuevas recargas), así como el saldo disponible.

Palabras clave: TPV, Código QR

Abstract

In this final degree work I am going to design and implement an application to retail payments via gift cards. The application must be easy enough in order to be used by users with little computer knowledge, in small businesses with only a simple personal computer. The gift cards will have a QR code generated by the application. This QR code will be used to store a number that identifies the card. To do this, an application will be created in able to create and read QR codes. From the same application we can create a gift card generating a QR code and assigning the card initial credit. Once created, the gift card information will be recorded in the database and then, the owner of the gift card can make payments. The movements of the card will be registered in the database (whether receipts, returns, or new refills) and so, the available balance.

Keywords : POS, QR code.

Tabla de contenidos

1. Introducción

En este proyecto se diseña y desarrolla una aplicación para la gestión de un sistema de pago de tarjetas regalo para pequeños negocios. Las tarjetas regalo son un servicio muy útil para los compradores que actualmente es prestado principalmente por las grandes cadenas comerciales. El pequeño comercio de proximidad debe buscar la forma de ofrecer servicios (incluidas las tarjetas regalo) pero teniendo en cuenta sus limitaciones. La aplicación que se propone utiliza muy pocos recursos, apenas requiere formación para el personal y está diseñada para ser fácilmente escalable (por ejemplo para ser empleada por asociaciones de comerciantes).

La aplicación es capaz de crear códigos QR que se usarán como tarjeta regalo. Para crear un código QR, se necesitan los datos del cliente (DNI, nombre y apellidos). Tras esto, se podrá crear una tarjeta asignada a un cliente creado y con un saldo inicial que abonará el cliente. Con esto, se creará un código QR asociado al número de tarjeta generado por la aplicación y se podrá, o bien enviar el código QR a una dirección de e-mail, o bien imprimir el código.

Cuando se utilice la tarjeta regalo para efectuar un pago , se puede insertar manualmente el número de la tarjeta o leer el código desde una webcam conectada al ordenador que lo ejecute. Al efectuarse esta operación, se descontará el importe de la compra del saldo disponible en la tarjeta. Si la tarjeta no dispone de suficiente saldo se informará del problema para permitir realizar un pago parcial con otro medio de pago.

La aplicación también dispone de otras funciones, como recargar el saldo de una tarjeta, mostrar un listado de los clientes y poder exportarlos a un Excel, recuperar una tarjeta en caso de extravío, y mostrar el saldo y las operaciones de una tarjeta.

Contexto y motivación

Una tarjeta regalo es una tarjeta precargada con la que se pueden hacer compras en una cadena de establecimientos o en uno concreto. Físicamente suelen ser similares a una tarjeta de crédito: un rectángulo de plástico con una banda magnética para leerla.

Normalmente estas tarjetas se usan cuando una persona quiere hacer un regalo pero no sabe exactamente que regalar pero también pueden usarse como medio para realizar el reembolso del valor de un producto devuelto..

Debido al coste que conlleva incorporar este sistema, prácticamente sólo se puede encontrar estas tarjetas en grandes establecimientos. En España, hay varias cadenas donde se usan este tipo de tarjetas, tales como El Corte Inglés, Mediamarkt, Carrefour, Eroski, Decathlon, Zara...

Algunos ejemplos de tarjetas regalo.

Objetivos

Con este trabajo se intenta desarrollar un sistema de pago con tarjetas regalo de bajo coste, enfocado para los pequeños y medianos establecimientos. Para ello, en vez de usar tarjetas de banda magnética, se usan códigos QR.

Los códigos usados como tarjeta regalo son generados y leídos por la aplicación a través de la webcam de un ordenador, lo que conlleva un ahorro en cuanto a la creación de la tarjeta y a la adaptación de los datafonos para leer dicha tarjeta.

Como la tarjeta regalo se puede enviar por e-mail, también conlleva a un menor impacto en el medio ambiente, al no usar plástico que pueda dañarlo. Aun así, también se ha añadido la opción de imprimir el código por si el cliente no dispone de correo electrónico para recibir la tarjeta.

La aplicación también puede conectarse a un servidor local. Lo que para un solo establecimiento anula completamente los gastos de mantenimiento de un servidor externo.

También se intentará que la aplicación sea lo suficientemente ligera para que se pueda ejecutar en un ordenador sencillo y que no se requiera un equipo de gran potencia.

Al estar destinada para pequeñas empresas, la aplicación deberá poder ser usada por usuarios con pocos conocimientos informáticos. Para ello, tendrá que ser lo suficientemente intuitiva.

Ejemplo de un código QR.

2. Conceptos desarrollados

Para desarrollar este trabajo he puesto en práctica los conocimientos básicos del lenguaje Java, que he aprendido en las asignaturas de Introducción a la Informática y la Programación, Programación, Estructuras de Datos y Algoritmos.

También se ha aplicado técnicas de diseño multicapa aprendidos en las asignaturas de Ingeniería del Software y Diseño del Software.

Debido al uso de la webcam, también ha hecho falta conocimientos de concurrencia aprendidos en la asignatura de Concurrencia y Sistemas Distribuidos.

Y al usar una base de datos, se ha aplicado los conceptos aprendidos en Bases de Datos y sistemas de la información.

Para evitar posibles errores y asegurar un fácil mantenimiento, se usan las habilidades aprendidas en Mantenimiento y Evolución del Software.

Durante el desarrollo de este trabajo, se ha aplicado los conceptos de organización aprendidos en Proceso del Software y Proyecto de Ingeniería del Software.

Diseño multicapa

Se ha desarrollado la aplicación usando un diseño de 3 capas: lógica, persistencia y presentación.

La capa lógica es la encargada de proporcionar la funcionalidad de la aplicación. En ella se crea los objetos que se van a usar para modelar el programa. Esto permite crear nuevas clases, eliminarlas o fusionarlas con otras, crear relaciones y modificar relaciones ya existentes.

La capa de persistencia es la encargada de comunicarse con la base de datos. En ella se cargan los drivers necesarios para conectar con la base de datos. También se encuentran las instrucciones que consultan y modifican la base de datos. Se creará una clase para cada tabla que haya en la base de datos, y así, es más fácil encapsular el acceso a la base de datos al tener todos los accesos a ella en un nivel.

La capa de presentación incluye las clases que definen las interfaces del programa. En ella se introducen los datos del usuario y muestra los resultados. Esta capa se comunica tanto con la capa lógica como la de persistencia, ya que se crean objetos localizados en la capa lógica y se realizan consultas y modificaciones a través de la capa de persistencia.

Este diseño supone la ventaja de poder aislar las funciones de la aplicación en componentes separados, significando en un menor esfuerzo para la mantenibilidad del mismo.

Programación concurrente

La programación secuencial es aquella en que se ejecutan las instrucciones en serie, mientras que la programación concurrente se crea varias actividades que se ejecutan a la vez. (Muñoz Escoí, y otros, 2013)

Para este trabajo, será necesario el uso de la programación concurrente para la lectura de los códigos.

Serán necesarios dos hilos que se comuniquen entre sí. Uno para procesar las imágenes de la webcam y otro para realizar la función propia de la clase.

Base de datos relacionales

Para el trabajo se va a hacer uso de una base de datos relacional para almacenar los datos que se crean en la aplicación.

Una base de datos se define como "la tecnología informática disponible para la organización y gestión de grandes volúmenes de datos" y el modelo relacional se define como "un conjunto de estructuras de datos donde éstas son definidas en el mismo marco teórico [...], es decir, definiendo sus operadores asociados" (Celma Giménez, Casamayor Ródenas, & Mota Herranz, 2003).

Será necesario crear una base de datos ya que se hace uso de grandes volúmenes de datos y también, se tiene almacenar y modificar dichos datos de una manera no volátil.

También se necesita que se almacene la relación que existe entre dichos datos. Por eso, vamos a usar un modelo relacional.

Mantenimiento

El mantenimiento permite a las aplicaciones adaptarse a los cambios, ya sean de entorno o de requisitos. Ya que la aplicación intenta ser válida para usarse en diferentes establecimientos, es importante que tenga un alto índice de mantenibilidad.

Para ello se prestará atención al diseño que, como se ha explicado anteriormente, será una arquitectura multicapa que separe las distintas funciones de las clases.

También se intentará que se use una implementación sencilla, para que el código sea entendible.

Y se intentará que se tenga un número mínimo de bugs para prevenir posibles problemas con la aplicación y alargar el ciclo de vida del mismo.

3. Entorno de trabajo

Esta aplicación se ha realizado en Java, usando el entorno de desarrollo Eclipse (Eclipse). La base de datos se ha desarrollado en MySQL (Oracle).

Para desarrollar la aplicación ha hecho falta el uso de múltiples librerías para Java.

Las librerías usadas son mysql-connector, para la conexión de la aplicación con la base de datos; zxing, que es una de las librerías más usadas para trabajar con códigos QR (Zxing); webcam-capture, para el uso de la webcam (Firyn); y Apache POI, que permite trabajar con hojas de cálculo Excel (Apache, 2015).

Eclipse

Eclipse es un entorno de desarrollo integrado con el que se puede trabajar con múltiples lenguajes de programación, de entre ellos Java, que usaremos en este trabajo. (D'Anjou, Fairbrother, Kehn, Kellerman, & McCarthy, 2005)

Además de la librería de Java, se le puede incluir librerías adicionales, como las que se menciona en los siguientes puntos.

También incorpora plug-ins que van a ser usados en este trabajo como WindowBuilder, que será usado para realizar las interfaces gráficas, así como JUnit con el que se realizarán las pruebas unitarias.

MySQL

MySQL es un sistema de gestión de base de datos en SQL. Se usará para realizar consultas y modificaciones en la base de datos. Tiene las ventajas de ser un sistema seguro, rápido y barato. (DuBois, 2009)

Eclipse no incluye librerías para conectarse a MySQL desde Java. Se puede descargar la librería gratuitamente. (Oracle)

En Java es fácil de usar. Se carga el driver, se realiza la conexión, se ejecuta la consulta o modificación y se cierra la conexión.

Zxing

Zxing es una librería de código abierto para Java capaz de leer códigos QR, así como códigos de barras. (Zxing)

Es capaz de crear códigos especificando el contenido, el tipo de código y el tamaño que ha de tener el código resultante. También es capaz de descifrarlos. Para este trabajo, se usará esta librería para la generación y lectura de códigos QR.

Webcam-capture

Es una librería que permite trabajar con webcams en Java. Puede trabajar tanto con webcams ya incorporadas en el ordenador como externas. (Firyn)

Combinada con Zxing, esta librería servirá para leer los códigos QR mientras enseña su contenido en la aplicación.

Apache POI

Apache POI es una librería que permite trabajar Microsoft Office. Puede trabajar tanto con documentos Office 97-2007 (.doc .xls) como con documentos Office 2007 (.docx .xlsx)

Se usará para exportar la lista de los clientes registrados en la aplicación a un Excel. La pega es que no es una librería fácil de usar, ya que posee un número elevado de opciones y se necesitan importar varios archivos .jar (poi-3.12-20150511.jar, poi-excelant-3.12-20150511.jar, poi-ooxml-3.12-20150511.jar y poi-ooxml-schemas-3.12-20150511.jar), además de otra librería externa (xmlbeans-2.4.0.jar). Pero analizando otras opciones, esta librería es la más fiable para crear documentos Excel.

4. Resumen de la aplicación

El código está estructurado en tres capas: Lógica, que contiene los objetos a usar; persistencia, que se comunica con la base de datos mysql; y presentación, que incluye las interfaces del programa. La aplicación se ejecuta desde el método main de la clase "Ventana Principal" localizada en la capa de presentación. Desde esta clase se podrá acceder a todas las funcionalidades de la aplicación, tales como crear un cliente, una tarjeta, una operación, consultar el saldo de una tarjeta, consultar las operaciones de una tarjeta, recuperar la tarjeta de un cliente insertando el dni y sacar una lista de los clientes y exportarlos a un Excel.

La aplicación comunicará con la base de datos, previamente creada en el localhost con nombre tarjetasregalo y con la estructura que se explica en el siguiente punto.

5. Base de datos

La aplicación está conectada a una base de datos mysql. A continuación aparece el script usado para crearla.

```
CREATE TABLE CLIENTE (
        DNI CHAR(9) NOT NULL
      , NOMBRE CHAR(20)
      , APELLIDOS CHAR(50)
      , PRIMARY KEY (DNI)
);

CREATE TABLE TARJETA (
        NUMERO INTEGER NOT NULL
      , DNI CHAR(9)
      , SALDO DOUBLE
      , FECHA CHAR(10)
      , PRIMARY KEY (NUMERO)
      , CONSTRAINT FK_TARJETA FOREIGN KEY (DNI)
                   REFERENCES CLIENTE (DNI)
);

CREATE TABLE OPERACION (
        ID INTEGER NOT NULL
      , IMPORTE DOUBLE
      , NUMERO INTEGER NOT NULL
      , FECHA CHAR(10)
      , PRIMARY KEY (ID)
      , CONSTRAINT FK_OPERACION FOREIGN KEY (NUMERO)
                   REFERENCES TARJETA (NUMERO)
);
```

La tabla cliente contiene el DNI, nombre y apellidos del cliente.

La tabla tarjeta contiene el número de la misma, el saldo, la fecha que se usó la tarjeta por última vez y el DNI del cliente asociado a la tarjeta.

La tabla operación contiene un número identificativo, el importe, la fecha de realización y el número de la tarjeta usada.

6. Capa lógica

Cliente

La clase Cliente contiene las variables de tipo String dni, nombre y apellidos. Incluye un constructor de la clase con las tres variables, asi como los getters y setters de las variables.

```
package logica;

public class Cliente {
    private String dni;
    private String nombre;
    private String apellidos;

    public Cliente(String dni, String nombre, String apellidos) {
        this.dni=dni;
        this.nombre=nombre;
        this.apellidos=apellidos;
    }

    public String getDni() {
        return dni;
    }
    public String getNombre() {
        return nombre;
    }
    public String getApellidos() {
        return apellidos;
    }
    public void setDni(String dni) {
        this.dni = dni;
    }
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
    public void setApellidos(String apellidos) {
        this.apellidos = apellidos;
    }
}
```

Fecha

Se ha creado una clase propia para la fecha por comodidad. Incluye un constructor vacío que crea una fecha con el día actual, un constructor que crea una fecha a partir de una String con formato “dd/mm/aaaa”, una función toString que devuelve la fecha en formato “dd/mm/aaaa”, un compareTo y los getters y setters.

```
package logica;

import java.time.*;

public class Fecha {

    private int dia;
    private int mes;
    private int año;

    public Fecha() {
        dia=LocalDateTime.now().getDayOfMonth();
        mes=LocalDateTime.now().getMonthValue();
        año=LocalDateTime.now().getYear();
    }

    public Fecha(String s){
        this.dia=Integer.parseInt(s.substring(0, 2));
        this.mes=Integer.parseInt(s.substring(3, 5));
        this.año=Integer.parseInt(s.substring(6, 10));
    }

    public int getDia() {
        return dia;
    }

    public int getMes() {
        return mes;
    }

    public int getAño() {
        return año;
    }
```

```
    public void setDia(int dia) {
        this.dia = dia;
    }

    public void setMes(int mes) {
        this.mes = mes;
    }

    public void setAño(int año) {
        this.año = año;
    }

    public String toString(){
        return String.format("%02d", dia)+"/"+String.format("%02d", mes)+"/"+String.format("%04d", año);
    }

    public int compareTo(Fecha fecha) {
        if (this.año>fecha.año) return -1;
        else {
            if (this.año<fecha.año) return 1;
            else {
                if (this.mes>fecha.mes) return -1;
                else {
                    if (this.mes<fecha.mes) return 1;
                    else {
                        if (this.dia>fecha.dia) return -1;
                        else {
                            if (this.dia<fecha.dia) return 1;
                            else return 0;
                        }
                    }
                }
            }
        }
    }
}
```

Tarjeta

La clase Tarjeta tiene asociadas las variables número, saldo, un cliente y una fecha. La fecha de la tarjeta indica la fecha del último uso de la misma, ya que la tarjeta caduca al año de desuso. Tiene dos constructores, uno con todas las variables y otro sin fecha, que crea una tarjeta con fecha de hoy. También dispone de los getters y setters correspondientes.

```
package logica;

public class Tarjeta {

    private int numero;
    private double saldo;
    private Fecha fecha;
    private Cliente cliente;

    public Tarjeta(int numero, double saldo, Cliente cliente) {
        this.numero=numero;
        this.saldo=saldo;
        this.fecha=new Fecha();
        this.cliente=cliente;
    }

    public Tarjeta(int numero, double saldo, Cliente cliente, Fecha fecha) {
        this.numero=numero;
        this.saldo=saldo;
        this.fecha=fecha;
        this.cliente=cliente;
    }

    public int getNumero() {
        return numero;
    }

    public double getSaldo() {
        return saldo;
    }

    public Fecha getFecha() {
        return fecha;
```

Operación

La clase operación contiene las variables id, importe, una tarjeta y una fecha. De nuevo, hay dos constructores, uno con todas las variables y otro sin fecha que crea una operación con fecha actual. También incluye los getters y setters.

```
package logica;

public class Operacion {

    private int id;
    private double importe;
    private Tarjeta tarjeta;
    private Fecha fecha;

    public Operacion (int id, Tarjeta tarjeta, double importe) {
        this.id=id;
        this.importe=importe;
        this.tarjeta=tarjeta;
        this.fecha=new Fecha();
    }

    public Operacion (int id, Tarjeta tarjeta, double importe, Fecha fecha) {
        this.id=id;
        this.importe=importe;
        this.tarjeta=tarjeta;
        this.fecha=fecha;
    }

    public int getId() {
        return id;
    }

    public double getImporte() {
        return importe;
    }

    public Tarjeta getTarjeta() {
        return tarjeta;
    }
```

QR

Esta clase sólo incluye un constructor que pasándole un número, genera un archivo .png que contiene la imagen del código. Esto es gracias a la librería zxing (Zxing). La imagen se crea en el directorio raíz del proyecto con un tamaño de 200*200

```
package logica;

import java.io.*;
import java.util.*;

import com.google.zxing.BarcodeFormat;
import com.google.zxing.EncodeHintType;
import com.google.zxing.MultiFormatWriter;
import com.google.zxing.NotFoundException;
import com.google.zxing.WriterException;
import com.google.zxing.client.j2se.MatrixToImageWriter;
import com.google.zxing.common.BitMatrix;
import com.google.zxing.qrcode.decoder.ErrorCorrectionLevel;

public class QR {

    public QR(int num) throws WriterException, IOException, NotFoundException{
        String s = ""+num;
        String path = num+".png";
        String codificacion = "UTF-8"; // or "ISO-8859-1"
        Map<EncodeHintType, ErrorCorrectionLevel> map = new HashMap<EncodeHintType, ErrorCorrectionLevel>();
        map.put(EncodeHintType.ERROR_CORRECTION, ErrorCorrectionLevel.L);
        BitMatrix matriz = new MultiFormatWriter().encode(new String(s.getBytes(codificacion), codificacion), BarcodeFormat.QR_CODE, 200, 200, map);
        MatrixToImageWriter.writeToFile(matriz, path.substring(path.lastIndexOf('.') + 1), new File(path));
    }
}
```

7. Capa persistencia

ClienteImp

Es la clase usada para conectar con la tabla cliente de la base de datos. El constructor carga el driver para conectar con la base de datos. También contiene los métodos para realizar modificaciones en dicha tabla.

Dichos métodos son crearCliente, que crea el cliente que se ha dado como argumento, encontrarCliente, que devuelve el cliente con el dni dado como argumento, y encontrarClientes, que devuelve una lista con todos los clientes.

```
package persistencia;

import java.sql.*;
import java.util.*;

import logica.*;

public class ClienteImp{

    public ClienteImp() {
        super();
        try {
            Class.forName("com.mysql.jdbc.Driver").newInstance();
        } catch (Exception e) {
            return;
        }
    }

    public void crearCliente(Cliente c){
        try {
            Connection conexion = DriverManager.getConnection ("jdbc:mysql://localhost/tarjetasregalo","root", "root");
            Statement s = conexion.createStatement();
            s.executeUpdate("insert into cliente values ('"+c.getDni()+"','"+c.getNombre()+"','"+c.getApellidos()+"');");
            conexion.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
```

```
    public Cliente encontrarCliente(String dni){
        Cliente c = null;
        try {
            Connection conexion = DriverManager.getConnection ("jdbc:mysql://localhost/tarjetasregalo","root", "root");
            ResultSet rs = conexion.createStatement().executeQuery("select * from cliente where dni='"+dni+"';");
            if (rs.next()) {
                c = new Cliente(dni, rs.getString("nombre"), rs.getString("apellidos"));
            }
            conexion.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return c;
    }

    public List<Cliente> encontrarClientes(){
        List<Cliente> listaClientes=new ArrayList<Cliente>();
        try{
            Connection conexion = DriverManager.getConnection ("jdbc:mysql://localhost/tarjetasregalo","root", "root");
            ResultSet rs = conexion.createStatement().executeQuery("select * from cliente;");
            try{
                while (rs.next()){
                    Cliente c = encontrarCliente(rs.getString("dni"));
                    listaClientes.add(c);
                }
                return listaClientes;
            } catch (SQLException e) {
                e.printStackTrace();
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return listaClientes;
    }
}
```

TarjetaImp

Es la clase usada para conectar con la tabla tarjeta de la base de datos. El constructor carga el driver para conectar con la base de datos. También contiene los métodos para realizar modificaciones en dicha tabla.

Al igual que en la clase ClienteImp, se incluyen los métodos crearTarjeta, encontrarTarjeta y encontrarTarjetas, que tienen un funcionamiento similar, pero en este caso para la tabla tarjeta. También se incluye los métodos encontrarTarjetasCliente, que devuelve una lista con las tarjetas asociadas al cliente dado como argumento, y actualizarSaldoTarjeta, que modifica el saldo de una tarjeta y actualiza la fecha de la tarjeta, tanto el saldo como la tarjeta se dan como argumento.

```
public List<Tarjeta> encontrarTarjetasCliente(Cliente c){
    try{
        List<Tarjeta> listaTarjetas=new ArrayList<Tarjeta>();
        Connection conexion = DriverManager.getConnection ("jdbc:mysql://localhost/tarjetasregalo","root", "root");
        ResultSet rs = conexion.createStatement().executeQuery("select * from tarjeta where dni='"+c.getDni()+"';");
        while (rs.next()){
            Tarjeta t = encontrarTarjeta(rs.getInt("numero"));
            listaTarjetas.add(t);
        }
        return listaTarjetas;
    } catch (SQLException e) {
        e.printStackTrace();
        return null;
    }
}

public void actualizarSaldo(Tarjeta t, double saldo){
    try {
        Connection conexion = DriverManager.getConnection ("jdbc:mysql://localhost/tarjetasregalo","root", "root");
        Statement s = conexion.createStatement();
        s.executeUpdate("update tarjeta set saldo='"+saldo+"' , fecha='"+new Fecha().toString()+"' where numero='"+t.getNumero()+"';");
        conexion.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

OperacionImp

Es la clase usada para conectar con la tabla operacion de la base de datos. El constructor carga el driver para conectar con la base de datos. También contiene los métodos para realizar modificaciones en dicha tabla.

Al igual que en la clase ClienteImp y TarjetaImp, se incluyen los métodos crearOperacion, encontrarOperacion y encontrarOperacion, que tienen un funcionamiento similar, pero en este caso para la tabla operacion. También se incluye el método encontrarOperacionesTarjeta, que devuelve una lista con las operaciones asociadas a la tarjeta dada como argumento.

```
public List<Operacion> encontrarOperacionesTarjeta(Tarjeta t){
    try {
        List<Operacion> listaOperaciones=new ArrayList<Operacion>();
        Connection conexion = DriverManager.getConnection ("jdbc:mysql://localhost/tarjetasregalo","root", "root");
        ResultSet rs = conexion.createStatement().executeQuery("select * from operacion where numero='"+t.getNumero()+"';");
        while (rs.next()){
            Operacion o = encontrarOperacion(rs.getInt("id"));
            listaOperaciones.add(o);
        }
        return listaOperaciones;
    } catch (SQLException e) {
        return null;
    }
}
```

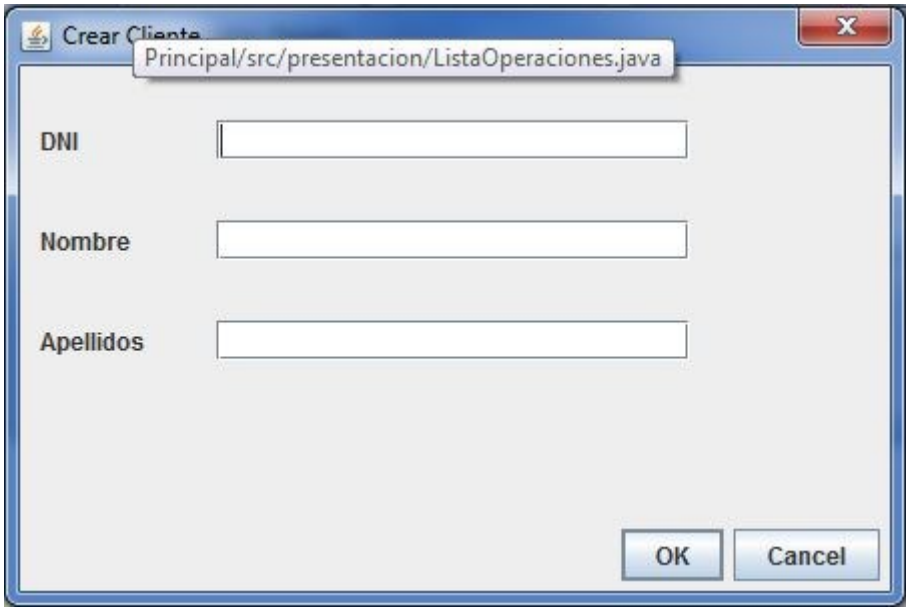
8. Capa presentación

VentanaPrincipal

Es la clase principal de la aplicación. Desde esta ventana se puede acceder a todas las funcionalidades. Consiste en una simple ventana con 8 botones para acceder a las distintas funciones.

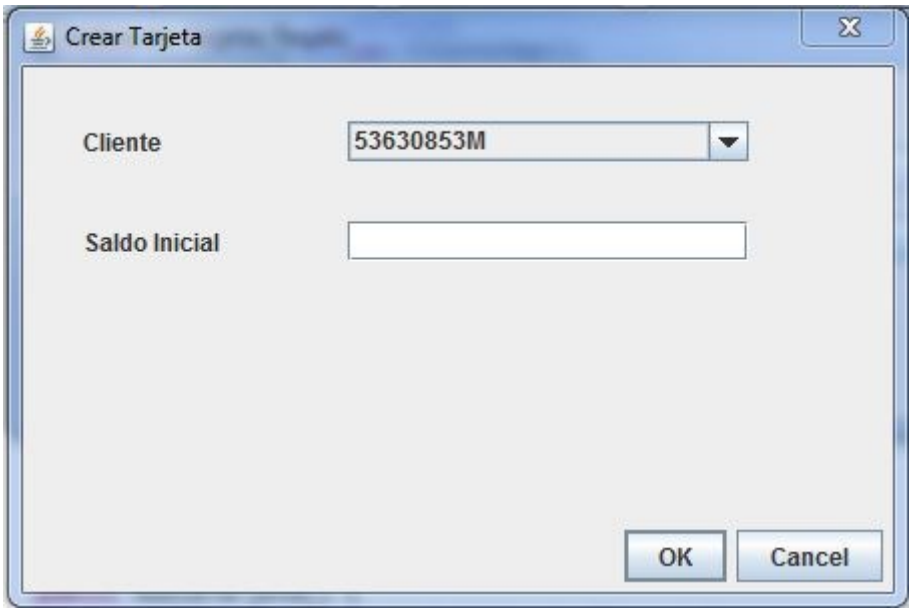
NuevoCliente

En esta ventana se solicitan los datos necesarios para crear un nuevo cliente, o sea, el dni, el nombre y los apellidos. Una vez rellenados los campos, la clase comprueba que no haya campos vacíos y que no haya un cliente existente con el mismo dni. De no ser así, se muestra una ventana informando del problema. En caso contrario, llama al método crearCliente de la clase ClienteImp y se cierra la ventana.



NuevaTarjeta

En esta ventana se solicitan los datos necesarios para crear una nueva tarjeta, o sea, el saldo inicial y un cliente asociado. La fecha se pone por defecto la actual. Una vez rellenados los campos, la clase comprueba que se haya asignado un saldo válido (un número positivo). De no ser así, se muestra una ventana informando del problema. En caso contrario, genera un número aleatorio para la tarjeta (un número entre el 10000000 y el 99999999. Después llama al método crearTarjeta de la clase TarjetaImp. Se crea un nuevo fichero .png conteniendo el código QR asociado al número generado y lo muestra en pantalla llamando a la clase MostrarQr.

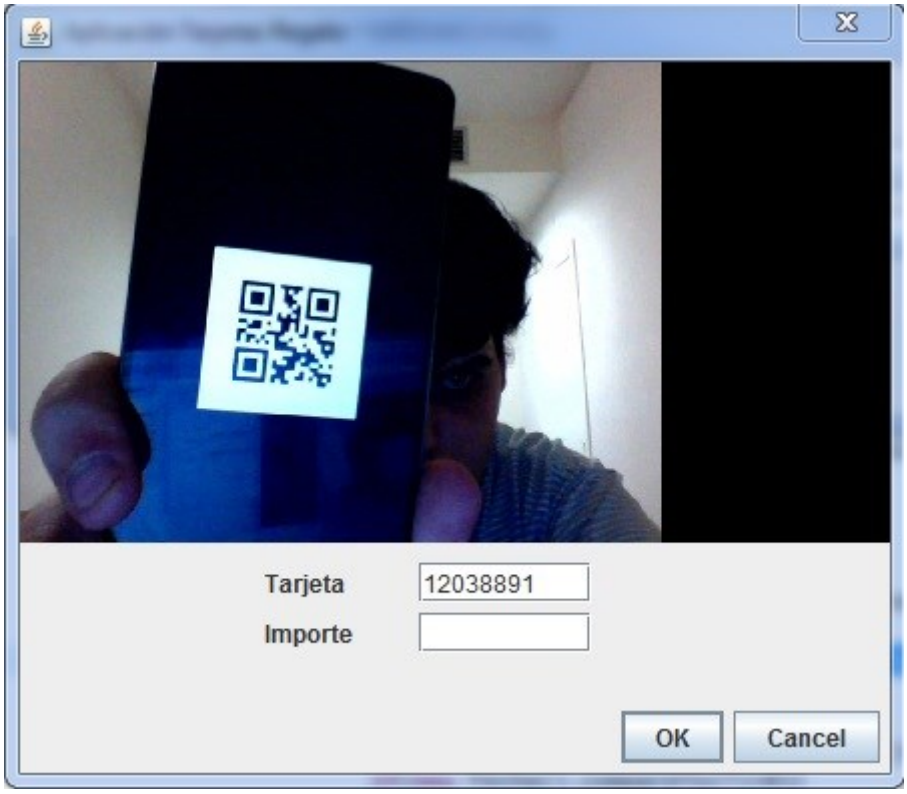


MostrarQr

Esta clase enseña el código QR generado en las clases NuevaTarjeta y ListaTarjetas. En la ventana se ve la imagen de la tarjeta generada. Debajo de la imagen ofrece dos opciones: enviar el código por mail o imprimirlo. Si se decide enviar por mail, se debe de insertar la dirección e-mail a la que se debe enviar en el campo de la izquierda. Si se decide imprimirlo, se puede elegir distintas opciones de impresión. Una vez elegida una opción, se comprueba si se ha especificado una dirección e-mail válida (si se ha elegido esta opción), y envía o imprime el código creado.

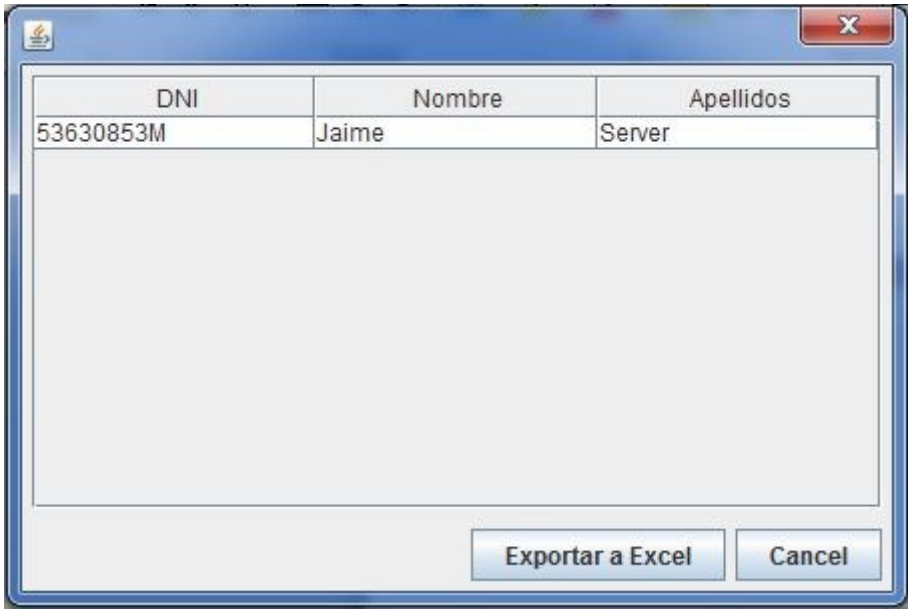
NuevaOperacion

En esta ventana se realizan los nuevos cobros. Al abrir la ventana, se abre una imagen enseñando lo que graba la webcam. Esto ha sido posible gracias a la librería webcam-capture (Firyn). Aunque es posible detectar el código QR por la webcam, también ofrece la posibilidad de insertar el número manualmente. Una vez insertado el número y el importe de la operación y haber clickado Ok, se comprueba que no haya campos vacíos, que el número de tarjeta introducido corresponda a una tarjeta existente, que la tarjeta no esté caducada (ya que caduca al año de desuso), y que la tarjeta tenga suficiente saldo como para realizar la operación. Si todas estas comprobaciones son correctas, se procede a llamar al método crearOperacion de la clase OperacionImp y al método actualizarSaldo de la clase TarjetaImp.



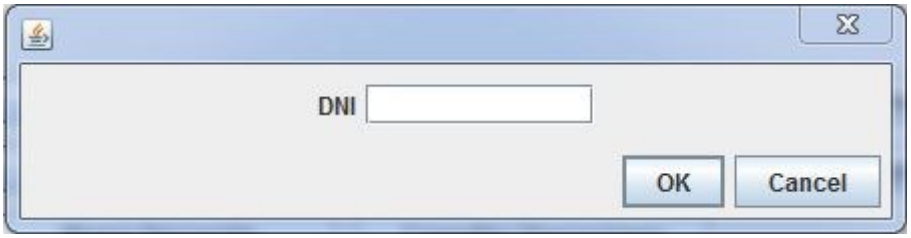
ListaClientes

En esta clase se muestra la lista de clientes. En la ventana se ve los datos de los clientes creados. Hay dos botones, uno para cerrar la ventana y otro para exportar la lista de los clientes a un Excel. En caso de que se exporte, se creará un documento Excel en el directorio raíz de la aplicación con nombre “ClientesAAAA-MM-DD.xlsx”.



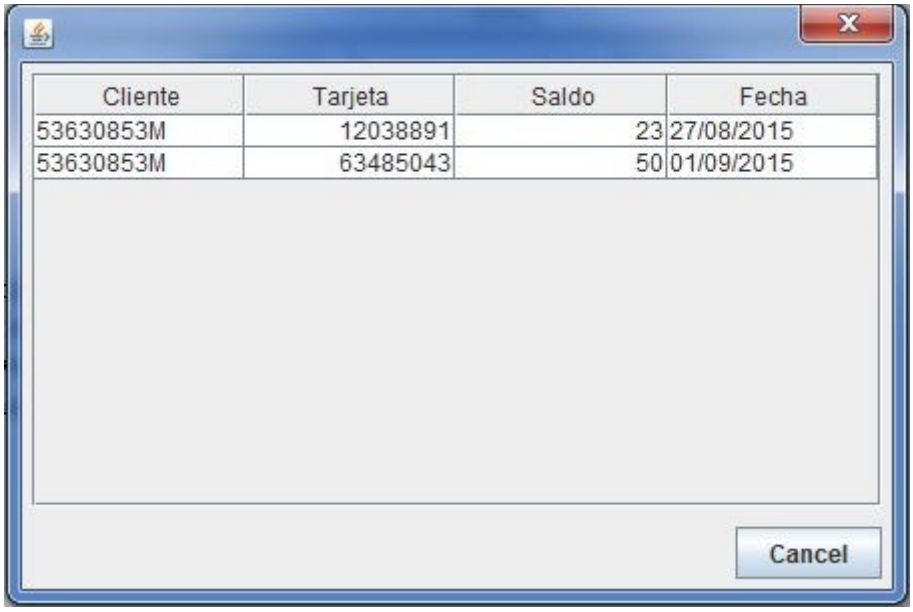
RecuperarTarjeta

Es una ventana sencilla donde se puede insertar el DNI de un cliente. Se comprueba si el DNI es de un cliente existente y si tiene tarjetas creadas. De ser así, se abrirá otra ventana de la clase ListaTarjetas con las tarjetas del cliente.



ListaTarjetas

En esta ventana se muestra las tarjetas asociadas al cliente insertado en la ventana de la clase RecuperarTarjeta. Se muestra en una tabla las tarjetas del cliente. Al hacer doble clic en una de ellas, se abrirá nuevamente una ventana de la clase MostrarQr donde el cliente puede volver a recibir la tarjeta por e-mail o imprimirla.

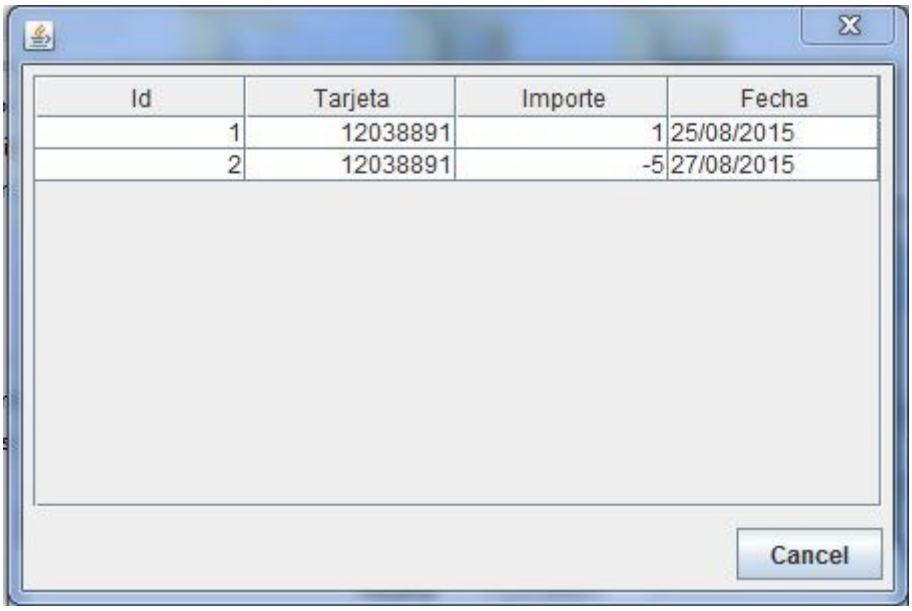


ConsultarOperaciones

Para esta clase se abre una ventana donde aparece lo que ve la webcam y escanea el código QR. Nuevamente, se puede insertar el número de la tarjeta manualmente. Tras esto, se comprueba si existe alguna tarjeta con dicho número. Si es así, abrirá una nueva ventana de la clase ListaOperaciones donde se muestra las operaciones de la tarjeta.

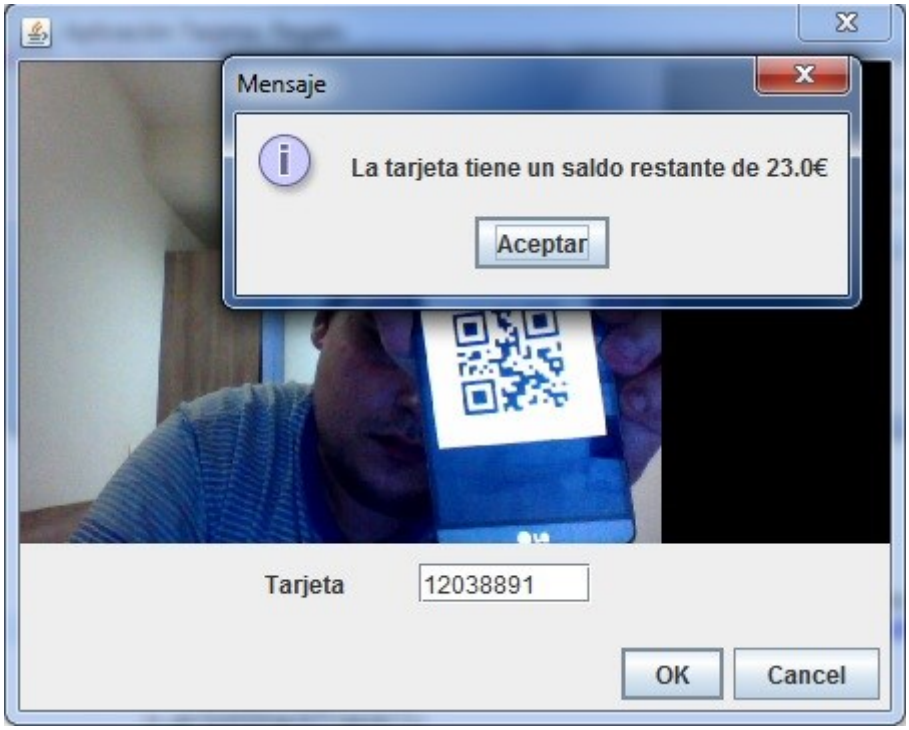
ListaOperaciones

En la clase se muestra las operaciones de la tarjeta insertada. Las operaciones aparecen en una tabla.



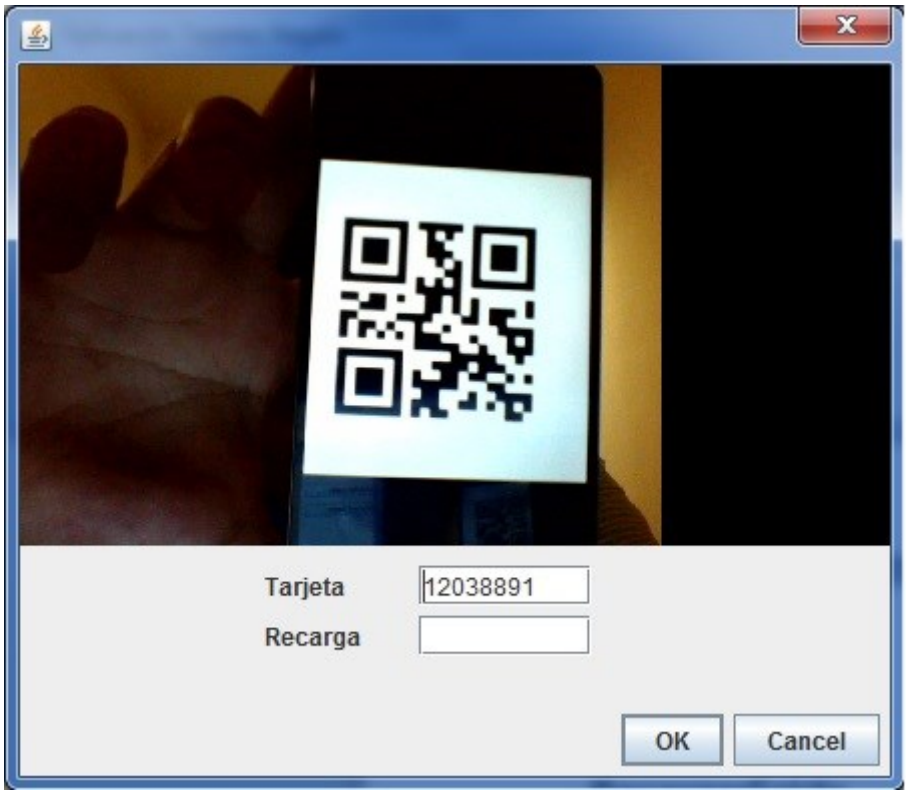
ConsultarSaldo

Aquí se muestra otra ventana donde muestra lo visto por la webcam y un campo donde insertar el código manualmente. Se comprueba si la tarjeta existe y, si es así, aparece una ventana donde nos comunica el saldo de la tarjeta.



RecargarSaldo

En esta clase se puede añadir saldo a la tarjeta. El funcionamiento de la clase es similar al de la clase NuevaOperacion, excepto que esta operación aparece en la tabla operación de la base de datos con importe negativo.



9. Conclusiones

En este proyecto se ha conseguido realizar una aplicación de pago con tarjetas regalo de bajo coste, gracias al uso de los códigos QR. Este programa se ha desarrollado para un solo establecimiento, pero puede ser fácilmente escalable para ser empleado en cadenas de establecimientos o asociaciones locales de comerciantes.

La aplicación es capaz de crear y leer los códigos QR a través de una webcam. Para poder usarla sólo es necesario un ordenador y una webcam, lo que no supone un gran gasto en hardware para adaptar el equipo a esta aplicación.

Ha sido posible incluir las opciones de enviar el código por e-mail y así evitar el uso de papel o plástico para crear una tarjeta, y así tener un menor impacto medioambiental. También se ha podido incluir la opción de imprimir el código QR y, aunque esto signifique el uso de papel, las tarjetas también podrán ser usadas por clientes que no disponen de correo electrónico o les sea más cómoda esta opción.

La aplicación se conecta a un servidor local. Basta con que el ordenador tenga instalado MySQL y que se haya ejecutado el script para crear la base de datos. De esta manera, no será necesario que se tenga un servidor externo para usar la aplicación.

Se ha intentado que la aplicación sea lo suficientemente descriptiva e intuitiva para que pueda ser usada por usuarios con conocimientos básicos de informática. De hecho para usarla sólo se tiene que tener en cuenta 3 pasos sencillos. Primero crear el cliente, luego la tarjeta y por último crear la operación.

10. Futuras ampliaciones

Para un futuro, se ha pensado diversas ampliaciones para hacer la aplicación más atractiva y aplicable a más establecimientos.

Se podría integrar con otro programa de cobro para establecimientos que sea de código abierto para asegurar la completa integración con el mismo.

Es posible mejorar el diseño de las interfaces, ya que el usado hasta ahora es muy básico.

Podría realizarse una mejora para que las tarjetas puedan usarse en distintas tiendas de una misma cadena. Bastaría con conectar la aplicación con una base de datos situada en un servidor externo y añadir en la base de datos un registro para el código de cada tienda.

También se podría crear una pequeña aplicación móvil para que se pueda realizar las lecturas de los códigos QR desde un Smartphone y envíe los datos al ordenador, y así evitar la necesidad de tener una webcam.

11. Bibliografía

Apache. (22 de 07 de 2015). *Apache POI*. Recuperado el 27 de 08 de 2015, de https://poi.apache.org/

Celma Giménez, M., Casamayor Ródenas, J. C., & Mota Herranz, L. (2003). *Base de Datos Relacionales*. Pearson Educación.

D'Anjou, J., Fairbrother, S., Kehn, D., Kellerman, J., & McCarthy, P. (2005). *The Java Developer's Guide to Eclipse*. Boston: Pearson Education.

DuBois, P. (2009). *MySQL Developer's Library*. Pearson Education.

Eclipse. (s.f.). Recuperado el 25 de 08 de 2015, de https://eclipse.org/

Firyn, B. (s.f.). *Github*. Recuperado el 19 de 08 de 2015, de https://github.com/sarxos/webcam-capture

Muñoz Escoí, F. D., Argente Villaplana, E., Espinosa Minguet, A. R., Galdámez Saiz, P., García-Fornes, A., de Juan Marín, R., y otros. (2013). *Concurrència i sistemes distribuïts*. València: Editorial Universitat Politècnica.

Oracle. (s.f.). Recuperado el 25 de 08 de 2015, de https://www.mysql.com/

Oracle. (s.f.). *MySQL download connector*. Recuperado el 1 de 9 de 2015, de http://dev.mysql.com/downloads/connector/j/

Solvermedia. (s.f.). Recuperado el 10 de 8 de 2015, de http://www.solvermedia.com/es/productos/tpv-supermercados-minimarket.html

Zxing. (s.f.). *Github*. Recuperado el 11 de 08 de 2015, de https://github.com/zxing/zxing