

14 DE OCTUBRE DE 2014



TAREA PROGRAMADA #2

ALGORITMOS Y ESTRUCTURAS DE DATOS TI-2402

Kenneth Martínez Avendaño (2014027338)

Silvia Elena Alpízar Rivera (2014128772)

Andrey Sánchez Aguilar (2014010852)

SEGUNDO SEMESTRE DEL 2014

Tabla de contenido

Descripción del problema	2
Diseño del Programa.....	3
.....	6
.....	6
Librerías Utilizadas.....	9
JFree Chart.....	9
Ejemplo de uso	10
JavaMail.....	11
Instalación:	11
Clases y métodos básicos usados:	12
Análisis de Resultados	14
Objetivos Alcanzados	14
Objetivos no alcanzados.....	15
Manual de Usuario	16
Vista General	16
Configuración de datos del banco	17
Registro de clientes	18
Correo electrónico.....	19
Asignación de tiquetes	19
Atención a los clientes	19
Reportes	19
Gráfico de clientes atendidos	20
Gráfico de prioridad.....	20
Lista de clientes	20
Ejecución de reproductor en el terminal de Linux	20
Conclusiones.....	21

Descripción del problema

Este sistema proporcionará de atención a clientes para el BAC San José. La idea es que el sistema les asigne citas de atención a los clientes, para evitar que tengan que hacer largas filas en las sucursales.

Este sistema será utilizado por el guarda de seguridad de las sucursales en las cuales será implementado este sistema. El hará el registro de nuevas personas que ingresen al banco, posteriormente les otorgará un tiquete y les tomará los datos personales incluyendo el nombre y correo electrónico, este último servirá para notificarle al cliente que debe esperar su turno según su prioridad.

Las prioridades se manejan de la siguiente manera:

1. Persona con discapacidad.
2. Adulto mayor.
3. Mujer embarazada.
4. Cliente corporativo.
5. Cliente regular.

La persona con discapacidad tiene la mayor prioridad de atención, le sigue la persona adulta mayor, luego las mujeres embarazadas, los clientes corporativos, y por último el cliente regular.

Ya establecido el orden, la atención de cada cliente será asignado según su prioridad, e inmediatamente serán atendidos; o por el contrario tendrá que esperar varios turnos hasta que le corresponda el suyo.

Cuando llegue su turno el sistema enviará un mensaje vía correo electrónico indicándole al usuario que ya le corresponde.

Mediante la configuración del sistema se podrán establecer la cantidad de cajas existentes de la sucursal y el nombre del banco.

Estos cambios serán evidenciados en la pantalla principal además en esta se podrá agregar el logo del banco.

Este sistema genera reportes de la cantidad de clientes atendidos por fecha y por hora del banco.

También será posible observar los gráficos de las personas que según la prioridad han sido atendidas por el banco. Este se podrá observar mediante dos representaciones un gráfico de barras y un gráfico pastel.

Finalmente podrá observarse una lista que muestra el historial de todas las personas que el banco ha recibido, esta misma tendrá la capacidad e ordenarse por fecha, orden alfabético,

Diseño del Programa

Este programa está diseñado para otorgar una mayor facilidad a la atención del BAC San José.

El primer paso fue la creación de las colas de prioridad, las cuales se manejaron por medio de nodos, lo que significa que cada nueva persona que entra se colocará en una misma fila, los cuales se irán

“colando” con respecto a su prioridad. A continuación se muestran algunos segmentos de código de esta clase.

```
package tareaprogramada2;

public class ColaPrioridad <E>{

    private Nodo primerNodo;
    private Nodo ultimoNodo;
    private int tamaño;
    private int maxTamaño;

    public ColaPrioridad(){
        this.primerNodo = null;
        this.ultimoNodo = null;
        this.tamaño = 0;
    }

    public ColaPrioridad(int maximoTamaño){
        this.primerNodo = null;
        this.ultimoNodo = null;
        this.tamaño = 0;
        this.maxTamaño = maximoTamaño;
    }
}
```

```

public class Nodo{

    // Atributos
    private E dato;
    private int prioridad;
    private Nodo siguiente;
    private Nodo anterior;

    // Constructor con un elemento
    // Persona con discapacidad = 1
    // Adulto Mayor = 2
    // Mujer embarazada = 3
    // Cliente corporativo = 4
    // Cliente regular = 5
    public Nodo(E elemento,String prioridad){

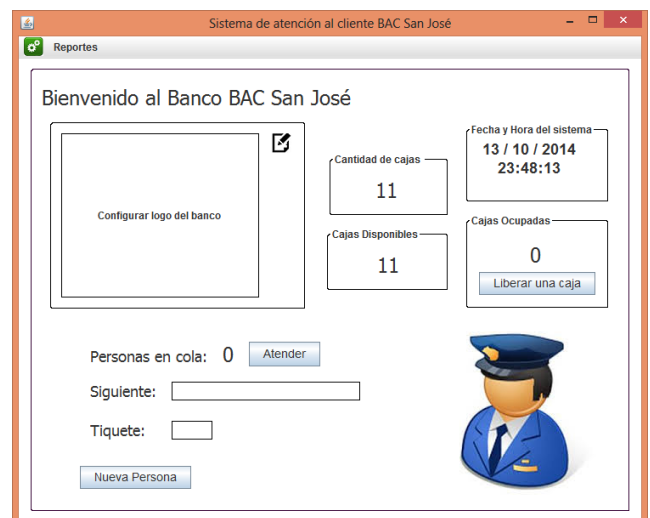
        this.dato = elemento;
        if(prioridad == "Persona con discapacidad")this.prioridad = 1;
        else if(prioridad == "Adulto Mayor")this.prioridad = 2;
        else if(prioridad == "Mujer embarazada")this.prioridad = 3;
        else if(prioridad == "Cliente corporativo")this.prioridad = 4;
        else if(prioridad == "Cliente regular")this.prioridad = 5;

        this.siguiente = null;
        this.anterior = null;

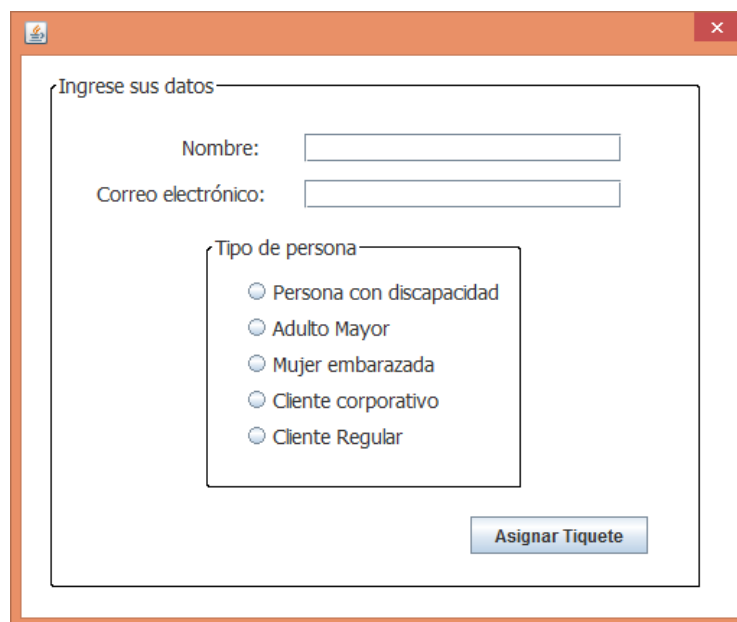
    }
}

```

Luego se implementó una interface que consta de 6 ventanas, las cuales presentan funciones específicas, en la pantalla principal se encuentran las principales funciones, las cuales consisten en observar el manejo de las cajas, cuantas hay disponibles, cuantas hay ocupadas, cuantas hay en total; las personas que estén en la cola, y su paso para ser atendidas por el botón aceptar. En esta pantalla también se encuentra el menú para poder acceder a las demás pantallas.



Para registrar una nueva persona se utilizó un pequeño formulario donde el usuario ingresa su nombre y correo electrónico y su prioridad. A esta ventana se tiene acceso gracias al botón “Nueva Persona”. A la hora de asignar el ticket, se envían los correos electrónicos confirmando los datos del usuario y signan dando a la cola según su prioridad. Para registrar una persona se utilizó una clase por separado, a continuación se mostrarán algunos segmentos del código de dicha clase.

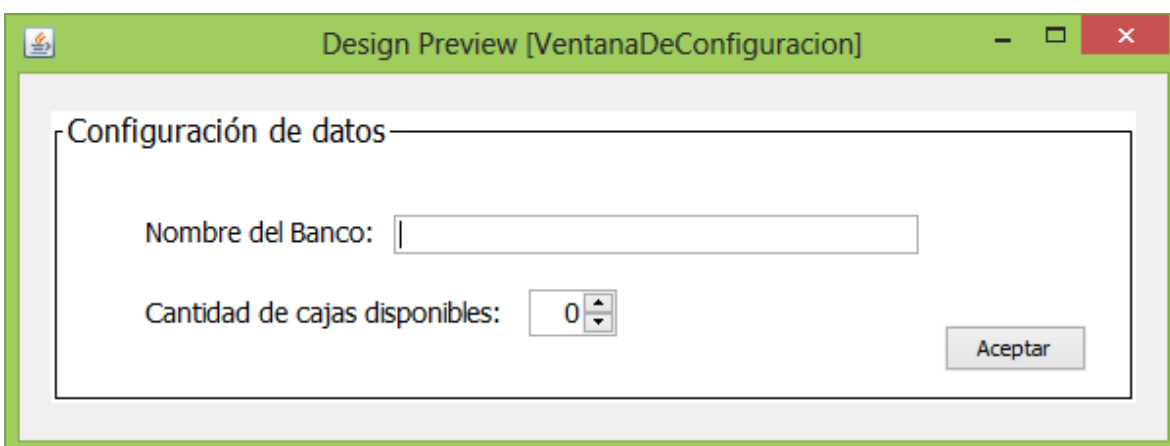


Formulario de registro de una nueva persona. El formulario tiene un título "Ingrese sus datos" y contiene los siguientes campos:

- Nombre:
- Correo electrónico:
- Tipo de persona:
 - ☐ Persona con discapacidad
 - ☐ Adulto Mayor
 - ☐ Mujer embarazada
 - ☐ Cliente corporativo
 - ☐ Cliente Regular

En la parte inferior derecha del formulario hay un botón "Asignar Tiquete".

La pantalla de configuración se creó como una opción del menú, la cual consiste en configurar los datos principales del banco, como su nombre, la cantidad de cajas disponibles, el logo del banco será posible cambiarlo en la pantalla principal.



Pantalla de configuración de datos. El formulario tiene un título "Configuración de datos" y contiene los siguientes campos:

- Nombre del Banco:
- Cantidad de cajas disponibles:

En la parte inferior derecha del formulario hay un botón "Aceptar".

Para el manejo de los datos del banco se realizó una clase banco, a continuación, se presentará segmento del código de esta clase.

```
public class Persona {  
public class Banco {  
  
    /**  
     * Atributos de la clase  
     */  
    public String nombre;  
    public int cantidadCajas;  
    public int cantidadDisponibles;  
  
    /**  
     * Costructor predeterminado  
     */  
    public Banco() {  
    }  
  
    /**  
     * Costructor de la clase banco  
     * @param nombre Se refiere al nombre del banco  
     * @param cajas Cantidas de cajas que contiene el banco  
     */  
    public Banco(String nombre, int cajas) {  
        this.nombre = nombre;  
        this.cantidadCajas = cajas;  
        this.cantidadDisponibles = cajas;  
    }  
  
    /**  
     * Obtiene el nombre del banco  
     * @return El nombre del banco  
     */  
    public String getNombre() {  
        return nombre;  
    }  
  
    /**  
     * Obtiene la cantidad de cajas disponibles  
     * @return La cantidad de cajas disponibles  
     */  
    public int getDisponibles() {  
        return this.cantidadDisponibles;  
    }  
}
```

Para el reloj y el calendario que se ven en la pantalla principal, se utilizó una clase por separado, por medio de hilos tanto el reloj como el calendario se actualizan con los del sistema, haciendo que se presentación sea dinámica. A continuación se presentarán segmentos del código de esta clase.

```
package tareaprogramada2;

import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
import java.util.GregorianCalendar;
import javax.swing.JLabel;

public class Reloj extends JLabel implements Runnable {

    private String dia, mes, año, hora, minutos, segundos;
    private Calendar calendario = new GregorianCalendar();
    Thread hilo;

    /**
     * Constructor predeterminado
     */
    public Reloj() {
    }

    /**
     * Constructor de la clase Reloj
     * @param x Posicion del reloj y el calendario en la pantalla
     * @param y Posicion del reloj y el calendario en la pantalla
     * @param p Posicion del reloj y el calendario en la pantalla
     * @param p1 Posicion del reloj y el calendario en la pantalla
     */
    public Reloj( int x, int y, int p, int p1) {
        this.setBounds(x, y, p, p1); // posicion del reloj y el calendario
        hilo = new Thread(this);
        hilo.start(); // comienza a ejecutarse el hilo
    }
}
```



```

/**
 * Método que permite cambiar los valores del calendario y del reloj
 */
@Override
public void run() {
    Thread hiloActual = Thread.currentThread();
    while (hiloActual == hilo) {
        try {
            actualizar(); // actualiza los datos
            int mesE = Integer.valueOf(mes) + 1;
            this.setText("<html><center>" + dia + " / " + mesE + " / " + año + "<br>" + hora + ":" + mi
            Thread.sleep(1000); // velocidad de los segundos
        } catch (InterruptedException ex) {
            System.out.println(ex.getMessage());
        }
    }
}
}

```

Para el obtener los días meses y años:

```

/**
 * Método que permite actualizar los datos del reloj y la fecha en tiempo real
 */
public void actualizar() {
    Date fechaHoraActual = new Date(); // guardara la fecha y hora actual
    calendario.setTime(fechaHoraActual); //Proporciona la fecha y la hora actual segun el sistema

    hora = String.valueOf(calendario.get(Calendar.HOUR_OF_DAY)); // obtencion de la hora

    minutos = calendario.get(Calendar.MINUTE) > 9 ? "" + calendario.get(Calendar.MINUTE) : "0"
    + calendario.get(Calendar.MINUTE); //obtencion de los minutos

    segundos = calendario.get(Calendar.SECOND) > 9 ? "" + calendario.get(Calendar.SECOND) : "0"
    + calendario.get(Calendar.SECOND); // obtencion de los segundos

    dia = calendario.get(Calendar.DATE) > 9 ? "" + calendario.get(Calendar.DATE) : "0"
    + calendario.get(Calendar.DATE); // optencion del día

    mes = calendario.get(Calendar.MONTH) > 9 ? "" + calendario.get(Calendar.MONTH) : "0"
    + calendario.get(Calendar.MONTH); //optencion del mes

    año = calendario.get(Calendar.YEAR) > 9 ? "" + calendario.get(Calendar.YEAR) : "0"
    + calendario.get(Calendar.YEAR); // optencion del año
}

```

La generación de los reportes es una sección bastante importante de este proyecto, El primer reporte es la lista de todos los clientes que el banco ha recibido, se decidió colocarlos por medio de un JTable, en este será posible ordenar a todos los clientes según la prioridad que tengan.

Los gráficos de las prioridades que tiene el banco serán representados por dos tipos de gráfico, pastel y de barras, estos deberán darme la misma.

Los gráficos de clientes atendidos, corresponden a la representación de todos los clientes que el banco ha atendido en una fecha y hora determinada, es posible consultar sobre clientes atendidos en determinado momento mediante un filtro.

Librerías Utilizadas

JFree Chart

Es una librería que ofrece la posibilidad de crear todo tipo de gráficas de manera sencilla. Esta desarrollada en JAVA y es de código abierto. Las principales características a reseñar son:

- Una Api bien documentada que da soporte a un amplio rango de tipos de gráfica.
- Diseño flexible y fácil de extender pensado para aplicaciones cliente - servidor
- Puede exportar en ficheros gráficos (JPEG, PNG) y archivos (PDF, EPS, SVG)
- Presentación de datos en Servlet y JSP
- Nos ofrece numerosos interfaces para el tratamiento de datos en base al gráfico que queramos crear
- Manejo de gráficos: funciones de zoom, impresión,...

Se permite la representación gráfica en los siguientes formatos:

- Gráficos de tarta: 2d y 3d
- Gráficos de barras: regulares y en pilas. También ofrece la posibilidad de gráficos 3d
- Gráficos de líneas y áreas
- Gráficos de dispersión
- Gráficos de tiempos
- Gráficos combinados
- Diagramas de Gantt

Ejemplo de uso

Cada uno de los posibles gráficos que podemos obtener con JFreeChart necesita unos datos, que son los que se pintarán en dicho gráfico. Estos datos normalmente no se pasan directamente al gráfico, sino que se pasan a una clase encargada de almacenarlos, que se suele conocer como modelo de datos. En este ejemplo vamos a centrarnos en `CategoryDataset`, que nos permitirá pintar gráficos de barras (`BarChart`), de áreas (`AreaChart`), de líneas (`LineChart`) y de tarta múltiple (`MultiplePieChart`).

Vamos metiendo los datos de las visitas

```
/** Sitio web 1 */
private static final String SITIO_2 = "www.sitio1.com";

/** Sitio web 2 */
private static final String SITIO_1 = "www.sitio2.com";

...

// Creamos y rellenamos el modelo de datos
DefaultCategoryDataset dataset = new DefaultCategoryDataset();

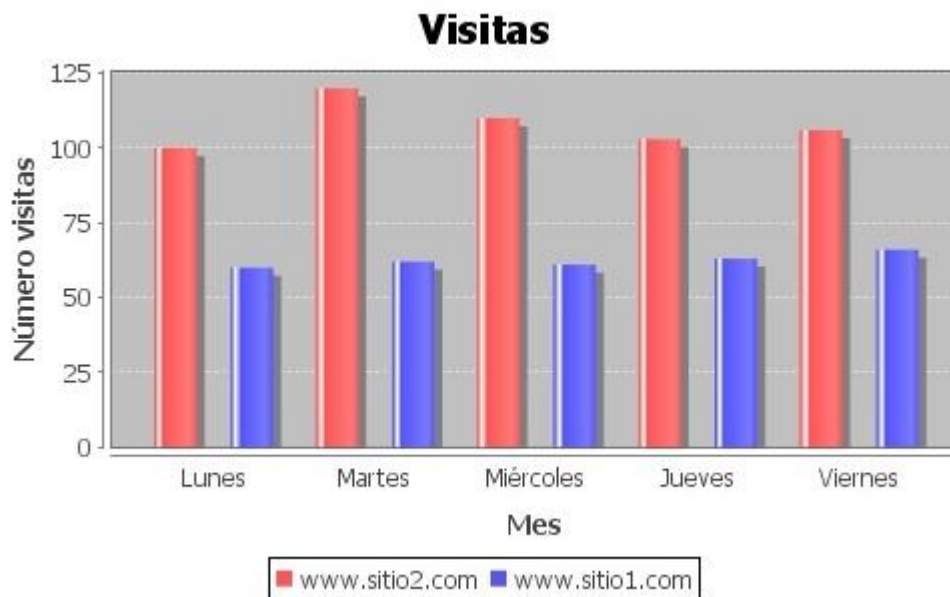
// Visitas del sitio web 1
dataset.setValue(100, SITIO_1, "Lunes");
dataset.setValue(120, SITIO_1, "Martes");
dataset.setValue(110, SITIO_1, "Miércoles");
dataset.setValue(103, SITIO_1, "Jueves");
dataset.setValue(106, SITIO_1, "Viernes");

// Visitas del sitio web 2
dataset.setValue(60, SITIO_2, "Lunes");
dataset.setValue(62, SITIO_2, "Martes");
dataset.setValue(61, SITIO_2, "Miércoles");
dataset.setValue(63, SITIO_2, "Jueves");
dataset.setValue(66, SITIO_2, "Viernes");
```

Una vez que tenemos los datos metidos en el modelo de datos, para obtener el gráfico con unas opciones por defecto, podemos llamar a la clase ChartFactory. En nuestro ejemplo lo haremos así

```
JFreeChart chart = ChartFactory.createBarChart3D("Visitas", "día",  
    "Número visitas", dataset, PlotOrientation.VERTICAL, true,  
    true, false);
```

Teniendo como resultado:



JavaMail

JavaMail se trata de una librería desarrollada por SUN encaminada al envío de correos electrónicos directamente desde una aplicación Java. El uso de ésta librería es muy sencillo pero detallaremos paso a paso como realizar la instalación y uso de ella.

Instalación:

La instalación de la librería para poder hacer uso de ella en nuestro proyecto se puede realizar de 2 maneras diferentes, o bien la importamos desde el entorno de

desarrollo que estemos usando (en mi caso “Eclipse”) o bien modificamos el CLASSPATH del sistema.

Clases y métodos básicos usados:

- **Clase Properties:** Ésta clase es la encargada de almacenar las propiedades de la conexión que vamos a establecer con el servidor de correo Saliente SMTP.
- **Método Put:** Mediante éste método asignaremos las propiedades que necesitamos, como son:
- **Servidor SMTP.**
- **Valor booleano de habilitación TLS.**
- **Puerto SMTP**
- **E-mail emisor del mensaje**
- **Usuario de acceso al servidor SMTP**
- **Valor booleano de contraseña requerida.**
- **Método Get:** Obtención de los parámetros anteriores ya guardados.
- **Clase Session:** Será la clase encargada de manejar la sesión de usuario
- **Método GetDefaultInstance:** e introduciremos la variable de la clase Properties que nos hemos creado anteriormente y ésta nos creará una sesión para dichas propiedades.
- **Método GetTransport:** Indicaremos a éste método el protocolo de transporte a utilizar (en nuestro caso smtp).
- **Clase MimeMessage:** Aquí formaremos el mensaje que deseamos enviar.
- **Constructor:** Se le introduce al constructor la sesión sobre la que vamos a enviar el mensaje
- **Método SetFrom:** Recibe como parámetro la dirección del emisor del mensaje de tipo InetAddress.
- **Método AddRecipient:** Recibe 2 parámetros, por un lado el tipo de receptor que vamos a especificar descritos en la clase Message.RecipientType (TO, CC, BCC). Como segundo parámetro le pasaremos igual que en el método

anterior, una variable de la clase `InternetAddress` con la dirección del receptor.

- **Método `SetSubject`:** Introducimos el asunto del mensaje como único parámetro en forma de `String`.
- **Método `SetText`:** De igual forma que en el método anterior, introducimos el texto del mensaje en forma de `String`.
- **Clase `Transport`:** Define los parámetros del protocolo de transporte. Para empezar, inicializaremos la variable obteniendo el tipo de protocolo de transporte de la clase `session` explicada anteriormente.
- **Método `Connect`:** Se encarga de establecer la conexión con el servidor, introduciendo el nombre de usuario y contraseña (si es requerida).
- **Método `SendMessage`:** Envía el mensaje que hemos creado anteriormente a los destinatarios especificados.
- **Método `Close`:** Cierra la conexión.

A continuación se presenta un caso de uso:

```
package com.autentia.training.javamail;

import java.util.Properties;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;

public class EmailSenderService {
    private final Properties properties = new Properties();

    private String password;

    private Session session;

    private void init() {
        properties.put("mail.smtp.host", "mail.gmail.com");
        properties.put("mail.smtp.starttls.enable", "true");
        properties.put("mail.smtp.port", 25);
        properties.put("mail.smtp.mail.sender", "emisor@gmail.com");
        properties.put("mail.smtp.user", "usuario");
        properties.put("mail.smtp.auth", "true");

        session = Session.getDefaultInstance(properties);
    }

    public void sendEmail(){
        init();
        try{
            MimeMessage message = new MimeMessage(session);
            message.setFrom(new InternetAddress((String)properties.get("mail.smtp.mail.sender")));
            message.addRecipient(Message.RecipientType.TO, new InternetAddress("receptor@gmail.com"));
            message.setSubject("Prueba");
            message.setText("Texto");
            Transport t = session.getTransport("smtp");
            t.connect((String)properties.get("mail.smtp.user"), "password");
            t.sendMessage(message, message.getAllRecipients());
            t.close();
        }catch (MessagingException me){
            //Aquí se debería o mostrar un mensaje de error o en lugar
            //de no hacer nada con la excepción, lanzarla para que el modulo
            //superior la capture y avise al usuario con un popup, por ejemplo.

            return;
        }
    }
}
```

Análisis de Resultados

Objetivos Alcanzados

- **Configuración de Datos del Banco**

Se cumplió el objetivo de que desde una pantalla diferente se pudieran configurar los datos de la sucursal del banco en donde estaría implementado y estos cambios pudieran observarse en la pantalla principal. Los posibles cambios que se pueden realizar son el nombre del banco, la cantidad de espacios disponibles y el logo del banco.

- **Hora del sistema**

Mediante un hilo, se logró implementar un reloj y un calendario que mostrarán la fecha y hora actual del sistema, el calendario y el reloj cambian en tiempo real.

- **Registro de personas**

Cada nueva persona que ingresa al banco podrá registrarse en una pantalla donde indicarán su nombre, su correo electrónico y marcarán cuál es su prioridad.

- **Envío de correo electrónico**

Luego de que una persona es registrada, se le enviará un primer correo en el que se mostrarán sus datos personales, en este el banco agradece la visita, muestra su logo y le indica que debe esperar su turno. Un segundo correo será enviado cuando al usuario le toca su turno.

- **Reportes:**

Dentro de los reportes realizados se encuentran:

- **Gráfico de Clientes atendidos:** En este gráfico se podrán observar la cantidad de clientes que han sido atendidos por el banco, según la fecha y la hora. Este gráfico se mostrará en forma de barras y pastel.
- **Gráfico de prioridades:** En este gráfico se mostrarán la clasificación de los clientes según la prioridad, (persona con discapacidad, adultos mayores, mujeres embarazadas, clientes corporativos y clientes regulares). Se podrán observar en forma de pastel y barras.
- **Lista de clientes:** En este reporte se mostrarán la lista de todos los clientes que ha recibido el banco, de cualquier prioridad, cualquier hora, cualquier día. Esta lista será mostrada en una tabla que contendrá los datos de la persona su prioridad, la fecha y la hora en que fue atendida.
- **Ordenamiento:** Para mayor facilidad de establecer un filtro para observar los clientes en determinado momento se logró que la lista de clientes pudiera ordenarse según la hora, fecha, nombre y prioridad.
- **Manejo de prioridades:** Mediante colas de prioridad, se le otorgarán los campos a esas personas que según su categoría deberán atenderse primero.

Objetivos no alcanzados

Una de las metas que el equipo deseaba lograr era que en la interfaz hubiera movimiento, simulando la fila de personas moviéndose a las cajas, por falta de tiempo no fue posible.

Manual de Usuario

Este manual servirá como herramienta para aprender a utilizar todas las funcionalidades del asistente de atención a clientes del BAC San José. En su contenido podremos observar los aspectos más esenciales para poder utilizar este programa de una manera más simple.

A continuación se presentan las instrucciones de uso:

Vista General

Sistema de atención al cliente BAC San José

Reportes

Bienvenido al Banco BAC San José

Configurar logo del banco

Cantidad de cajas
11

Cajas Disponibles
11

Fecha y Hora del sistema
13 / 10 / 2014
23:48:13

Cajas Ocupadas
0
Liberar una caja

Personas en cola: 0 Atender

Siguiete:

Tiquete:

Nueva Persona

En la pantalla principal del programa se encontrarán los siguientes elementos

- Una imagen simulando al guarda de seguridad que es el que va a usar el sistema.
- Un espacio y un botón para asignarle el logo al banco.
- Un espacio que indica la cantidad de cajas que posee la sucursal.
- Un espacio donde se indica la cantidad de cajas.
- Un espacio que indica la cantidad de cajas ocupadas
- Un botón que se encarga de liberar las cajas
- Un botón “Nueva Persona”, que es el que abre la pantalla para cuando un nuevo cliente que llego al banco, al registrar a la persona se le otorga un ticket.
- Inmediatamente después de que el cliente se registra entra en el espacio que se indica de “Personas e cola”
- Botón atender, este saca a un persona de la cola para que pueda ser atendida.
- En el espacio que indica siguiente, puesta el nombre de la persona que está siguiente en la cola.
- El espacio del ticket indicará el número del ticket que está en espera.

Configuración de datos del banco

En el menú de la pantalla principal se puede observar un icono de configuración, al presionarlo se mostrará una pantalla en donde se debe colocar el nombre del banco y la cantidad de cajas que tiene la sucursal de ese banco.

Design Preview [VentanaDeConfiguracion]

Configuración de datos

Nombre del Banco:

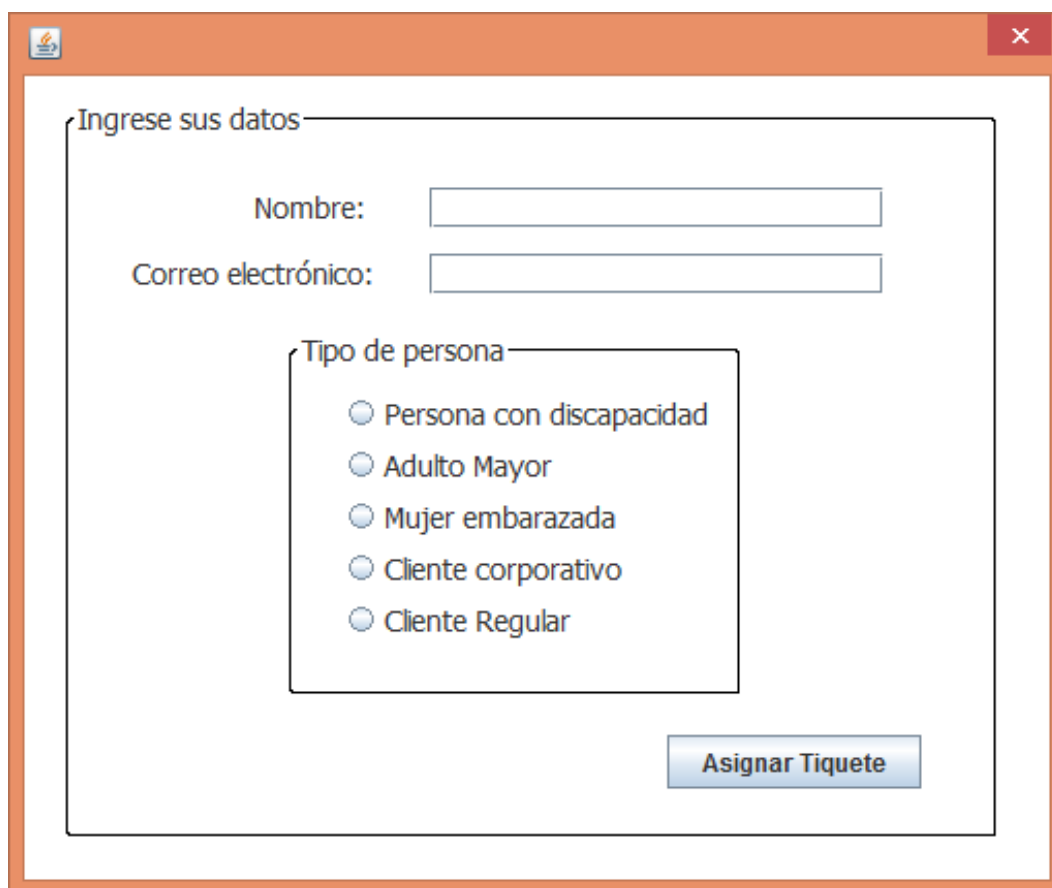
Cantidad de cajas disponibles:

Aceptar

Al presionar el botón Aceptar los datos del banco serán registrados y modificados internamente y en la pantalla principal, las cajas disponibles que se colocaron en esta ventana serán la base para manejar la atención de los clientes, si al siguiente día o en algún momento determinado se desea cambiar la cantidad de cajas disponibles, se debe entrar nuevamente a esta ventana y colocar las cajas que desea que estén disponibles, el nombre no es necesario cambiarlo pues es el que ya está registrado. De lo contrario si se desea reestablecer la configuración se debe entrar a la carpeta de documentos de la computadora y eliminar la carpeta que es generada por este programa.

Registro de clientes

En el botón registrar persona se desplegará una nueva pantalla en donde se registrarán los datos de la persona, el nombre completo, el correo electrónico y el tipo de cliente que es, esto para efecto de saber cuál es su prioridad.



The image shows a software window with an orange title bar and a close button (X) in the top right corner. The main content area is titled "Ingresa sus datos" and contains the following elements:

- A label "Nombre:" followed by a text input field.
- A label "Correo electrónico:" followed by a text input field.
- A section titled "Tipo de persona" containing five radio button options:
 - ☐ Persona con discapacidad
 - ☐ Adulto Mayor
 - ☐ Mujer embarazada
 - ☐ Cliente corporativo
 - ☐ Cliente Regular
- A blue button labeled "Asignar Tiquete" located at the bottom right of the form area.

Correo electrónico

Al presionar el botón asignar ticket, internamente se ha enviado un correo electrónico a la dirección que el cliente anteriormente proporciono, con los datos que este recién ingresó, y el ticket que a este le corresponde según su prioridad, además del agradecimiento por ingresar al banco y el logo del banco.

Asignación de tickets

La asignación de un ticket para un cliente corresponderá al nivel de prioridad que este tenga, recordando que para este programa se están manejando prioridades que se clasifican en:

1. Personas con discapacidad
2. Adultos Mayores
3. Mujeres Embarazadas
4. Clientes corporativos
5. Clientes Regulares

Atención a los clientes

En la pantalla principal se pueden observar tres espacios específicos en la parte posterior derecha.

“Personas en cola”: Indica la cantidad de clientes que se encuentran en el la cola.

“Siguiente”: Este botón permitirá que esa persona sea atendida en una de las cajas disponibles, sumando uno a las cajas ocupadas.

“Ticket”: Indica el ticket de la persona que está de primero en la cola.

Reportes

Para tener acceso a los reportes, se deben seleccionar del menú principal la opción que indica reportes, esta desplegará tres opciones que serán analizadas a continuación.

Gráfico de clientes atendidos

En la pantalla se muestran los datos recolectados de todos los clientes que han sido atendidos en el banco, contra los que aún están en cola. Este gráfico se presenta en dos versiones, gráfico pastel y gráfico de barras. También contiene un filtro para buscar esa misma información pero en una fecha establecida.

Gráfico de prioridad

En esta opción se podrá observar la clasificación de todos los clientes que han ingresado al banco, según su prioridad. Este también puede observarse en dos representaciones; gráfico de barras y gráfico tipo pastel. Este contiene un filtro en el que se podrá observar la información como un historial, de la clasificación de los clientes del banco según su prioridad.

Lista de clientes

Esta lista contiene todos y cada uno de los clientes que han sido registrados en el banco, en esta lista se podrá observar el nombre del cliente, el correo electrónico, la fecha en la que ingreso al banco, y la hora en la que fue atendido.

Esta lista es capaz de ofrecer los datos en forma ordenada según fecha, hora de atención y nombre del cliente, para esto nada más se debe presionar el sobre las diferentes columnas, para obtener dicho ordenamiento.

Ejecución de reproductor en el terminal de Linux

Para poder ejecutar este programa desde la plataforma de Linux se deben seguir los siguientes pasos:

1. Dirigirse a línea de comandos (terminal) y buscar la respectiva carpeta donde se encuentra el programa.
2. En la línea de comandos y dentro de la carpeta colocar la palabra "ant".
3. Entrar a la carpeta "dist".
4. Escribir `java -jar main`.

Conclusiones

Al finalizar este proyecto, logramos comprender de manera más profunda la conceptualización de las estructuras de datos por medio de colas. Entendimos el concepto de colas de prioridad al aplicarlo en este proyecto.

La utilización del repositorio de Github nos deja una gran experiencia, de cómo trabajar de forma más ordenada y llevando un control continuo de las actualizaciones y cambios que se realizan a todos los archivos.

La herramienta NetBeans utilizada para la implementación de la interface nos proporcionó nuevos conocimientos sobre el lenguaje de programación y de su largo alcance.

Aparte de NetBeans se utilizó el editor de texto SublimeText2, que si bien es un poco más complicado, también sumo su cuota a nuestro aprendizaje sobre java. En muchas ocasiones surgieron problemas por falta de librerías y recursos desconocidos que no podíamos implementar directamente desde SublimeText pero gracias a que utilizamos NetBeans y este provee un fácil acceso a estas, se hizo posible.

La nueva experiencia de enviar correos electrónicos desde java fue sencillo y gratificante el ver que los correos se enviaban de forma real, al igual que la implementación de los gráficos, esta es una funcionalidad muy útil para cualquier contexto y fácil de utilizar he implementar.