Instrucciones Programa Hotel Gran Dpoo.

Juan Leonardo Rangel

Catalina Muñoz

Karen Fuentes

1. La primera interfaz que se desplegará es la de la ventana principal del hotel. Nuestro propósito es que se pueda ver sin iniciar sesión lo que ofrece el establecimiento, su restaurante, servicios, y su estadística de ocupación.
2. En el botón de iniciar sesión será llevado a una ventana de LogIn. Contamos con 3 usuarios predeterminados de cada tipo para iniciar sesión. “ADMIN”, “RES”, “PER”, todos con “12345” como clave. Adicionalmente podrá registrarse, pero ojo, en esta parte solo se pueden agregar empleados del hotel, no administradores.
3. Si inicia como administrador, entrara a una ventana especializada para ese tipo de usuario. En esta tendrá opciones relacionadas al management del hotel, como agregar habitaciones, cargar tarifas, administrar reservas y agregar personal (desde acá si se pueden agregar admins).
4. Si inicia como recepcionista tendrá acceso a lo relacionado con la facturación, reservas y pagos de los clientes.
5. Si inicia como empleado tendrá acceso a las opciones relacionadas con el personal del hotel, como agregar servicios y generar facturación.
6. Como lo habrá notado, moverse dentro de nuestro programa es muy sencillo, contamos con un menú lateral con el que están accesibles todas las funciones de cada una de las ventadas de los distintos tipos de usuarios. Adicionalmente tienen un botón de salir, que lo llevara de nuevo a la ventana iniciar con los datos del hotel. Si desea volver a ingresar a las ventanas de los usuarios, usted deberá iniciar sesión.
7. Como también habrá notado, no todas las funcionalidades están disponibles en el momento, pero este seguro de que sabemos como se pone, pero hubo problemas en el desarrollo ☹. La persistencia no está colocada por que se supone que funcionaba con Serializable, no funciono, así que por recomendación de Mario intentamos con apache commons pero las librerías por alguna razón no funciona en nuestros Eclipse’s. Espero lo entienda. Adjunto la clase “modelo” de nuestra persistencia (En el repositorio aún está la de Serializable):

package posterityKeeper;

import java.io.FileWriter;

import java.io.IOException;

import java.util.ArrayList;

import org.apache.commons.csv.CSVFormat;

import org.apache.commons.csv.CSVPrinter;

public class ArchivadorCsv {

public void guardarHabitacionesCSV(ArrayList<Habitacion> habitaciones) {

try (FileWriter writer = new FileWriter(RUTA\_HABITA);

CSVPrinter csvPrinter = new CSVPrinter(writer, CSVFormat.DEFAULT.withHeader("numero", "tipo", "precio", "disponible").withIgnoreHeaderCase().withTrim());) {

for (Habitacion habitacion : habitaciones) {

csvPrinter.printRecord(habitacion.getNumero(), habitacion.getTipo(), habitacion.getPrecio(), habitacion.isDisponible());

}

csvPrinter.flush();

} catch (IOException e) {

System.out.println("Error writing CSV file: " + e.getMessage());

}

}

public ArrayList<Habitacion> cargarHabitaciones() {

try (FileReader reader = new FileReader(RUTA\_HABITA);

CSVParser csvParser = new CSVParser(reader, CSVFormat.DEFAULT.withHeader("numero", "tipo", "precio", "disponible").withIgnoreHeaderCase().withTrim());) {

ArrayList<Habitacion> habitaciones = new ArrayList<Habitacion>();

for (CSVRecord record : csvParser) {

int numero = Integer.parseInt(record.get("numero"));

String tipo = record.get("tipo");

double precio = Double.parseDouble(record.get("precio"));

boolean disponible = Boolean.parseBoolean(record.get("disponible"));

Habitacion habitacion = new Habitacion(numero, tipo, precio, disponible);

habitaciones.add(habitacion);

}

return habitaciones;

} catch (IOException e) {

System.out.println("Error reading CSV file: " + e.getMessage());

}

return new ArrayList<Habitacion>();

}

}

1. Gracias, atentamente, Catalina, Leo, y Karen.