Departamento de TIC
Algoritmos y Programación II
Proyecto Final

Danna Sofía García Trujillo
Andrea Núñez Rodríguez

Objetivo.

Este proyecto tiene como finalidad modelar un proyecto el cual refleje los aprendizajes obtenidos en el curso de Algoritmos y Programación II, a partir de un aplicativo que realizarán los estudiantes encargados del proyecto, quienes plasmarán en este la planificación del mismo y las decisiones tomadas para modelarlo de una manera estructurada y organizada cumpliendo con los objetivos del curso.

Preparativos.

- Para este proyecto se utilizará git como plataforma para manejar el versionamiento del trabajo desarrollado localmente y se manejará **GitHub** como repositorio con visibilidad **privada.**

Enunciado

Un supermercado es un establecimiento que tiene como principal finalidad acercar a los consumidores una importante variedad de productos de diversas marcas, precios y estilos. Como en tantos otros establecimientos de venta, en los supermercados es esencial disponer de un software que ayude a centralizar la gestión de este**,** las ventas, los clientes, el personal y otros establecimientos que se encuentren dentro de este y afecten su contabilidad, todo en pro de optimizar la gestión del supermercado.

Es de esta manera, como dos estudiantes de la Universidad Icesi, tienen como proyecto crear un programa que simule el manejo de la gestión de un supermercado, desde sus elementos mínimos, que refleje el funcionamiento de este y sea agradable con quien lo maneje. Para esto, realizaron una investigación de los componentes que hacen parte de un supermercado, tales como los productos, mercancía importada y exportada, los clientes del supermercado, los empleados y uno de los establecimientos que encontraron muy importante dentro de cada

supermercado fue una farmacia, que a pesar de que esté dentro del mismo, contaba también con un funcionamiento más bien independiente al del supermercado, sin embargo, ambos contaban con aspectos similares en cuanto a funcionalidad.

Como principal objetivo, el programa estará direccionado a gestionar, en primera instancia, la información de los clientes del supermercado y de la farmacia, siendo estos últimos un subconjunto del conjunto universal de clientes. De estos clientes se desea conocer el nombre, el apellido, el código y el correo; de esta manera se podrá acceder fácilmente a una base de datos con la información de cada uno. En segundo lugar, se desea conocer quienes son los empleados que están actualmente en el supermercado, que son aquellos que gestionan la información de los productos y demás. Un aspecto a tener en cuenta es que los empleados van a estar organizados en una estructura jerárquica que va a tener en cuenta su cargo; de esta manera, quien esté manejando el software administrativo, estará en la capacidad de agregar un empleado, buscar a un empleado, despedir a un empleado, agregar un cargo o eliminar un cargo existente. Este mismo usuario también podrá administrar la materia prima del supermercado, es decir, la mercancía. Para esto, se necesita que los productos estén enumerados por códigos y de estos se desea conocer su nombre, su precio y la ubicación del producto dentro del supermercado.

Uno de los beneficios que recibirán aquellos que estén registrados en el supermercado, es que recibirán promociones especiales que serán establecidas por la persona que esté manejando el software en ese momento.
Como tercera instancia, el supermercado tendrá un acceso directo a la información de la farmacia, la cual estará compuesta por el subconjunto de clientes registrados a esta, y debido a que los productos farmacéuticos son de mayor cuidado, se llevará un control de la mercancía que entra a la farmacia a partir de un inventario.
En este inventario se especificará el número de productos de los cuales se desea revisar la información. Esta lista de productos puede estar ordenada por nombre del medicamento, su fecha de llegada a la farmacia, el horario de llegada del producto, la marca del producto, el tipo o el código.

Como los clientes de la farmacia son clientes especiales dentro del establecimiento, de ellos se tendrá información más especifica como lo es su cumpleaños y una foto de este para poder identificarlos más fácilmente.
El software contará con elementos dinámicos que sean amenos a la vista de quien lo maneja y le dará datos como la hora local, desde que entra hasta que sale del aplicativo, también podrá interactuar con algunos elementos que se encuentran dentro de la interfaz proporcionándole al mismo un efecto visual llamativo y armonioso a la vista.

Requerimientos funcionales:

Nombre:	R. #1 Registrar un cliente
Resumen:	El programa recibe los datos del cliente y lo registra en el árbol binario
Entradas:	
Resultados:	Cliente registrado con éxito

Nombre:	R. #2. Eliminar cliente
Resumen:	El programa recibe el código del cliente que se desea eliminar y procede a eliminarlo del árbol binario
Entradas:	
Resultados:	Cliente eliminado con éxito

Nombre:	R. #3. Buscar cliente
Resumen:	El programa recibe el código del cliente que se desea buscar y procede a hacer la búsqueda. Si lo encuentra muestra la información de este cliente.
Entradas:	
Resultados:	Cliente buscado con éxito, información mostrada en pantalla
	Cliente no encontrado

Nombre:	R. #4. Cargar información de los productos de un archivo plano
Resumen:	El programa carga la información de los productos desde un archivo plano
Entradas:	
Resultados:	Información cargada con éxito

Nombre:	R. #5. Exportar información de los productos en un archivo plano
Resumen:	El programa exporta la información de los productos en un archivo plano
Entradas:	
Resultados:	La esfera rebota con éxito

Nombre:	R. #6. Insertar producto nuevo
Resumen:	El programa recibe la información del producto nuevo y lo agrega a una lista doblemente enlazada
Entradas:	
Resultados:	Producto insertado con éxito

Nombre:	R. #7. Eliminar producto
Resumen:	El programa recibe la información del producto que se desea eliminar y procede a quitarlo de la lista doblemente enlazada
Entradas:	
Resultados:	Producto eliminado con éxito

Nombre:	R. #8. Buscar producto
Resumen:	El programa recibe el código del producto que se desea buscar y procede a realizar la búsqueda en la lista doblemente enlazada
Entradas:	
Resultados:	Producto encontrado con éxito
	El producto no existe

Nombre:	R. #9. Modificar nombre producto

Resumen:	El programa recibe el código del producto cuyo nombre se desea modificar y procede a cambiarlo con el nombre nuevo que ingresa el usuario
Entradas:	
Resultados:	Nombre cambiado con éxito
	Producto no encontrado

Nombre:	R. #10. Modificar código producto
Resumen:	El programa recibe el código del producto cuyo código se desea modificar y procede a cambiarlo con el código nuevo que ingresa el usuario
Entradas:	
Resultados:	Código cambiado con éxito
	Producto no encontrado

Nombre:	R. #11. Modificar precio del producto
Resumen:	El programa recibe el código del producto cuyo precio se desea modificar y procede a cambiarlo con el precio nuevo que ingresa el usuario
Entradas:	
Resultados:	Precio cambiado con éxito
	Producto no encontrado

Nombre:	R. #12. Modificar ubicación del producto
Resumen:	El programa recibe el código del producto cuya ubicación se desea modificar y procede a cambiarlo con la nueva ubicación que ingresa el usuario
Entradas:	
Resultados:	Ubicación cambiada con éxito
	Producto no encontrado

Nombre:	R. #13. Serializar información
Resumen:	El programa persiste la información por medio de serialización en un archivo txt
Entradas:	
Resultados:	Serialización realizada con éxito

Nombre:	R. #13. Deserializacion información
Resumen:	El programa carga la información por medio de deserializacion de un archivo txt
Entradas:	
Resultados:	Deserialización realizada con éxito

Nombre:	R. #14. Contratar empleado
Resumen:	El programa recibe la información de un empelado que se desea agregar a un cargo en especifico
Entradas:	Información del empleado a agregar
Resultados:	Empleado contratado con éxito

Nombre:	R. #15. Buscar empleado
Resumen:	El programa recibe el código del empleado que desea ser buscado
Entradas:	Código del empleado a buscar
Resultados:	Empleado encontrado con éxito

Nombre:	R. #16. Despedir empleado
Resumen:	El programa recibe el código del empleado que desea ser despedido
Entradas:	Código del empleado a despedir
Resultados:	Empleado despedido con éxito

Nombre:	R. #17. Agregar cargo
Resumen:	El programa recibe la información del cargo que se desea agregar
Entradas:	Nombre del cargo a agregar con sus respectivas características.
Resultados:	Cargo agregado con éxito

Nombre:	R. #18. Eliminar cargo
Resumen:	El programa recibe la información del cargo que se desea eliminar
Entradas:	Nombre del cargo a eliminar
Resultados:	Cargo eliminado con éxito

Requerimientos No Funcionales

RFN1: En el proceso de búsqueda de los elementos, las operaciones a realizar deben ser realizadas por los métodos implementados por la aplicación.

RNF2: La aplicación debe seguir los principios de usabilidad y diseño, por lo tanto, debe ser intuitivo y agradable con el usuario.

RNF3: La aplicación debe ser inicializada inmediatamente apenas se ejecute.

RNF4: Todas las excepciones, ya sean de tipo de Runtime o errores propios del programa serán manejados por la propia aplicación, permitiendo de esta manera que el proceso de ejecución no termine por estos errores.

RNF5: La aplicación debe ser persistente con la información que el usuario ingrese; esto quiere decir, que una vez ingresados ciertos datos, no importa si el programa es cerrado, los datos van a seguir en la aplicación.

Pruebas unitarias

Nombre	Clase	Escenario
setUpEscenary1	FarmaciaTest	Un objeto de la clase Farmacia y un objeto de la clase ClienteFarmacia con id=1 , nombre=Danna , apellido=garcia , email=garcia@gmail.com , genero= female, país=Colombia , foto=null ,cumpleanios=march ,
Setup2	FarmaciaTest	Un objeto de la clase Farmacia
Setup3	FarmaciaTest	Un objeto de la clase Farmacia y cuatro objetos de la clase ClienteFarmacia. El primero con con id=0 , nombre=Danna , apellido=garcia , email=garcia@gmail.com , genero= female, país=Colombia , foto=null ,cumpleanios=march. El segundo con con id=1 , nombre=Danna , apellido=garcia , email=garcia@gmail.com , genero= female, país=Colombia , foto=null ,cumpleanios=march. El tercero con con id=2 , nombre=Danna , apellido=garcia , email=garcia@gmail.com , genero= female, país=Colombia , foto=null ,cumpleanios=march. El cuarto con con id=3, nombre=Danna , apellido=garcia , email=garcia@gmail.com , genero= female, país=Colombia , foto=null ,cumpleanios=march.
setUpScenary2	FarmaciaTest	Un entero que representa la cantidad de productos de la farmacia
setUpScenary3	FarmaciaTest	Un objeto de la clase Farmacia con 10 productos

Clase	Método	Escenario	Valores de entrada	Resultado
Farmacia	addParticipantIntoTree	setUpEscenary1	1,"Danna","garcia","garcia@gmail.com", "female","Colombia",**null**,"march"	El primer objeto del árbol fue agregado
Farmacia	buscarClientesEspeciales	setUpEscenary1	1,"Danna","garcia","garcia@gmail.com", "female","Colombia",**null**,"march"	El cliente especial es el que fue agregado
Farmacia	generarInformacionDeLos ProductosFarmacia	setUpScenary2	10	Farmacia creada con diez productos
Farmacia	OrdenamientoPorTipo	setUpScenary3	10	Los productos fueron ordenados por tipo
Farmacia	OrdenamientoPorNombre DeLaMarca	setUpScenary3	10	Los productos fueron ordenados por marca
Farmacia	OrdenamientoPorNombre DelProducto	setUpScenary3	10	Los productos fueron ordenados por producto
Farmacia	OrdenamientoPorCodigo	setUpScenary3	10	Los productos fueron ordenados por código

Farmacia	OrdenamientoPorFechaDe Llegada	setUpScenary3	10	Los productos fueron ordenados por fecha de llegada
Farmacia	OrdenamientoPorTiempo	setUpScenary3	10	Los productos fueron ordenados por tiempo

---

Nombre	Clase	Escenario
setUpEscenario1	CargoSupermercadoTest	Un objeto de la clase CargoSupermercado con un cargo llamado cargo1
setUpEscenario2	CargoSupermercadoTest	Un objeto de la clase CargoSupermercado con 20 cargos

Clase	Método	Escenario	Valores de entrada	Resultado
CargoSupermercado	AgregarCargo	setUpEscenario1	1 cargo y se agregan 20 más	El supermercado tiene ahora 21 cargos.
CargoSupermercado	buscarCargo	setUpEscenario2	20 cargos	Busca cargos existentes y no existentes
CargoSupermercado	Contratar	setUpEscenario2	20 cargos	Le asigna un cargo a cada empleado
CargoSupermercado	buscarEmpleado	setUpEscenario2	20 cargos	Verifica que los empleados existentes se encuentren
CargoSupermercado	buscarJefe	setUpEscenario2	20 cargos	Verifica que los cargos jefes existentes se encuentren

---

Nombre	Clase	Escenario
setUpEscenario1	ClienteSupermercadoTest	Un objeto de la clase ClienteSupermercado con entradas: nombre = "Marta"; direccion="dirección"; telefono = "1234567"; email:marta@email.com
setUpEscenario2	ClienteSupermercadoTest	Dos objetos de la clase ClienteSupermercado con entradas: nombre = "Marta"; direccion="dirección";

		telefono = "1234567"; email:marta@email.com nombre = "Alberto"; direccion="dirección"; telefono = "1234567"; email:alberto@email.com
setUpEscenario3	ClienteSupermercadoTest	Tres objetos de la clase ClienteSupermercado con entradas nombre = "Marta"; direccion="dirección"; telefono = "1234567"; email:marta@email.com nombre = "Alberto"; direccion="dirección"; telefono = "1234567"; email:alberto@email.com nombre = "Paco"; direccion="dirección"; telefono = "1234567"; email:paco@email.com
setUpEscenario3	ClienteSupermercadoTest	Cinco objetos de la clase ClienteSupermercado con entadas nombre = "Marta"; direccion="dirección"; telefono = "1234567"; email:marta@email.com nombre = "Alberto"; direccion="dirección"; telefono = "1234567"; email:alberto@email.com nombre = "Paco"; direccion="dirección"; telefono = "1234567"; email:paco@email.com nombre = "Luis"; direccion="dirección";

		telefono = "1234567"; email:luis@email.com nombre = "Xiomara"; direccion="dirección"; telefono = "1234567"; email:xiomara@email.com

Clase	Método	Escenario	Valores de entrada	Resultado
ClienteSuperemercado	insertar	setUpEscenario4	nombre = "Marta"; direccion="dirección"; telefono = "1234567"; email:nombre@email.com	El supermercado sigue teniendo 5 clientes ya que al insertar un cliente ya existente se genera error.
ClienteSuperemercado	buscar	setUpEscenario4	"Petronila", "Marta", "Xiomara"	Busca clientes correctamente, y los que no existen indica que no existe

---

Nombre	Clase	Escenario
setUpEscenario1	SuperMercadoTest	Un objeto de la clase CargoSupermercado con un 15 cargos
setUpEscenario2	CargoSupermercadoTest	Un objeto de la clase CargoSupermercado con un 15 cargos y 6 personas contratadas

Clase	Método	Escenario	Valores de entrada	Resultado
Supermercado	crearCargo	setUpEscenario1	3 cargos con entradas: "Vendedor puerta a puerta", "Gerente General"; "Vendedor puerta a puerta2", "Vendedor 4";	El supermercado tiene ahora 18 cargos.

			"Vendedor puerta a puerta3", "Contador2".	
Supermercado	eliminarCargo	setUpEscenario1	3 cargos con entradas: “Contador3”; “contador2; “contador1”.	El supermercado elimina los cargos y queda con 15 cargos en total
Supermercado	buscarEmpleado	setUpEscenario2	idPersona=560	El empleado ha sido encontrado

NOTA: Los requerimientos funcionales que actualmente funcionan son el requerimiento funcional número 1 y el requerimiento funcional número 2.