PROYECTO 2 – SISTEMA DE ATENCION A CLIENTES

202000562 – Pedro Luis Pu Tavico

Resumen

La empresa “Soluciones Guatemaltecas, S.A.” está construyendo un sistema de atención a clientes diseñado para cualquier organización que necesite brindar servicios presenciales a sus clientes.
“Soluciones Guatemaltecas, S.A” está innovando la experiencia del cliente y es por ello que las empresas que se suscriban al servicio de atención podrán llevar el control de todos sus puntos de atención y de sus escritorios de una manera sencilla y ordenada.

Para esta aplicación se hará uso de un algoritmo que permita a las empresas llevar un control ordenado de todos lo clientes que han sido atendidos en cada uno de sus puntos de atención y los tiempos mínimos, máximos y promedios tanto los de espera como los de atención.

Palabras clave

TDA: es un conjunto de datos u objetos creado de manera personalizada por un programador para un fin específico.

Nodo: es un registro que contiene un dato de interés y al menos un puntero para referenciar (apuntar) a otro nodo.

POO: La Programación Orientada a Objetos es un paradigma de programación que parte del concepto de "objetos" como base.

Apuntador: es una variable que contiene la dirección de memoria de otra variable.

Abstract

The company "Soluciones Guatemaltecas, S.A." is building a customer service system designed for any organization that needs to provide face-to-face services to its customers.
"Soluciones Guatemaltecas, S.A." is innovating the customer experience and that is why companies that subscribe to the customer service will be able to keep track of all their customer service points and desks in a simple and orderly manner.

This application will make use of an algorithm that allows companies to keep an orderly control of all customers who have been served in each of its points of care and the minimum, maximum and average times both waiting and attention.

Keywords

TDA: is a set of data or objects created in a personalized way by a programmer for a specific purpose. A TDA is an abstraction that allows modeling the characteristics of a particular element.

Node: is a record that contains a data of interest and at least one pointer to reference (point) to another node.

OOP: Object Oriented Programming is a programming paradigm that starts from the concept of "objects" as a base.

Pointer: is a variable that contains the memory address of another variable.

Introducción

Esta aplicación software consiste en brindarle un servicio al usuario, donde el usuario suscribirá su empresa al servicio y obtendrá el control de cada uno de sus puntos de atención y también los escritorios que se encuentran en cada punto. La empresa suscrita podrá activar y desactivar escritorios, como también atender a clientes conforme se vayan incorporando en la cola. También podrá visualizar tiempos y cantidad de clientes en espera de atención y atendidos en un punto de atención seleccionado.

TDA

Un Tipo de dato abstracto (en adelante TDA) es un conjunto de datos u objetos al cual se le asocian operaciones. El TDA provee de una interfaz con la cual es posible realizar las operaciones permitidas, abstrayéndose de la manera en como están implementadas dichas operaciones. Esto quiere decir que un mismo TDA puede ser implementado utilizando distintas estructuras de datos y proveer la misma funcionalidad.

El paradigma de orientación a objetos permite el encapsulamiento de los datos y las operaciones mediante la definición de clases e interfaces, lo cual permite ocultar la manera en cómo ha sido implementado el TDA y solo permite el acceso a los datos a través de las operaciones provistas por la interfaz.

Memoria dinámica

Se refiere a aquella memoria que no puede ser definida ya que no se conoce o no se tiene idea del número de la variable a considerarse, la solución a este problema es la memoria dinámica que permite solicitar memoria en tiempo de ejecución, por lo que cuanta más memoria se necesite, más se solicita al sistema operativo. El sistema operativo maneja la memoria gracias al uso de punteros, por la misma naturaleza del proceso nos impide conocer el tamaño de la memoria necesaria en el momento de compilar.

Un dato importante es que como tal este tipo de datos se crean y se destruyen mientras se ejecuta el programa y por lo tanto la estructura de datos se va dimensionando de forma precisa a los requerimientos del programa, evitándonos así perder datos o desperdiciar memoria si hubiéramos tratado de definir la cantidad de memoria a utilizar en el momento de compilar el programa.

XML

XML es el acrónimo de Extensible Markup Language, es decir, es un lenguaje de marcado que define un conjunto de reglas para la codificación de documentos.

El lenguaje de marcado es un conjunto de códigos que se pueden aplicar en el análisis de datos o la lectura de textos creados por computadoras o personas. El lenguaje XML proporciona una plataforma para definir elementos para crear un formato y generar un lenguaje personalizado.

Un archivo XML se divide en dos partes: prolog y body. La parte prolog consiste en metadatos administrativos, como declaración XML, instrucción de procesamiento opcional, declaración de tipo de documento y comentarios. La parte del body se compone de dos partes: estructural y de contenido (presente en los textos simples).

El diseño XML se centra en la simplicidad, la generalidad y la facilidad de uso y, por lo tanto, se utiliza para varios servicios web. Tanto es así que hay sistemas destinados a ayudar en la definición de lenguajes basados en XML, así como APIs que ayudan en el procesamiento de datos XML - que no deben confundirse con HTML.

Listas Simple Enlazadas

Una lista enlazada es una de las estructuras de datos fundamentales, y puede ser usada para implementar otras estructuras de datos. Consiste en una secuencia de nodos, en los que se guardan campos de datos arbitrarios y una o dos referencias, enlaces o punteros al nodo anterior o posterior. El principal beneficio de las listas enlazadas respecto a los vectores convencionales es que el orden de los elementos enlazados puede ser diferente al orden de almacenamiento en la memoria o el disco, permitiendo que el orden de recorrido de la lista sea diferente al de almacenamiento.

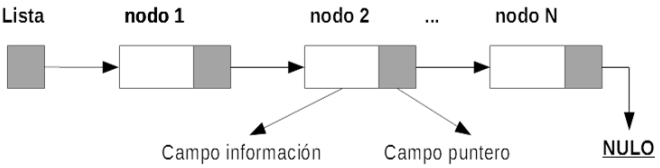


Figura 1. Ejemplo de lista simple enlazada.
Fuente: elaboración propia,

Listas Doblemente Enlazadas

Es una estructura de datos que consiste en un conjunto de nodos enlazados secuencialmente. Cada nodo contiene tres campos, dos para los llamados enlaces, que son referencias al nodo siguiente y al anterior en la secuencia de nodos, y otro más para el almacenamiento de la información (en este caso un entero). El enlace al nodo anterior del primer nodo y el enlace al nodo siguiente del último nodo, apuntan a un tipo de nodo que marca el final de la lista, normalmente un nodo centinela o puntero null, para facilitar el recorrido de la lista. Si existe un único nodo centinela, entonces la lista es circular a través del nodo centinela.

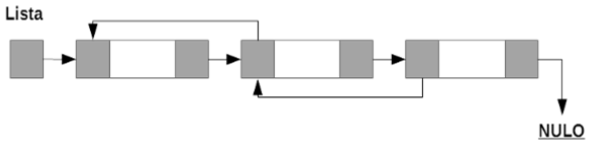


Figura 2. Ejemplo de lista doble enlazada.
Fuente: elaboración propia,

El doble enlace de los nodos permite recorrer la lista en cualquier dirección. Mientras que agregar o eliminar un nodo en una lista doblemente enlazada requiere cambiar más enlaces que en estas mismas operaciones en una lista enlazada simple, las operaciones son más simples porque no hay necesidad de mantener guardado el nodo anterior durante el recorrido, ni necesidad de recorrer la lista para hallar el nodo anterior, la referencia al nodo que se quiere eliminar o insertar es lo único necesario.

Lista Enlazada Empresas

Esta clase almacena todos los datos como id, nombre, abreviatura, puntos de atención y transacciones que le pertenecen a la empresa a agregar a la clase. Debido a que es una clase de tipo lista enlazada simple nos permite almacenar la cantidad de empresas que necesarias sin tener que ingresar la cantidad de empresas a almacenar ya que se irán guardando dinámicamente.

Lista Enlazada Transacciones

En esta clase se almacenarán todos transacciones que le pertenecen a una empresa. La clase será dada como parámetro a la clase empresa a la que pertenezca y esta clase es de tipo lista enlazada simple ya que las transacciones que puede llegar a tener una empresa pueden ser varias.

Lista Enlazada Puntos de Atención

En esta clase se almacenarán todos los puntos de atención que le pertenecen a una empresa. La clase será dada como parámetro a la clase empresa a la que pertenezca.

Lista Enlazada Escritorios

Esta clase contendrá todos los escritorios que le pertenecen a un solo punto de atención. Los escritorios almacenados contendrán información

como, id, identificación del escritorio, nombre del encargado, estado en el que se encuentra el escritorio y los clientes que han sido atendidos por ese escritorio.

Pilas

Una pila representa una estructura lineal de datos en que se puede agregar o quitar elementos únicamente por uno de los dos extremos. En consecuencia, los elementos de una pila se eliminan en el orden inverso al que se insertaron. Debido a esta característica, se le conoce como estructura LIFO (last input, first output).
En otras palabras, Una pila es una lista ordinal o estructura de datos en la que el modo de acceso a sus elementos es de tipo LIFO que permite almacenar y recuperar datos.

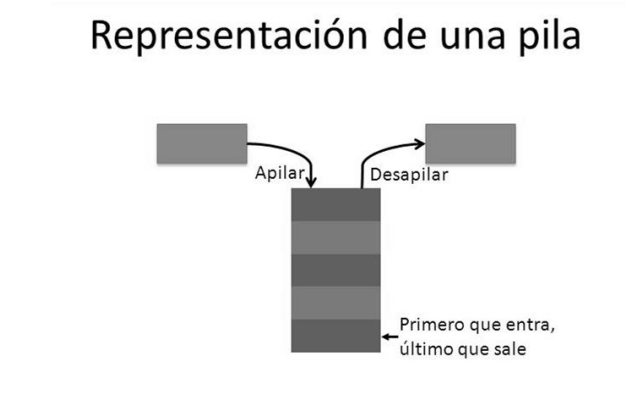


Figura 3. Representación de una pila.
Fuente: elaboración propia,

Pila Escritorios Activos e Inactivos

Para determinar si un escritorio en un determinado punto de atención se encuentra activo se implementó el uso de pilas, funcionando de la siguiente manera:
Primeo se verifica el estado del escritorio, si el escritorio se encuentra en estado inactivo se agrega a la pila y si se encuentra activo se elimina de la pila dejando así solo los escritorios activos dentro de la pila. Representado la pila así, todos los escritorios activos que se encuentren dentro de ella.

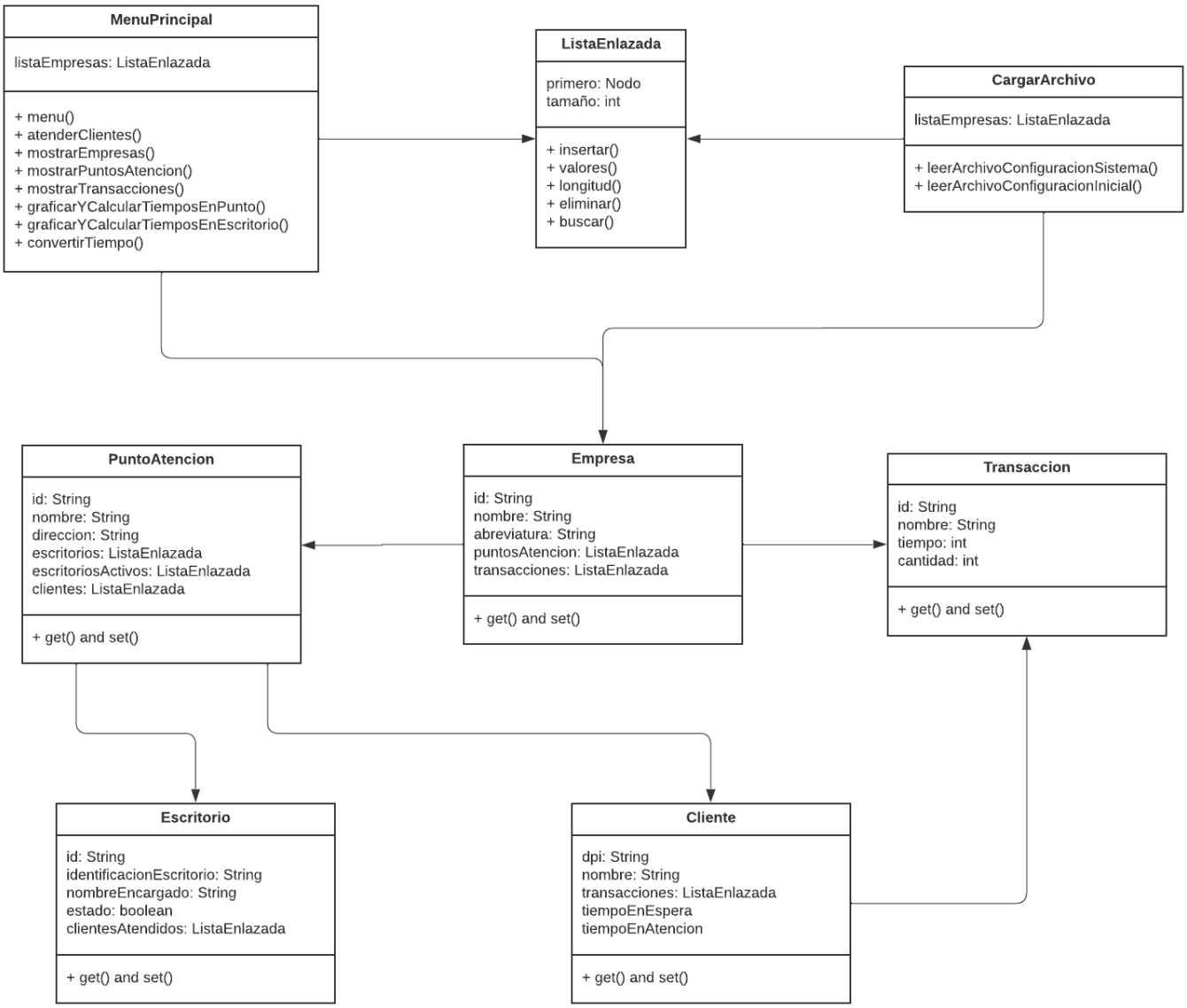
Conclusiones

- Para este proyecto se hicieron uso de estructuras de datos abstractas (TDA) utilizando el paradigma de la programación orientada a objetos, que permite hacer estructuras personalizadas que facilitan la resolución de este problema, estructurando las distintas partes del programa como módulos que se unen entre sí y son fáciles de modificar.
- Con el uso de los distintos algoritmos elaborados se puede concluir que son óptimos para la resolución de este problema ya que todos los datos al ser almacenados en listas enlazadas son borrados de la memoria cuando no se utilizan mas y esto hace que la memoria este mejor optimizada y el programa fluya mejor.
- La utilización de TDA's permite hacer un uso óptimo de la memoria ya que son estructuras personalizadas.

Referencias bibliográficas

- Problemas resueltos de listas / Mónica Adriana Carreño León 2010.
- CAIRÓ O., GUARDATI S., Estructura de Datos, México, Mc Graw-Hill, 2001.
- JOYANES A. LUIS, Algoritmos y Estructuras de Datos: Una Perspectiva en C., España, Mc Graw-Hill, 2005.

Anexos


MenuPrincipal
listaEmpresas: ListaEnlazada
+ menu()
+ atenderClientes()
+ mostrarEmpresas()
+ mostrarPuntosAtencion()
+ mostrarTransacciones()
+ graficarYCalcularTiemposEnPunto()
+ graficarYCalcularTiemposEnEscritorio()
+ convertirTiempo()
ListaEnlazada
primero: Nodo
tamaño: int
+ insertar()
+ valores()
+ longitud()
+ eliminar()
+ buscar()
CargarArchivo
listaEmpresas: ListaEnlazada
+ leerArchivoConfiguracionSistema()
+ leerArchivoConfiguracionInicial()
PuntoAtencion
id: String
nombre: String
direccion: String
escritorios: ListaEnlazada
escritoriosActivos: ListaEnlazada
clientes: ListaEnlazada
+ get() and set()
Empresa
id: String
nombre: String
abreviatura: String
puntosAtencion: ListaEnlazada
transacciones: ListaEnlazada
+ get() and set()
Transaccion
id: String
nombre: String
tiempo: int
cantidad: int
+ get() and set()
Escritorio
id: String
identificacionEscritorio: String
nombreEncargado: String
estado: boolean
clientesAtendidos: ListaEnlazada
+ get() and set()
Cliente
dpi: String
nombre: String
transacciones: ListaEnlazada
tiempoEnEspera
tiempoEnAtencion
+ get() and set()