Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2, 1er. Semestre 2021.

PROYECTO 2 SEGUNDO SEMESTR2 2022

202106484 – Wendi Paulina Vicente Pérez

Resumen

Hoy en día el tiempo es un factor muy importante para todo ser humano y el hacer largas colas de espera en algún sitio es algo muy tedioso y que a menudo deseamos evitar es por esto se inicia un prototipo de servicio de atención para que el usuario tenga contemplado y pueda escoger un punto de servicio que mejor se ajuste a su tiempo y localidad esto con ayuda de todos los datos proporcionados en el enunciado del proyecto mencionado y bajo el lenguaje Python.

Tratando de tomar en consideración varios factores que se mencionan en dicho proyecto como lo es los clientes en espera, puntos de atención, escritorios activos en dichos puntos de atención, además de dar un distintos tiempos para que el usuario e de una idea de cuánto tiempo le tomara realizar sus transacciones en la empresa y punto de atención que haya seleccionado con antelación.

Palabras clave

Nodo
Versionamiento
Archivo xml

Abstract

Nowadays, time is a very important factor for every human being and making long queues somewhere is something very tedious and that we often want to avoid, which is why a prototype of a customer service is started so that the user has contemplated and you can choose a service point that best suits your time and location with the help of all the data provided in the statement of the mentioned project and under the Python language.

Trying to take into consideration several factors that are mentioned in said project, such as waiting customers, service points, active desks at said service points, in addition to giving a different one so that the user can have an idea of how much time It will take to carry out your transactions in the company and point of service that you have selected in advance.

Keywords

Node
Versioning
xml file

Introducción

En el presente trata de dar a conocer el fundamento teórico en la cual se basa la lógica de programación que se ha implementado para la realización de este proyecto. Así como lo es la teoría en la que se basan los tipos de datos abstractos que es lo que se ha implementado en este proyecto, para ser precisos el TDA pila TDA Cola y una lista simplemente enlazadas con las cuales se les da vida a dichos TDA's. los cuales tienen distintas formas de procesarse, pero esto se verá más claro en el desarrollo del tema

Desarrollo del tema
Estructura de datos Abstractos:

Una estructura es una colección de una o más variables, de iguales o diferentes tipos, agrupadas bajo un solo nombre, es decir, es un tipo de dato compuesto que permite almacenar un conjunto de datos de diferente tipo (agrupar un grupo de variables relacionadas entre sí) que pueden ser tratadas como una unidad (bajo un mismo nombre).

Las estructuras pueden contener tipos de datos simples y tipos de datos compuestos. Los tipos de datos simples (o primitivos) son: carácter, números enteros o números de punto flotante. Los tipos de datos compuestos son: los arreglos y las estructuras. Por lo tanto, los tipos de datos abstractos (TDA) en lenguaje Python se pueden crear a través de una estructura. Cada ente u objeto es una abstracción de un elemento y, por ende, se puede modelar a través de una estructura

Pila

La pila es un contenedor de nodos y tiene dos operaciones básicas: **push** (o apilar) y **pop** (o desapilar). «Push» añade un nodo a la parte superior de la pila, dejando por debajo el resto de los nodos ya presentes en la pila. «Pop**»** devuelve y elimina el actual nodo superior de la pila. Una metáfora que se utiliza con frecuencia es la idea de una pila de platos dispuesta en una cafetería en un contenedor con un muelle que mantiene la pila a nivel. En esa serie, solo el primer plato es visible y accesible para el usuario, todos las demás permanecen ocultos. Como se añaden nuevos platos, cada nuevo plato se convierte en la parte superior de la pila, permaneciendo escondidos debajo los demás. A medida que el plato superior se extrae de la pila, el inmediatamente inferior pasa a ocupar la parte superior de la pila. Dos principios importantes son ilustrados por esta metáfora: únicamente se accede al plato que se encuentra en la parte superior (el último en depositarse), y el resto de platos de la pila permanecen ocultos. Para extraer un plato distinto al superior habrá que extraer antes los que se encuentran sobre él.

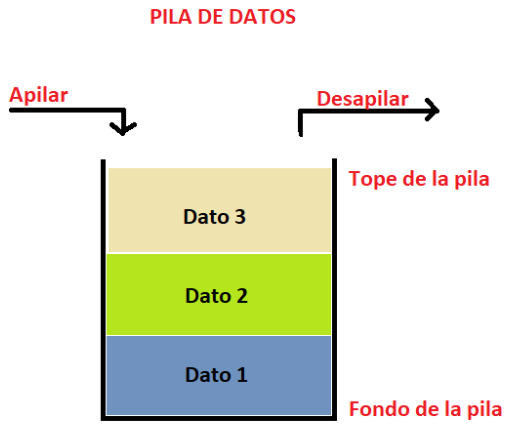
Operaciones:

Habitualmente, junto a las dos operaciones básicas de apilar y desapilar (push, pop), las pilas puede implementar otra serie de funciones:

- **Crear (constructor):** crea la pila vacía.
- **Tamaño (*size*):** regresa el número de elementos de la pila.
- **Apilar (*push*):** añade un elemento a la pila.
- **Desapilar *(pop*):** lee y retira el elemento superior de la pila.
- **Leer último** (***top* o *peek***): lee el elemento superior de la pila sin retirarlo.
- **Vacía (*empty*):** devuelve cierto si la pila está sin elementos o falso en caso de que contenga alguno.

Una pila puede implementarse fácilmente ya sea mediante una matriz o una lista enlazada. Lo que identifica a una estructura de datos como una pila en cualquier caso no es su estructura sino su interfaz: al usuario solamente se le permite colocar y extraer datos en el modo que se espera de una pila y algunas otras operaciones auxiliares.

Otro tipo de estructura de datos es la *cola* (FIFO, del inglés *First In First Out*), «primero en entrar, primero en salir».



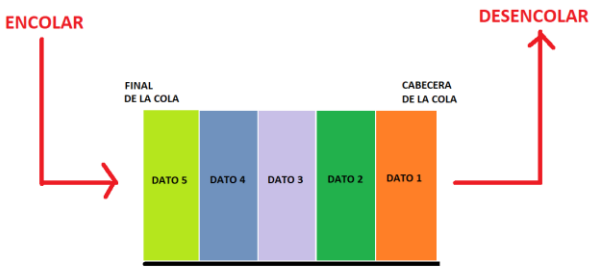
Una cola es una estructura de datos, caracterizada por ser una secuencia de elementos en la que la operación de inserción *push* se realiza por un extremo y la operación de extracción *pop* por el otro. También se le llama estructura FIFO (del inglés First In First Out), debido a que el primer elemento en entrar será también el primero en salir.

Las colas se utilizan en sistemas informáticos, transportes y operaciones de investigación (entre otros), dónde los objetos, personas o eventos son tomados como datos que se almacenan y se guardan mediante colas para su posterior procesamiento. Este tipo de estructura de datos abstracta se implementa en lenguajes orientados a objetos mediante clases, en forma de listas enlazadas.

Operaciones:

- **Crear**: se crea la cola vacía.
- **Encolar** (añadir, entrar, push): se añade un elemento a la cola. Se añade al final de esta.
- **Desencolar** (sacar, salir, pop): se elimina el elemento frontal de la cola, es decir, el primer elemento que entró.
- **Frente** (consultar, front): se devuelve el elemento frontal de la cola, es decir, el primero elemento que entró.

Teóricamente, la característica de las colas es que tienen una capacidad específica. Por muchos elementos que contengan siempre se puede añadir un elemento más y en caso de estar vacía borrar un elemento sería imposible hasta que no se añade un nuevo elemento A la hora de añadir un elemento podríamos darle una mayor importancia a unos elementos que a otros (un cargo VIP) y para ello se crea un tipo de cola especial que es la cola de prioridad.

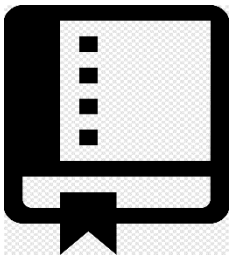


¿Qué es el control de versiones?

Se llama control de versiones a los métodos y herramientas disponibles para controlar todo lo referente a los cambios en el tiempo de un archivo. Difícilmente un archivo de código o un documento de texto está terminado con la primera escritura; necesita cambios o reescrituras para corregir errores, modificar su contenido... A medida que el documento cambia existen dos opciones, mantener un historial de cambios o dejar que evolucione sin memoria. El control de versiones es un método estándar para mantener esta memoria haciendo además que sea útil para el desarrollo futuro. En documentos sencillos como un ensayo o un pequeño programa la memoria no es algo esencial, pero en la escritura de un libro o un programa con centenares de páginas y una docena de manos involucradas no hay otra manera de trabajar. Esta es precisamente la palabra clave, mantener un control de las versiones de todos los archivos de un proyecto es una manera de trabajar completamente estandarizada; todas las prácticas tienen un nombre. La buena noticia es que todo este formalismo es generosamente recompensado por el uso de sistemas de control de versiones automáticas como CVS, Subversion, git…

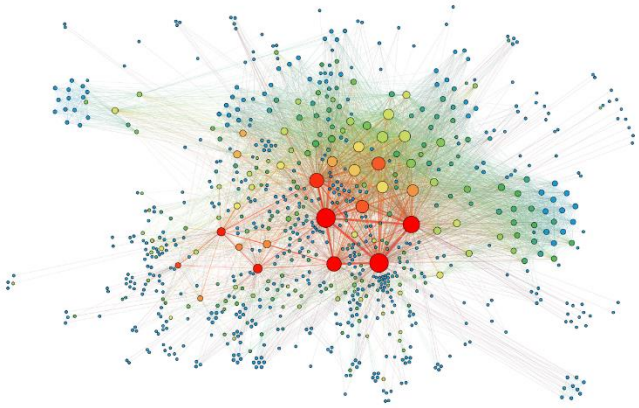
¿Qué son los repositorios y las copias de trabajo?

Todas las herramientas de control de versiones se basan en la típica comunicación servidor-cliente. 2 El control de versiones Desde el punto de vista de los documentos se podría decir que el repositorio es el código está en el servidor y la copia de trabajo en el cliente. La copia de trabajo es entonces una imagen del contenido en el repositorio de código y es lo que se utiliza para el trabajo diario (modificar archivos existentes, crear documentos nuevos...). Todos los cambios que se hagan en dicha imagen no son definitivos hasta que se suban al repositorio. El control de versiones se ve influido en gran parte por cómo se realiza esta comunicación entre copia de trabajo-repositorio. El lector puede plantearse la necesidad de duplicar los datos del proyecto. Esta arquitectura tiene la ventaja de que pueden existir tantas copias de trabajo como sean necesarias, normalmente tantas como participantes tenga el proyecto. Cada participante tiene su copia de trabajo y es el servidor el encargado de que los cambios aportados por los desarrolladores no entren en conflicto



Graphviz

Graphviz es un conjunto de herramientas que nos permite representar información estructural por medio de diagramas gráficos o de redes. Una herramienta que pueda dibujar grafos automáticamente es de gran importancia para aplicaciones que dependan de la visualización de grafos que representen algo significativo, han tenido éxito en campos tales como ingeniería de software, base de datos y diseño de redes de Internet, entre otros. Esta tecnología visualiza los grafos de diferentes maneras. Existen diferentes visualizadores de código libre accesibles en Internet tales como WebDot, Grappa, y Graphviz



Conclusión

En base a la información planteada y las necesidades de nuestro proyecto se puede decir que la implementación de pilas y colas para la gestión de nuestro proyecto es una muy buena opción ya que cuentan con métodos que nos facilitan la gestión de información que se manejara de cada uno de nuestras empresas, puntos de atención, escritorios y puntos de atención , también hay que tomar en cuenta que se puede implementar listas enlazada independientemente del tipo que esta sea como lo es la lista simplemente enlazada o doblemente enlazada esto dependerá de la facilidad que estas nos proporcionan en base a lo que se desea trabajar con ellas al igual que siempre tener en cuenta un buen manejo de versiones en cualquier herramienta que nos ayude con este manejo

Referencias

Copyright © 2006 Guillem Borrell Nogueras El control de versiones

https://sites.google.com/site/estdatinfjiq/unidad-iii-listas-enlazada

Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2, 1er. Semestre 2021.

Anexos:

OBJETIVO GENERAL

Se busca que el estudiante sea capaz de dar una solución al problema que se le plantea, mediante la lógica y conocimientos que se le han impartido durante la clase y que han sido aplicados en el laboratorio.

OBJETIVOS ESPECÍFICOS

- Que el estudiante sea capaz de aplicar abstracción a un problema dado.
- Implementar una solución utilizando el lenguaje de programación Python.
- Utilizar estructuras de programación secuenciales, cíclicas y condicionales
- Que el estudiante genere reportes con la herramienta Graphviz.
- Que el estudiante sea capaz de manipular archivos XML.
- Que el estudiante desarrolle programación orientada a objetos utilizando estructuras propias
- Que el estudiante utilice los conceptos de TDA y aplicarlos a memoria dinámica.

El archivo XML para configurar el Sistema de atención al cliente

```
<?xml version="1.0"?>
<listaEmpresas>
  <empresa id="$codigoEmpresa">
    <nombre> $nombreEmpresa <nombre/>
       <abreviatura> $abreviatura <abreviatura/>
       <listaPuntosAtencion>
         <puntoAtencion id="$codigoPunto">
           <nombre> $nombrePuntoServicio <nombre/>
           <direccion> $direccionPuntoServicio <direccion/>
              <listaEscritorios>
                <escritorio id="$codigoEscritorio">
                  <identificacion> $identificacionEscritorio <identificacion/>
                  <encargado> $nombreEncargado <encargado/>
                <escritorio/>
                ...
              <listaEscritorios/>
      <puntoAtencion/>
         ...
    <listaPuntosAtencion/>
       <listaTransacciones>
         <transaccion id="$codigoTransaccion">
           <nombre> $nombreTransaccion <nombre/>
              <tiempoAtencion> minutosAtencion <tiempoAtencion/>
         <transaccion/>
         ...
       <listaTransacciones/>
  <empresa/>
  ...
</listaEmpresas>
```

El archivo XML para inicializar la prueba del sistema de atención a clientes

```
<?xml version="1.0"?>
<listadoInicial>
   <configInicial id="$codigoConfiguracion" idEmpresa="$codEmpresa" idPunto="$codPunto">
     <escritoriosActivos>
        <escritorio idEscritorio="$codEscritorio"/>
        <escritorio idEscritorio="$codEscritorio"/>
        ...
     <escritoriosActivos/>
     <listadoClientes>
          <cliente dpi="$DPI">
           <nombre> $nombreCliente <nombre/>
           <listadoTransacciones>
               <transaccion idTransaccion="$codTrans" cantidad="cantTrans"/>
               <transaccion idTransaccion="$codTrans" cantidad="cantTrans"/>
               ...
           <listadoTransacciones/>
          <cliente/>
          ...
     <listadoClientes/>
   <configInicial/>
   ...
<listadoInicial/>
```

Diagrama de Clases: