SISTEMAS DE GESTIÓN EMPRESARIAL

TEMA 5. ELEMENTOS AVANZADOS DE PYTHON

PROYECTO CON PYTHON

El proyecto consiste en realizar una aplicación para gestionar un parking. Es necesario que el parking gestione los clientes que se abonan al parking, así como la información de usos de las plazas del parking por los distintos vehículos que entran y salen. El parking dispone de N plazas en total: el 70% son específicas para turismos, 15% para aparcar motocicletas y 15% para personas de movilidad reducida.La tarificación por minutos es la siguiente:

- Turismos - 0,12 €
- Motocicletas - 0,08 €
- Personas de movilidad reducida - 0,10 €

La aplicación tiene dos subsistemas: **una zona cliente y una zona administrador**.

ZONA CLIENTE

En la zona cliente de la aplicación se pueden realizar básicamente las siguientes acciones:

Depositar vehículo

- El sistema informa en todo momento del número de plazas libres que existen de cada tipo.
- El cliente introduce la matrícula y el tipo (turismo, motocicleta o caravana). El sistema asigna una plaza de las posibles, si existen plazas libres. El cliente debe aparcar en la plaza asignada.
- El sistema genera un ticket donde aparece la matrícula del vehículo, la fecha de depósito, el identificador de la plaza asignada y un pin de seis dígitos numéricos que servirá para retirar el vehículo posteriormente. Este ticket aparecerá en la consola del sistema. Esta información se debe guardar **de forma persistente**[1] para poder ser consultada cuando el cliente proceda a la retirada del vehículo.
- No es necesario guardar información de los clientes si se hace uso del parking sin abono.

Retirar vehículo

- El cliente introduce la matrícula, el identificador de la plaza y el pin asociado. El sistema calcula el coste total a pagar e informa de la tarifa al cliente.

[1] Como por ahora no sabemos acceder a una base de datos desde Python, lo guardaremos en una colección del tipo que creas más conveniente, y persistiremos la colección con pickle

- Una vez realizado el pago, el cliente puede recoger el vehículo y salir del parking. El sistema actualiza el número de plazas libres, así como la información relativa al coste final y fecha de salida del vehículo.
- Los cobros realizados se deben guardar en una colección[2] para, posteriormente, poder consultarlos desde la zona de administración.

Depositar abonados

- El cliente abonado introduce en el sistema la matrícula del vehículo y su DNI. Se supone que un cliente tiene un solo vehículo y un vehículo pertenece a un solo cliente.
- El cliente aparca el vehículo en la plaza asignada al abono y el sistema actualiza el estado de la plaza para saber que el vehículo del abonado está en el parking. Asocia siempre el mismo pin para poder retirar el vehículo tantas veces como sea necesario. El PIN se genera al crear el abono del cliente.
- De los clientes abonados es necesario saber su DNI, nombre, apellidos, número de tarjeta de crédito, tipo de abono que tienen y su email.

Retirar abonados

- El cliente introduce la matrícula, el identificador de plaza asignada y el pin.
- El sistema actualiza el estado de la plaza del parking, que no queda libre, sigue estando reservada, pero el vehículo del abonado no está en el parking.

ZONA ADMINISTRADOR

La zona admin de la aplicación se encarga de:

Estado del parking

Controlar el estado del parking. Se debe mostrar por pantalla el estado de las plazas (libre, ocupada, abono ocupada y abono libre) y el identificador de cada plaza.

Facturación

Entre fechas. El sistema solicita dos fechas[3] y horas concretas para saber los cobros realizados entre las mismas. Los abonos no se contemplan en esta opción.

Consulta de Abonados.

El sistema informa de los abonos anuales, con los cobros realizados.

Abonos

Alta. El sistema solicita datos personales del abonado y un número de tarjeta donde se realizan los cargos mensuales del abono. El cliente debe elegir entre los distintos abonos: mensual (25€), trimestral (70€), semestral (130€) y anual (200€). Todos los abonos tienen una fecha de activación y una fecha de cancelación. La fecha de activación se actualiza con la fecha en la que se da de alta y la fecha de cancelación se calcula en función del tipo de abono.

---

[2] La colección anterior, esta, y todas, se deben persistir en ficheros con pickle.
[3] Sería bueno usar objetos de tipo datetime:
https://docs.hektorprofe.net/python/modulos-y-paquetes/modulo-datetime/

Modificación. Existirá la opción de cambiar los datos personales del abonado o bien cambiar la fecha de cancelación del abono, porque el abono ha sido renovado.

Baja. Se eliminará el registro del abonado pero no se podrán borrar los datos asociados a su facturación.

Caducidad de abonos

Mes. El sistema solicita un mes y nos informa de los abonos que caducan en ese mes.

Consultar últimos 10 días. El programa informa por consola de los abonos que caducan en los siguientes 10 días a la fecha actual.

ELEMENTOS A UTILIZAR EN EL PROYECTO

- Se tendrán que utilizar todos los elementos necesarios y conocidos (*properties*, métodos mágicos, herencia, polimorfismo, …)
- Se debe usar modularización, es decir, que tenemos que repartir nuestro código en diferentes módulos y paquetes.
- Se deberán manejar todas las excepciones pertinentes.
- Como se ha indicado más arriba, ya que aun no manejamos el acceso a bases de datos desde Python, tendremos que persistir nuestras colecciones a través de pickle. Para esto, podemos utilizar varios enfoques:
  - En cada modificación de una colección, almacenar los cambios de las colecciones en los ficheros correspondientes.
  - Leer desde los ficheros al iniciar el programa, y añadir una opción al menú para hacer un volcado de datos a ficheros explícitamente.
  - Investigar para añadir un mecanismo *automático* que haga un almacenamiento de los datos cada cierto tiempo (por ejemplo 5 minutos)[4].

ELEMENTOS COMPLEMENTARIOS Y DE AMPLIACIÓN

Puedes tener en cuenta este apartado si ya has terminado todo lo anterior, o si en un futuro quieres investigar:

- En un parking real la información iría a parar a una base de datos en lugar de a ficheros. En Python tenemos diferentes formas de conectar con ella. Puedes investigar sobre SQLAlchemy, que es uno de los ORMs más conocidos de Python.
- Por otro lado, en un parking real tendríamos a la entrada una cámara que sería capaz de reconocer la matrícula del vehículo. Puedes investigar sobre la librería OpenCV para realizar el reconocimiento por visión artificial del texto de la matrícula (para comprobar que funciona, tu programa podría recibir como argumento una foto de la parte frontal de un coche donde se vea bien la matrícula.
- Por último, tu aplicación merece no ser de consola. Así que tienes dos opciones:

[4] Para esto, te hace falta investigar sobre programación concurrente, y así crear un hilo de ejecución diferente al principal que se encargue de realizar el almacenamiento. Echa un vistazo a https://code.tutsplus.com/es/articles/introduction-to-parallel-and-concurrent-programming-in-python--cms-28612

- Añadir un frontend visual para escritorio. En Python tenemos diferentes alternativas. Una de ellas es Tkinter. La otra alternativa sería transformar tu aplicación en una aplicación web; para ello tienes disponibles dos frameworks: el más clásico y con mayor documentación es Django (que tiene su propio ORM); otro que es más ligero y moderno es Flask. Tienes mucha documentación online y en OW para implementarlo.

PLAZOS, ENTREGA y CALIFICACIÓN

La publicación oficial de este enunciado es el Viernes 11 de Diciembre a las 8:00, aunque se dejará abierto antes del puente de la Inmaculada.

Este proyecto tendrá el valor de examen para el Tema 5. Se publica el 11 de Diciembre y se debe entregar antes del lunes 21 de Diciembre a las 19:00 horas.

Para aquellos días a partir del 11 de Diciembre que no haya trabajo presencial publicado en el aula virtual de SGE (recuerda que debes mirar los Temas 4 y 5) la tarea consistirá en continuar trabajando en el proyecto.

Como entrega, se debe adjuntar la URL de un repositorio en github donde esté incluido el proyecto.

El proyecto tendrá valor de examen de la Unidad Didáctica 5. Se utilizará una rúbrica de evaluación que tendrá en consideración los siguientes aspectos:

- Cumplimiento de los requisitos: 35%
- Uso correcto de clases (herencia, polimorfismo, composición, …) y modularización: 15%
- Tratamiento de las excepciones necesarias: 10%
- Persistencia de la información: 20%
- Originalidad del código (que no sea igual que el de los compañeros, dar soluciones creativas a algún problema, …): 10%
- Inclusión de algún elemento de ampliación: 10%.

C/ CONDES DE BUSTILLO 17 | 41010 SEVILLA | T 954 331 488 | https://triana.salesianos.edu