Tienda UPM

UNIVERSIDAD
POLITÉCNICA
DE MADRID

Practica de Programación Orientada a Objetos 2025-2026

Universidad Politécnica de Madrid

E.T.S. de Ingeniería en Sistemas Informáticos

Departamento de Sistemas Informáticos

Entrega E2:

El cliente amplía el alcance: se incorporan al sistema **Usuarios** que podrán ser **Clientes** o **Cajeros**.

Se agregan **nuevos tipos de productos** para el ticket como son **Comidas en campus** y **Reuniones**, **con fecha de caducidad**, sin categoría y con número máximo de participantes (máximo 100) y cuyo precio es calculado por persona. No podrán crearse comidas o reuniones imposibles a nivel temporal. Para las reuniones se requiere un tiempo mínimo de planificación de 12 horas mientras que para las comidas es necesario 3 días, por lo que, estos tiempos, deben respetarse a la hora de crear el producto y cerrar el ticket.

También se extienden los productos con una versión **personalizable**. Estos productos personalizados tienen una **lista máxima de textos** permitidos para el producto y el precio de estos es calculado agregándole al precio del producto sin personalizar un recargo del 10% por cada texto personalizado agregado. De esto productos se conoce el número máximo de textos personalizables **por producto**. No todos los productos básicos serán personalizables. Un producto básico no puede pasar a personalizable en el futuro.

Para la gestión de usuarios se piden comandos de **altas/bajas** de usuarios y cajeros. Respecto a los tickets se necesitan comandos de **creación/impresión/cierre.**

Todas los tickets, clientes, cajeros y productos tendrán un identificador diferente que los define en el sistema dentro de su categoría. En el caso de los clientes, este identificador será el DNI en el caso de los clientes introducido en el registro. Para los cajeros, será un código compuesto por un duo de letras “UW” (UPM Worker) más 7 dígitos aleatorios, este podrá generarse internamente si no se pasa como parámetro en su creación y debe asegurarse que no hay dos empleados con el mismo identificador. Los productos registrados en la aplicación podrán registrarse con un identificador o se generara de manera automática internamente al igual que los cajeros, igualmente, debe asegurarse que no hay dos productos con el mismo id. Los tickets tendrán un identificador compuesto por la fecha de apertura “YY-MM-dd-HH:mm-“ más un numero aleatorio de 5 cifras, cuando se cierra un ticket, debe actualizarse el ID agregando la fecha de cierre “-YY-MM-dd-HH:mm”, este identificador puede pasarse como parámetro de construcción como en otros elementos de la aplicación o generarse, en cualquiera de los dos casos hay que asegurar que no hay repetición de identificadores.

Los clientes se darán de alta por parte de un cajero con el nombre, DNI, correo electrónico y cajero que lo da de alta, por lo que el usuario tendrá en su registro el cajero que le dio de alta, pero el cajero no guardara registro de los usuarios que dio de alta.

Los cajeros se darán de alta solo con un nombre y correo electrónico de empresa.

Los cajeros no podrán ser usuarios de manera simultánea y deberán registrarse como nuevos usuarios con su correo personal si así lo desean, aunque nada impide que se registren a sí mismos.

Los tickets se crearán con un usuario y un cajero asociado, el usuario conocerá los tickets que tiene asociados, pero el ticket no tendrá constancia del usuario o el cajero. Los cajeros guardaran constancia de los tickets creados por ellos. Si se borra un cajero se borrarán los tickets creados por él. Los tickets tendrán un estado actual (VACIO, ACTIVO, CERRADO).

Imprimir un ticket implicará el cierre de este ya que será el sinónimo de imprimir la factura. Cuando un tickets está cerrado puede printarse de nuevo, pero no puede abrirse ni añadir

nuevos productos porque ya se ha emitido la factura. Las operaciones de add, remove, print y close solo podrán hacerse por el mismo cajero que inicio el ticket.

Los comandos de E1 pasa a estar ampliados por:

- Clientes/Cajeros
  - *client add "<nombre>" <DNI> <email> <cashId>*
  - *client remove <DNI>*
  - *client list (incluye el dato del cash que lo creo y ordenados por nombre)*
  - *cash add [<id>] "<nombre>"<email>*
  - *cash remove <id>*
  - *cash list (Ordenados por nombre y sin mostrar sus tickets)*
  - *cash tickets <id> (Muestra los tickets del cajero ordenados por el Id del ticket, mostrando solo el ID y el estado)*

- Tickets (deben extenderse y ampliarse en consecuencia)
  - *ticket new [<id>] <cashId> <userId>*
  - *ticket add <ticketId><cashId> <prodId> <amount> [--p<txt> --p<txt>]*
    - *--p para cada personalización.*
    - *Un producto personalizable puede comprarse sin personalizar*
    - *amount es el numero de productos a añadir para los productos normales y personalizados, y el numero de personas para los productos de reuniones y comidas)*
    - *No puede añadirse nuevamente un producto de reunión o comida que ya esta en el ticket.*
  - *ticket remove <ticketId><cashId> <prodId>*
  - *ticket print <ticketId> <cashId>*
    - *cierra el carro y lo imprime en orden alfabético*
  - *ticket list*
    - *Muestra los tickets ordenados por id del cajero*

- Productos (deben extenderse y ampliarse en consecuencia)
  - *prod add [<id>] "<name>" <category> <price> [<maxPers>]*
    - *si tiene <maxPers> se considerara que el producto es personalizable)*
  - *prod update <id> NAME|CATEGORY|PRICE <value>*
  - *prod addFood [<id>] "< name>" <price> <expiration: yyyy-MM-dd> <max_people>*
    - *El precio es por persona apuntada*
  - *prod addMeeting [<id>] "<name>" <price> < expiration: yyyy-MM-dd> < max_people >*
    - *El precio es por persona apuntada*
  - *prod list*
  - *prod remove <id>*

- Generales (deben extenderse y ampliarse en consecuencia)
  - *help (lista los comandos)*
  - *echo "<text>" (imprime el texto en el valor texto)*
  - *exit*

PRACTICA - POO 25-26

Entregables

- Código fuente y empaquetado (jar) comprimidos en un zip subido a moodle.
- El código debe poder ejecutarse pasando un archivo txt como entrada en el terminal en lugar del teclado interactivo. En este caso, debe printar el comando que llega desde el archivo y ejecutarlo para poder tener el mismo resultado que en la versión interactiva.
    - java -jar MyTiendaUPM.jar **input.txt**
- Diagrama UML del modelo propuesto en un formato legible (PNG, JPG, SVG,…). Se deberá justificar el diseño de clases y uso de librerías (si lo hubiera). Se puede pedir justificación de cambios respecto a la primera entrega.

Consideraciones

- Para poder defender el proyecto el código entregado no debe tener errores de compilación.
- Se debe respetarse el formato de entrada de comandos y salida mostrados en el enunciado de la práctica.
- No se pueden crear más comandos de los pedidos en el enunciado.
- Para poder defender el proyecto el código debe ser capaz de ejecutar sin errores todos los comandos del ejemplo propuesto (archivo separado a este) . Este punto será verificado en la defensa.

PRACTICA - POO 25-26