UNIVERSIDAD DE COSTA RICA

Escuela de Ciencias de la Computación e Informática

Curso: CI-0128 II ciclo 2025
Grupo: 2

Examen final Ingeniería de software

Instrucciones: Este examen es individual, lea cuidadosamente lo solicitado. El proyecto solicitado en la parte 2 debe ser entregado el sábado 6 de noviembre antes de las 11:59PM en mediación virtual. Deben entregar los artefactos solicitados en la sección, además de enviar el link del proyecto en github para ser descargado, junto con las instrucciones para poder ser ejecutado en un ambiente local (Readme)

Part 1 (30%)

Selección única: Responda el formulario de selección múltiple que se encuentra en mediación virtual. Solo cuenta con un intento para responder y 35 min para completar las 30 preguntas, las preguntas se abrirán el 3 de noviembre a las 7:30am. Contarán con un intento y podrán responderlas por hasta las 10:30am.

Parte 2 (70%)

Una empresa cuenta con un API que desea utilizar para crear el sistema web de una máquina expendedora de café. El API del backend se encuentra en el siguiente enlace: https://github.com/rebeca-ov/examen2
Los endpoints del backend intentan cumplir con las historias de usuario que se detallan más adelante. Sin embargo, antes de construir el frontend, la empresa quiere invertir tiempo en **refactorizar** el backend actual, **resolver** sus problemas existentes y **agregar** pruebas unitarias.
En el backend **no deben utilizar una base de datos**; se pueden seguir usando estructuras de datos en memoria, tal como funciona actualmente.
Para el frontend, **únicamente deben diseñar una sola pantalla**, sin pantallas adicionales de configuración. Esta pantalla debe cumplir con lo solicitado en las historias de usuario y utilizar el backend previamente refactorizado.

1. Como consumidor quiero ver los tipos de café disponibles en la máquina para poder elegir cuales quiero.

   AC: Se debe visualizar el estado actual de cuántos cafés se pueden preparar y su precio. Inicialmente la máquina solo debe tener los siguientes tipos de cafés:

   a. 10 cafés americanos (precio 950 colones)
   b. 8 capuchinos (precio 1200 colones) .
   c. 10 lates (precio 1350 colones)
   d. 15 mocachinos (precio 1500 colones)

   AC: Cada vez que una compra es completada debe verse el cambio reflejado en la pantalla con la cantidad nueva por tipo de café.

AC: Si un tipo de café se queda sin unidades, no debe permitir seleccionarlo para compra.

2. Como consumidor quiero poder ingresar la cantidad de cafés que quiero de cada tipo para poder realizar una compra

   AC: el sistema debe tener la opción de escoger el tipo y la cantidad de cafés que se desean. Si el usuario ingresa una cantidad mayor a la que tiene el sistema un mensaje de error debe aparecer

   AC: El sistema debe enseñar el costo total de la compra en pantalla, cada vez que se selecciona otro café.

3. Cómo consumidor quiero poder ingresar la cantidad de dinero que quiero ingresar en el sistema.

   AC: El sistema debe admitir el pago con monedas de 500, 100, 50 y 25 colones. Además debe aceptar el ingreso de billetes de 1000 colones

   AC: Inicialmente el sistema debe contar con la siguiente cantidad de monedas para dar el vuelto:

   i. 20 monedas de 500 colones
   ii. 30 monedas de 100 colones
   iii. 50 monedas de 50 colones
   iv. 25 monedas de 25 colones

   AC: Si el sistema no tiene más monedas para realizar el vuelto, debe reflejar un mensaje de fuera de servicio.

4. Cómo consumidor quiero ser notificado del vuelto al realizar mi compra para poder saber cual es el resultado final de la compra.

   AC: El sistema debe mostrar en pantalla el monto total que debe devolverle al consumidor. Esto debe mostrarse de la siguiente manera:

   Su vuelto es de 650 colones.
   Desglose:
   1 moneda de 500
   1 moneda de 100
   1 moneda de 50

   AC: Si el sistema no tiene suficientes monedas para el vuelto de la compra, debe mostrar un mensaje de “Fallo al realizar la compra”

Deberán entregar las siguientes artefactos

1. Diagrama UML definiendo cómo van a refactorizar el backend (20%)
2. Crear un repositorio y utilizar versionamiento, haciendo uso adecuado del mismo. Haga commits pequeños e individuales y bien documentados. (15%)
3. Código fuente. Se calificará que el código siga en lo posible las ideas de código limpio, además de los principios SOLID y de ser necesario patrones de diseño. (40%)
4. Pruebas: El código debe tener un buen nivel de cobertura por medio de pruebas unitarias.(25%)
5. Mock up de la pantalla principal (5% extra)