

# Documentación de Práctica Cafetería

Andrea Sofía Pais Dos Santos

October 29, 2025

## Índice

1. Descripción del problema .....	página 3
2. Requisitos funcionales .....	página 3
3. Requisitos no funcionales .....	página 3
4. Casos de uso .....	página 3
5. Historias de usuario .....	página 4
6. Tecnologías Utilizadas .....	página 4
7. Clases e Interfaz .....	página 5

# 1 Descripción del problema

Tenemos una cafetería que necesita organizar a los camareros y se repartan los clientes de la forma más óptima posible. Para ello creamos un programa que gestione la entrada de clientes y los reparta correctamente a los distintos camareros. Luego crear una pequeña aplicación que refleje el funcionamiento del programa, tiene que ser capaz de:

1. Al ejecutar el programa deben llegar los clientes a la cafetería.
2. Esos clientes por orden de llegada, son atendidos por el camarero que esté disponible.
3. Si el camarero atiende al cliente dentro del tiempo de espera del cliente, el mismo se va contento de la cafetería. Si al camarero no le ha dado tiempo de entregar el café en el tiempo de espera, el cliente se marcha insatisfecho.
4. Al atender todos los clientes se acaba el programa.

# 2 Requisitos funcionales

Los requisitos funcionales que va a tener el sistema son:

RF01: Cada cliente debe esperar su café durante el tiempo especificado.

RF02: Los clientes deben poder irse si se acaba el tiempo de espera puesto.

RF03: Los camareros deben atender clientes en orden de llegada y deben de continuar trabajando si un cliente se va.

RF04: Gestionar pedidos concurrentemente.

RF05: Mostrar estado de clientes y camareros.

# 3 Requisitos no funcionales

Los requisitos no funcionales que va a tener el sistema son:

RNF01: Mensajes informativos y comprensibles.

RNF02: En cada ejecución el proceso debe ser distinto.

RNF03: Los hilos deben finalizar correctamente.

RNF04: Interfaz intuitiva y fácil de usar.

# 4 Casos de uso




Descripción de casos de uso principales de la aplicación:

Caso de Uso	Descripción
Llegada de Cliente	El cliente llega y espera su café en un tiempo límite. Si se agota el cliente se va sino recibe su pedido.
Preparar café	El camarero atiende pedidos en orden de llegada, el cliente se va y continúa con el siguiente.
Gestionar Clientes	El usuario puede eliminar la receta o modificar cualquier de sus puntos cuando desee.

## 5 Historias de usuario

Identificación	Nombre	Tarea	Objetivo
HU1	Cliente	El cliente llega a la cafetería, pide un café y tiene un tiempo de espera	Pedir un café y esperar por el
HU3	Cliente	El cliente tiene que esperar (tiempo de espera) y si no tiene el café en ese tiempo se marcha	Ser atendido
HU4	Camareros	Gestionar los clientes que van a llegar	Atender pedidos
HU5	Camareros	Cada camarero tiene que comprobar que el cliente no está siendo atendido por otro camarero antes de atenderlo, si fue atendido pues pasar al siguiente	Atender a clientes que no son atendidos

## 6 Tecnologías Utilizadas

Por terminal	Usamos java en IntelliJ para crear el programa y que funcione por terminal	
Interfaz	Para la interfaz usamos JavaFX en IntelliJ para enseñar el programa de una forma más visual	
IDE	IDE que se usó para realizar ambas partes	

## 7 Clases e Interfaces

Explicación breve de cada clase y interfaz y las herramientas necesarias para llevar a cabo la aplicación. La "base de datos" no existe, los datos están añadidos manualmente, por un lado los camareros y los clientes.

### 7.1 Interfaz gráfica con JavaFX

#### 7.1.1 HelloController

- Declaramos variables:
  1. TextArea -> para el contenido principal de todo el flujo del ejercicio.
  2. BotonEmpezar -> al darle inicia la simulación y llama a la función "iniciar()".
  3. CamarerosArea -> espacio que va a guardar la información de los camareros, enseñando una lista.
  4. ClientesArea -> espacio que va a guardar la información de los clientes, enseñando una lista de clientes contentos y cuales no.
  5. Lista Camareros y Lista Clientes -> son los que se van a enseñar en las pantallas de la derecha de la interfaz, datos de los camareros y los clientes.
- Función al ejecutar el ejercicio, nos aparece un mensaje de darle a un botón para inicializar y no nos deja editar los elementos.
- Función "iniciar()" que funciona al pulsar en el botón "BotonEmpezar" y dentro de la función tenemos:
  1. La lista de datos de los camareros y la de los clientes.
  2. llamamos a las funciones de las listas de los clients y camareros para enseñarlas en las pantallas de la derecha.
  3. Iniciamos los camareros y con un new Thread con los clientes entrando por tiempos a la cafetería y esperamos a que acaben. Eso todo lo ejecutamos ".start()".
- Funciones para actualizar el estado de los clientes y los camareros obteniendo los datos de las clases.
- Funciones para listar los clientes y camareros, que nos los lista en las ventanas de la derecha de la interfaz.

Para verlo mejor, una captura de la pantalla principal de la interfaz.



## 7.2 Clases en Java

Tenemos las mismas clases que en la otra parte del ejercicio pero hay que añadirle unas cosas para que los datos se impriman en la interfaz.

### 7.2.1 Clase Camarero

- Modelo de datos con propiedades: id, nombre, ingredientes, instrucciones
- Atributos:
  1. nombre -> nombre del camarero.
  2. ocupado -> booleano que se pone en "false" al entregar el café y empieza en "null".
- Constructor de la clase con los atributos.
- Función "prepararCafe()" que pasándole los clientes por parámetro y con un try catch gestiona:
  1. Los camareros a que cliente está preparando el café
  2. El tiempo aleatorio que tarda el camarero a preparar el café.
  3. Esto lo pasa a la clase Cliente y marca el booleano "CafeEntregado" como "true".
- Función run que se ejecuta en el main con .start() y nos dice que los camareros están listo para atender.
- Getters y setters de todos los atributos.

- La clase Camarero tiene la misma estructura y mismos atributos que la realizada en el ejercicio de Java por terminal, lo único nuevo es un atributo HelloController controller que va a pasar los datos que necesitamos al HelloController.
- Y en vez que imprimirlo con un "sout" lo pasamos por un (controller."funcionParaPasarDatos"), esos datos se pasan y los enseña en los textArea.
- Los getters, setters y la función "run()" se mantienen igual.

### 7.2.2 Clase Cliente

- Atributos:
  1. nombre -> nombre del cliente.
  2. tiempoEspera -> cantidad de tiempo que está dispuesto a esperar el cliente a ser atendido.
  3. Lista Camareros -> lista de los camareros para ver cual está ocupado o no.
  4. cafeEntregado -> buleano que se pone en true al entregar el café
- Constructor de la clase con los atributos.
- Función run que se ejecuta en el main con .start(), que contiene un try catch en el encontramos:
  1. Los clientes van llegan a la cafetería
  2. Creamos una variable "ocupado" que inicia siendo "null" porque ninguno de los camareros está ocupado al principio.
  3. Recorriendo la lista de camareros buscamos un camarero libre y lo marcamos como ocupado. Y si está libre, llamamos la función "prepararCafé()" que se encuentra en la Clase camarero.
  4. Luego para ver si sigue estando en el tiempo de espera del cliente, busqué como calcular el tiempo desde que se inicia con "currentTimeMillis()" y se lo restamos al tiempo de inicio si es mayor que el tiempo de espera, el cliente se va, sino el cliente ha recibido el café y se ha ido.
- Getters y setters de todos los atributos.
- La clase Cliente tienen los mismos atributos y la misma estructura que la realizada en el ejercicio de Java anterior, lo nuevo es un atributo HelloController controller que va a pasar los datos que necesitamos de los clientes al HelloController.
- Y en vez que imprimirlo con un "sout" lo pasamos por un (controller."funcionParaPasarDatos"), esos datos se pasan y los enseña en los textArea.
- Los getters, setters y la función "run()" se mantienen igual.

### 7.2.3 Hello View (xml)

Todos los elementos visuales con los ids y las acciones vinculadas para que se ejecute todo correctamente en la interfaz.

- Tenemos un AnchorPane que envuelve a todos los elementos y es el que se conecta al controller HelloController.
- Hay dos fondos que se paran las pantallas, por un lado la ejecución y por otro los datos de Cliente y Camarero.
- Un TextArea con el fx:id="texto".
- El botón empezar que inicializar el simulador.
- Dos TextArea que nos enseñan los datos de los clientes y los camareros.
- Tenemos una leyenda que nos indica que representa cada icono.
- A parte hay un archivo "style.css" que tiene el estilo del botón, los fondos, etc.