**Método de la ingeniería**

**Aplicación para problema de tienda de videojuegos**

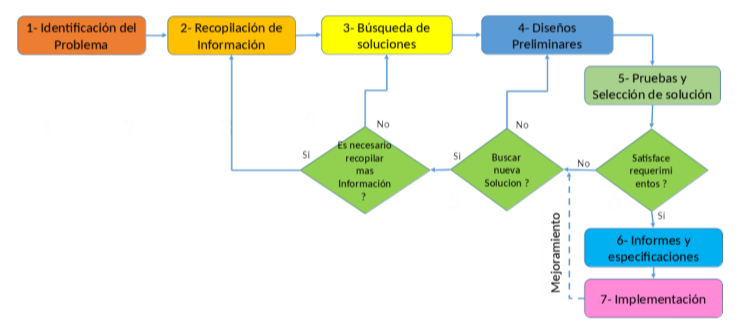
**Contexto problemático**

Una tienda de videojuegos recién creada por un excéntrico millonario extranjero requiere una solución para la entrada, procesamiento y salida de los clientes a la tienda.

**Desarrollo de la solución**

Para resolver la situación anterior se eligió el Método de la Ingeniería para desarrollar la solución siguiendo un enfoque sistemático y acorde con la situación problemática planteada.

Con base en la descripción del Método de la Ingeniería del libro “Introduction to Engineering” de Paul Wright, se definió el siguiente diagrama de flujo, cuyos pasos seguiremos en el desarrollo de la solución.



**Paso 1: Identificación Del Problema**

Normalmente, en una tienda de videojuegos habitual, al cliente le suele tomar una cantidad de tiempo considerable buscar los títulos de su interés o incluso encontrar títulos nuevos que le parezcan interesantes para aventurarse a probarlos. Además, por diferentes razones la excesiva, y en algunos casos innecesaria, interacción extra con el personal de la tienda puede acarrear consigo que el tiempo de permanencia en la tienda de parte del cliente se extienda o que la experiencia de este no sea la mejor.

*Necesidades y Síntomas*

* Los clientes necesitan, antes de entrar al establecimiento, tener conocimiento del catálogo que tiene la tienda a disposición.
* En el catálogo, debe contener reseñas y críticas de los juegos al momento de ser consultados por el cliente.
* El cliente debe poder añadir a una lista de compra los juegos de su interés.
* La lista de compra de cada cliente, después de terminada, debe generar un código de identificación que se usará al momento de entrar al establecimiento.
* Cada juego debe de tener un código de identificación propio.
* En el local, deben existir un conjunto de estaciones que permitan la realización de consultas rápidas.
* Al ingresar el código de la lista se deben cargar los nombres de los videojuegos a la primera estación, la cual debe indicar los bloques o estanterías donde debe buscar cada uno de los juegos de su interés.
* El listado de videojuegos debe organizarse dependiendo de la ubicación de cada una de las estanterías, de tal forma que el comprador siga la mejor ruta.
* Si el juego se encuentra agotado, su código no aparecerá en la lista ordenada según la ubicación de su respectiva estantería.
* El cliente debe poder elegir entre 2 algoritmos de ordenamiento para cumplir la tarea de organizar su lista dependiendo de la ubicación de cada estantería.
* Al cliente se le debe asignar un cesto automatizado que le sigue para ir ubicando los juegos que va encontrando, uno encima del otro, siguiendo el orden suministrado en la etapa previa.
* Cada cliente debe tener un tiempo dependiendo de su entrada a la tienda, al cual se le suma el tiempo que tardó el cliente buscando los videojuegos.
* El orden en el que los clientes entran a la caja, está dado por el tiempo en el que entró el cliente a la tienda más el tiempo que le tomó recoger los juegos.
* Todos los clientes deben hacer una única cola aunque haya varios cajeros o puntos de atención.
* En caso de haber puntos de atención disponibles, se atenderán en simultáneo tantos clientes como puntos de atención disponibles haya.
* El orden de salida puede variar, pues depende de cuánto se demore en la caja cada cliente, lo cual está dado por la cantidad de elementos que van a comprar.
* Los videojuegos, al momento de ser facturados el orden será de la forma en que el último juego añadido al cesto deberá ser el primero facturado y empacado.
* Después de completar la facturación, debe indicarse el costo total de los juegos listados por cada cliente.

*Definición del problema*

La tienda de videojuegos requiere el desarrollo de un simulador de compras que permita que los caleños conozcan un poco acerca de cómo funcionará esta nueva tienda en la ciudad.

**Paso 2: Recopilación de Información**

Con el objetivo de entender un poco más el funcionamiento de la aplicación, es necesario conocer aquellos algoritmos que van a hacer posible la ejecución de esta.

**Generics:** La programación con generics nos permite crear código reusable en múltiples entidades. Con este tipo de código genérico se trabajará automáticamente con el tipo de datos pasados a su parámetro de tipo. También, permiten al compilador informar de muchos errores de compilación que hasta el momento solo se descubrirán en tiempo de ejecución, al mismo tiempo, permiten eliminar los cast simplificando, reduciendo la repetición y aumentando la legibilidad del código.

**Stacks:** La clase Stack es una clase de las llamadas de tipo LIFO (Last In - First Out en español Último en Entrar - Primero en Salir). Es una pila, la cual tiene una secuencia de elementos del mismo tipo en la que el acceso a la misma se realiza por un único lugar denominado cima: Las operaciones que caracterizan la pila son las de introducir un nuevo elemento sobre la cima (push) y la de extraer el elemento situado en la cima (pop).

**Queue:** Una Cola o Queue es una estructura de datos que sigue la Filosofía FIFO (First In – First Out en español Primero en Entrar - Primero en Salir). Los elementos pueden ser insertados en cualquier momento, pero solamente el elemento que ha estado en la cola más tiempo puede ser removido en cualquier momento.

**TAD:** Un Tipo abstracto de datos (TAD) es un conjunto de datos u objetos al cual se le asocian operaciones. El TAD provee de una interfaz con la cual es posible realizar las operaciones permitidas, abstrayéndose de la manera en cómo están implementadas dichas operaciones.

**Tablas Hash:** Una Hashtable Java es una estructura de datos que utiliza una función hash para identificar datos mediante una llave o clave. En otras palabras, sería bastante conveniente poder mantener la capacidad de acceder fácilmente a cualquier descripción a partir de su clave, pero al mismo tiempo utilizar una cantidad razonable de memoria. Estas son dos de las propiedades que posee una **Tabla Hash**.

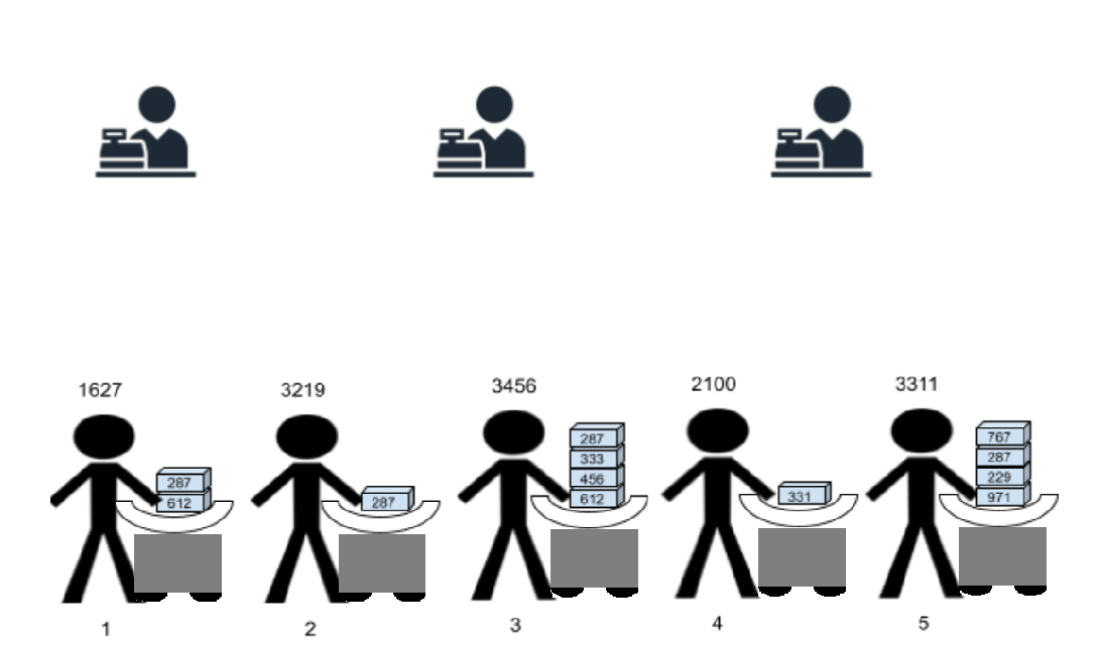
Por otro lado, es importante mencionar y explicar un poco en donde se va a encontrar contenida toda la información de la aplicación.

**Github:** Github es un portal creado para alojar el código de las aplicaciones de cualquier desarrollador, la plataforma está creada para que los desarrolladores suban el código de sus aplicaciones y herramientas, y que, como usuario no sólo puedas descargarte la aplicación, sino también entrar a su perfil para leer sobre ella o colaborar con su desarrollo. Git es uno de estos sistemas de control, que permite comparar el código de un archivo para ver las diferencias entre las versiones, restaurar versiones antiguas si algo sale mal, y fusionar los cambios de distintas versiones. También permite trabajar con distintas ramas de un proyecto, como la de desarrollo para meter nuevas funciones al programa o la de producción para depurar los bugs.

Así pues, Github es un portal para gestionar las aplicaciones que utilizan el sistema Git. Además de permitirte mirar el código y descargarte las diferentes versiones de una aplicación, la plataforma también hace las veces de red social conectando desarrolladores con usuarios para que estos puedan colaborar mejorando la aplicación.

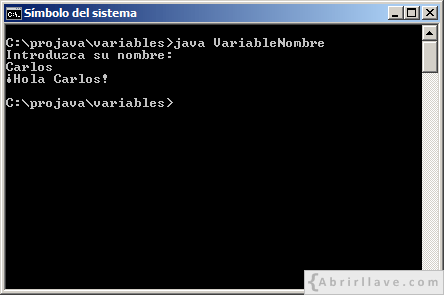
**Paso 3: Búsqueda de soluciones Creativas**

Alternativa 1. Programa de simulación (Con interfaz gráfica)



Se trata de un programa hecho en Java que simula la entrada y salida de los clientes a la tienda de videojuegos. Usando estructuras de datos tales como Colas (simulando la fila de clientes), Stacks (simulando la canasta y el carrito de los usuarios) y Hash Tables (simulando las estanterías y listas de los clientes) se puede llegar a una simulación aproximada del orden de salida de los clientes a la tienda basados en la cantidad de juegos, orden de llegada y tiempos estimados en la tienda de cada uno de los clientes.Con una interfaz que permita ver de manera aproximada los movimientos que realizaría el cliente por cada sección gráficamente ejemplificando la experiencia real.

Alternativa 2. Programa de simulación (Por Consola)



Se trata de un programa hecho en Java, con entrada por consola (de texto plano), que simula la entrada y salida de los clientes a la tienda de videojuegos. Usando estructuras de datos tales como Colas (simulando la fila de clientes), Stacks (simulando la canasta y el carrito de los usuarios) y Hash Tables (simulando las estanterías y listas de los clientes) se puede llegar a una simulación aproximada del orden de salida de los clientes a la tienda basados en la cantidad de juegos, orden de llegada y tiempos estimados en la tienda de cada uno de los clientes.

Alternativa 3. Programa de simulación con vista 3D de la tienda.



Se trata de un programa hecho en Java que simula, con una visualización 3D (en realidad aumentada), la entrada y salida de los clientes a la tienda de videojuegos. Usando estructuras de datos tales como Colas (simulando la fila de clientes), Stacks (simulando la canasta y el carrito de los usuarios) y Hash Tables (simulando las estanterías y listas de los clientes) se puede llegar a una simulación aproximada del orden de salida de los clientes a la tienda basados en la cantidad de juegos, orden de llegada y tiempos estimados en la tienda de cada uno de los clientes.Con una interfaz que permita que el usuario pueda ver de manera aproximada y en primera persona, los movimientos que realizaría por cada sección gráficamente ejemplificando la experiencia real de ir a la tienda con el nuevo e innovador sistema de atención.

**Paso 4: Transición de las Ideas a los Diseños Preliminares**

En este paso, descartamos las alternativas que son menos factibles. Por ello, descartamos la alternativa 3. Programa de simulación con vista 3D de la tienda, ya que por la dificultad que supone la simulación en 3D, su implementación puede ser imprecisa y no proveer una óptima solución al problema.

La revisión de las otras alternativas nos lleva a lo siguiente:

Alternativa 1. Programa de simulación (Con interfaz gráfica)

Alternativa 2. Programa de simulación (Por Consola)

**Paso 5: Evaluación y Selección De La Mejor Solución**

* Criterio A: La alternativa es amigable con el usuario. La alternativa entrega una solución:
  + [2] Amigable
  + [1] Poco amigable
* Criterio B: La alternativa grafica de manera entendible la simulación. Entrega una solución:
  + [3] Completamente entendible
  + [2] Medianamente entendible
  + [1] Poco entendible

Se dejan por fuera los criterios de funcionalidad ya que ambas soluciones implementan los mismos métodos para simular

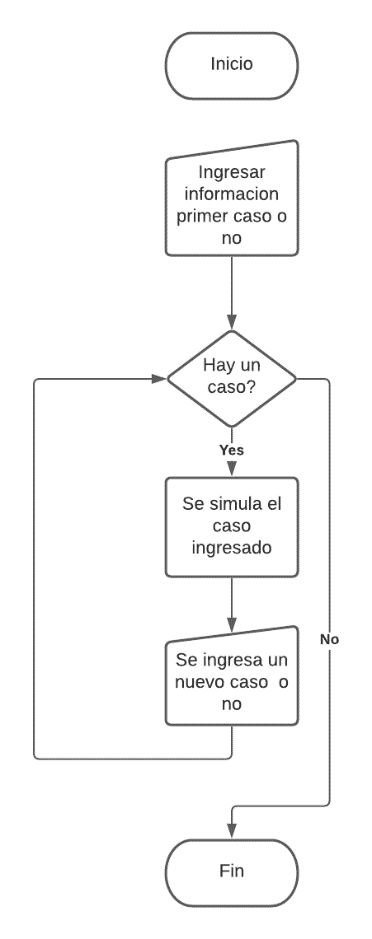
|  |  |  |  |
| --- | --- | --- | --- |
|  | Criterio A | Criterio B | Total |
| Alternativa 1. Interfaz grafica | Amigable  2 | Completamente entendible  3 | 5 |
| Alternativa 2. Sin interfaz grafica | Poco amigable  1 | Poco entendible  1 | 2 |

Selección

De acuerdo con la evaluación de criterios anterior, se debe implementar la simulación con interfaz gráfica para hacer más entendible la simulación para el usuario.

**Paso 6: Preparación de Informes y Especificaciones**

Los documentos de especificación, diseño y requerimientos que posee el programa a implementar se encuentran en la carpeta docs del repositorio de github donde está almacenado el proyecto. (Inventen un pseudocodigo con ese diagrama)



**Paso 7: Implementación del Diseño**

La implementación se encuentra en el repositorio de github, está realizado en código Java con Javafx.