Logotipo

Descripción generada automáticamente

McDonald’s (simulador)

Proyecto final de programación avanzada

**ANDRÉS FLORES SANTAMARÍA & DIANA LAURA CRISTINO MORALES**

Facultad de estadística e informática,  UNIVERSIDAD VERACRUZANA

**Contenido**

[**Introducción** 3](#_Toc74511993)

[**Objetivo del documento** 3](#_Toc74511994)

[**Descripción de la propuesta de solución** 3](#_Toc74511995)

[**Descripción de la aplicación** 4](#_Toc74511996)

[**Casos de uso** 4](#_Toc74511997)

[**Requerimientos funcionales** 5](#_Toc74511998)

[**Requerimientos no funcionales** 5](#_Toc74511999)

[**Diagrama de clases** 6](#_Toc74512000)

[**Diagrama de secuencia** 7](#_Toc74512001)

[**Diagrama de componentes** 7](#_Toc74512002)

[**Especificación de casos de uso** 8](#_Toc74512003)

[**CU-01 Agregar un cliente** 8](#_Toc74512004)

[**CU-02 Consultar ventas** 9](#_Toc74512005)

[**CU-03 Cancelar venta** 10](#_Toc74512006)

[**Metodología de Desarrollo** 10](#_Toc74512007)

[**Herramientas** 11](#_Toc74512008)

[**Lenguajes y Plataformas** 12](#_Toc74512009)

[**Plan de pruebas** 13](#_Toc74512010)

[**Descripción de los participantes** 13](#_Toc74512011)

[**Descripción de herramientas de medición** 13](#_Toc74512012)

[**Elementos de pruebas** 14](#_Toc74512013)

[**Enfoque de pruebas (estrategia)** 14](#_Toc74512014)

[**Criterios de aceptación o rechazo** 14](#_Toc74512015)

[**Criterios de suspensión** 14](#_Toc74512016)

[**Requerimientos de entornos – Hardware** 14](#_Toc74512017)

[**Requerimientos de entornos – Software** 15](#_Toc74512018)

[**Resultados de las pruebas** 16](#_Toc74512019)

[**Evidencias de las pruebas** 22](#_Toc74512020)

[**Conclusiones** 25](#_Toc74512021)

[**Referencias** 26](#_Toc74512022)

**Tabla de ilustraciones**

[1 - Agregando un cliente 22](#_Toc74507914)

[2 - Agregando 3 clientes 22](#_Toc74507915)

[3 - Agregando 10 clientes 23](#_Toc74507916)

[4 - Agregar 25 clientes 23](#_Toc74507917)

[5 - Verificando que los cocineros preparen las ordenes 24](#_Toc74507918)

[6 - Verificando el total acumulado de las ventas 24](#_Toc74507919)

# **Introducción**

## **Objetivo del documento**

El presente proyecto denominado "Simulador McDonald’s", desarrollado por estudiantes de la experiencia educativa Programación Avanzada de la carrera en Tecnologías Computacionales de la Facultad de Estadística e Informática, tiene como finalidad simular el modelo de trabajo de la cadena de restaurantes “McDonald’s”, ya que la rúbrica del proyecto final tiene como requisito el simular un mecanismo libre de control de comunicación entre un productor (empleados de McDonald´s) consumidor (Clientes que acuden al restaurante) de un producto definido. Este proyecto sirve para evaluar al estudiante, pero también para desarrollar habilidades de programación de los diferentes temas vistos durante el curso. Cabe mencionar que es solo una simulación para el ámbito educativo y no se tiene relación alguna con la cadena de restaurantes McDonald´s ®.

Por todo lo mencionado anteriormente es que surge el presente documento donde se explican, las herramientas, objetivos, plataformas y sistemas operativos en los que está basada la aplicación, así como los usuarios que se verán beneficiados ya sea de forma directa o indirectamente.

# **Descripción de la propuesta de solución**

Este simulador muestra el funcionamiento básico del despacho de ordenes en un restaurante de comida rápida. En el mostrador hay 3 cajeros que es el número de cajeros que habitualmente están (en la vida real). Los cocineros (por experiencia) pueden estar o no en la cocina, mientras no haya una orden, el cocinero puede estar lavando trastes, trayendo papas, carne, o pan, leche para helado, salsas del refrigerador o estar arreglando el almacén, pero cuando un cliente se acerque al mostrador y haga un pedido, el cocinero debe estar ahí para cocinar y despachar lo que el cliente pidió de la manera más rápida posible. Cuando haya más de un cliente se necesitan más cocineros en la cocina para generar los productos sin demorar tanto.

En este simulador se muestra:

* El momento del tiempo de producción, que es cuando el cocinero entra en acción después de que un cliente llega al cajero.
* El momento en donde el consumidor adquiere el producto, que es cuando paga y entonces desaparece del mostrador (porque se va)
* El monto (de dinero) que el cliente debe pagar que aparece en la caja registradora.
* El monto total de ventas del día.
* Los clientes que están en espera y los que están en el mostrador.

# **Descripción de la aplicación**

Para cumplir con los requisitos que exige la experiencia educativa correspondiente se optó por diseñar la aplicación con el lenguaje de programación Java, desarrollada con ayuda de la función “Design” de NetBeans para facilitar en gran medida el trabajo y para obtener una interfaz gráfica más limpia y agradable a la vista. La aplicación cuenta con una botón para generar un cliente, cada vez que se oprima ese botón se generará un cliente que pasará a espera y si no hay clientes en el mostrador, pasará directamente a pedir su orden al mostrador. Una vez se haya tomado la orden, el cocinero empieza a preparar la orden y una vez pagada la cantidad que aparece en la caja registradora, se le entrega la orden al cliente para dar paso a los demás. Ya que el espacio es pequeño se optó por agregar un Jlist donde se muestren los clientes que están “formados” en la fila para poder pedir su orden.

# **Casos de uso**

* Agregar un cliente
* Consultar ventas
* Cancelar venta

# **Requerimientos funcionales**

* El sistema debe contener elementos gráficos
* El sistema debe mostrar la interacción que se da entre el productor y consumidor
* El sistema debe contar con suficiente espacio para agregar clientes sin que colapse.
* El sistema debe mostrar al cliente el precio a pagar.
* El sistema debe llevar el conteo de todo lo que se ha vendido en el día.
* El sistema debe ser capaz de guardar y mostrar las ventas hechas.
* El sistema debe mostrar el momento del tiempo de producción.
* El sistema debe mostrar el momento en donde el consumidor adquiere el producto

# **Requerimientos no funcionales**

* Un sistema eficiente
* Tolerante a fallas
* Un sistema eficaz

# **Diagrama de clases**

Diagrama

Descripción generada automáticamente

En este diagrama mostramos las clase las cuales están involucradas con nuestro sistema en las que se muestran las funciones que realizan y su relación entre cada una de estas

# **Diagrama de secuencia**

Captura de pantalla de un celular

Descripción generada automáticamente

En este diagrama se muestras cómo es la interacción de intercambio de datos entre el usuario que utiliza la aplicación e interacciona con la interfaz de la misma, y cómo esta guarda y consulta la información en la base de datos.

# **Diagrama de componentes**

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja

Nos permite visualizar cuales son los componentes y las dependencias del sistema.

# **Especificación de casos de uso**

## **CU-01 Agregar un cliente**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nombre del caso de uso** | Agregar un cliente | **Código** | CU-01 |
| **Actor(es)** | Usuario | **Fecha** |  |
| **Descripción** | El usuario agregara al cliente en turno del cual va realizar su pedido | | |
| **Precondición** | A ver registrado el pedido del cliente anterior | | |
| **Postcondición** | Se realizo el pedido del cliente el cual estaba en turno | | |
| **Flujo Básico**   1. Abrir la aplicación McDonald’s 2. Atender el cliente en turno 3. Registra el pedido en la aplicación 4. Guardar el ´pedido el cual realizo el cliente 5. Se guarda la información en la base de datos | | | |
| **Flujo Alterno** | | | |

## **CU-02 Consultar ventas**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nombre del caso de uso** | Consultar una venta | **Código** | CU-02 |
| **Actor(es)** | Usuario | **Fecha** |  |
| **Descripción** | El usuario podrá consultar todas las ventas realizadas en la aplicación. | | |
| **Precondición** | Que los usuarios hayan realizados pedidos en el sistema y se hayan registro en la base de datos | | |
| **Postcondición** | Se mostrará la información que se registró en la base de datos | | |
| **Flujo Básico**   1. El usuario deberá entrar a la aplicación 2. El usuario debe dirigirse a la sección de ventas 3. El sistema deseplegará todas la venta realizadas | | | |
| **Flujo Alterno**   1. El sistema podrá mostrar que no se ha realizado ninguna venta | | | |

## **CU-03 Cancelar venta**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nombre del caso de uso** | Cancelar una venta | **Código** | CU-03 |
| **Actor(es)** | Usuario | **Fecha** |  |
| **Descripción** | En el caso que el cliente no quiera lo que pido el usuario podrá cancelar la venta | | |
| **Precondición** | El usuario debe haber realizado una venta | | |
| **Postcondición** | Se borrará la venta de la base de datos | | |
| **Flujo Básico**   1. El usuario debe ingresar en el sistema 2. El usuario buscara la venta la cual quiera cancelar 3. El sistema cancelara la venta y mandara un mensaje de respuesta | | | |
| **Flujo Alterno** | | | |

# **Metodología de Desarrollo**

Scrum es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos.

En Scrum se realizan entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto. Por ello, Scrum está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados pronto, donde los requisitos son cambiantes o poco definidos, donde la innovación, la competitividad, la flexibilidad y la productividad son fundamentales.

Scrum también se utiliza para resolver situaciones en que no se está entregando al cliente lo que necesita, cuando las entregas se alargan demasiado, los costes se disparan o la calidad no es aceptable, cuando se necesita capacidad de reacción ante la competencia, cuando la moral de los equipos es baja y la rotación alta, cuando es necesario identificar y solucionar ineficiencias sistemáticamente o cuando se quiere trabajar utilizando un proceso especializado en el desarrollo de producto.

Estas características de Scrum hacen una metodología ideal para este tipo de proyecto que está en constante cambio y que durante el desarrollo se necesitan de algunas funcionalidades para el sistema, cambiando las prioridades de otras tareas que se estén desarrollando o que estén próximas a desarrollarse, esto hace que se puedan cambiar las actividades conforme a las necesidades y el equipo de desarrollo (nosotros) sea capaz de adaptarse al cambio de manera rápida mostrando resultados inmediatos con las funcionalidades esperadas.

# **Herramientas**

**Java plataforma**

Es un lenguaje de programación y una plataforma informática comercializada por primera vez en 1995 por Sun Microsystems. Hay muchas aplicaciones y sitios web que no funcionarán a menos que tenga Java instalado y cada día se crean más. Java es rápido, seguro y fiable. Desde portátiles hasta centros de datos, desde consolas para juegos hasta súper computadoras, desde teléfonos móviles hasta Internet, Java está en todas partes.

**NetBeans**

Es un [entorno de desarrollo integrado](https://es.wikipedia.org/wiki/Entorno_de_desarrollo_integrado) [libre](https://es.wikipedia.org/wiki/Software_libre), hecho principalmente para el [lenguaje de programación](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n) [Java](https://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n)). Existe además un número importante de módulos para extenderlo. NetBeans IDE[1](https://es.wikipedia.org/wiki/NetBeans#cite_note-1) es un producto libre y gratuito sin restricciones de uso.

NetBeans es un proyecto de [código abierto](https://es.wikipedia.org/wiki/C%C3%B3digo_abierto) de gran éxito con una gran base de usuarios, una comunidad en constante crecimiento. [Sun Microsystems](https://es.wikipedia.org/wiki/Sun_Microsystems) fundó el proyecto de código abierto NetBeans en junio de 2000 y continúa siendo el patrocinador principal de los proyectos. Actualmente Sun Microsystems es administrado por [Oracle Corporation](https://es.wikipedia.org/wiki/Oracle_Corporation).

**Variables de entorno**

PATH es la variable del sistema que utiliza el sistema operativo para buscar los ejecutables necesarios desde la línea de comandos o la ventana Terminal.

La variable del sistema PATH se puede establecer utilizando la utilidad de sistema en el panel de control de Windows o en el archivo de inicio del shell en Linux y Solaris.

Normalmente no es necesario realizar cambios en la variable PATH del sistema en computadoras con Windows o Mac OS X.

## **Lenguajes y Plataformas**

**Java Lenguaje de programación**

Es un lenguaje de programación creado por Sun Microsystems, (empresa que posteriormente fue comprada por Oracle) para poder funcionar en distintos tipos de procesadores. Su sintaxis es muy parecida a la de C o C++, e incorpora como propias algunas características que en otros lenguajes son extensiones: gestión de hilos, ejecución remota, etc.

El código Java, una vez compilado, puede llevarse sin modificación alguna sobre

cualquier máquina, y ejecutarlo. Esto se debe a que el código se ejecuta sobre una máquina hipotética o virtual, la Java Virtual Machine, que se encarga de interpretar el código (ficheros compilados .class) y convertirlo a código particular de la CPU que se esté utilizando (siempre que se soporte dicha máquina virtual).

# **Plan de pruebas**

El Plan de pruebas aquí presentado se realizó con el propósito de identificar posibles fallas y errores de nuestra aplicación, también uno de los propósitos es verificar el correcto funcionamiento de los casos de uso implementados. De manera detallada podemos resaltar que la prueba se realizó a los casos de uso incluidos en el sistema, por cada caso de uso se realizó una prueba y en su gran mayoría los resultados obtenidos fueron exitosos, a excepción de las funciones que aún faltan.

## **Descripción de los participantes**

Debido a la contingencia que pasamos actualmente, las pruebas fueron realizadas por los mismos programadores y a distancia. Aquí se probó el correcto funcionamiento de las acciones de la aplicación para ver que hiciera lo que tenía que hacer y que no se cerrara por un error. De las observaciones realizadas se puede mencionar que los casos de uso programados tuvieron un total de 80 % de aceptación, ya que las pruebas finales no se pudieron realizar conforme lo estipulado, esto por la falta de funcionalidades que presentó la aplicación. Las observaciones se detallan a continuación:

Desarrollador 1: Andres flores santamaria - Se mostró una gran mejoría al momento de agregar muchos clientes, ya que anteriormente no se sabía cuantos clientes había en espera y algunas veces se quedaba “trabada la aplicación” con las imágenes estáticas.

Desarrollador 2: Diana Laura Cristino Morales- La implementación de la lista para los clientes en fila mejoró la manera de ver las filas de los clientes, los contadores de dinero funcionan correctamente.

## **Descripción de herramientas de medición**

Para realizar las pruebas estipuladas con anterioridad se compartió el mismo código fuente a los dos programadores y se necesitó de tener NetBeans Instalado para poder correr la aplicación. Con anterioridad ya se debía tener Java y las variables de entorno instaladas en el sistema

## **Elementos de pruebas**

Acciones que se probaron de la aplicación Simulador McDonald’s:

* Agregar un cliente
* Agregar 3 clientes de manera continua
* Agregar 10 clientes de manera continua
* Agregar 25 clientes de manera contínua
* Verificar producción de cocinero
* Sumar el monto total de las ventas

**Nuevas funcionalidades para probar:**

* Consultar las ventas
* Cancelar una venta

## **Enfoque de pruebas (estrategia)**

Todas las pruebas realizadas fueron hechas para los requerimientos funcionales del sistema, podemos decir que son pruebas funcionales.

## **Criterios de aceptación o rechazo**

Al realizarse pruebas unitarias, el criterio de aceptación es que la prueba quede un 100% concluida con éxito, sin fallos ni errores y con los resultados esperados, según los casos de uso.

## **Criterios de suspensión**

En caso de no contar con las funcionalidades listas para probar se procede a suspender la prueba.

## **Requerimientos de entornos – Hardware**

* Computadora de escritorio o laptop
* Procesador Intel Pentium III 1 GHz o superior
* RAM : 2 GB o superior
* 2 Mb de espacio libre en disco

## **Requerimientos de entornos – Software**

* Windows 10, Windows 8.1, Windows 7, Windows XP, Windows Vista.
* Java
* Variables de entorno Java
* NetBeans IDE 8.2

## **Resultados de las pruebas**

|  |  |
| --- | --- |
| **ID Caso de Prueba** | **CP-01** |
| **Fecha** |  |
| **Título** | Agregar un cliente |
| **Descripción** | El usuario que utiliza la aplicación agrega un cliente |
| **Condiciones de Entrada** | El usuario debe haber ingresado a la aplicación y haber hecho clic una vez en el botón “Comprar”. |
| **Resultado Esperado** | La aplicación muestra un cliente realizando una orden con el cajero 1, el cocinero de dicha caja se pone a preparar la orden y al finalizar el cliente paga y el cocinero desaparece.  Los estados del cajero cambian de “en espera” a “Atendiendo” y después a “orden entregada” |
| **Resultado Obtenido** | La aplicación mostró al cliente pidiendo la orden con el cajero 1, en seguida el cocinero apareció preparando una hamburguesa, los estados de la orden cambiaron de “En espera” a “Atendiendo” y después a “orden entregada” por lo que la prueba pasó con éxito. |

|  |  |
| --- | --- |
| **ID Caso de Prueba** | **CP-02** |
| **Fecha** |  |
| **Título** | Agregar tres clientes |
| **Descripción** | El usuario que utiliza la aplicación agrega 3 clientes de manera consecutiva. |
| **Condiciones de Entrada** | El usuario da tres clics consecutivos en menos de 1 segundo en el botón comprar. |
| **Resultado Esperado** | La aplicación muestra 3 clientes realizando una orden con el cajero 1, cajero 2 y cajero 3, respectivamente. Ya que la demanda es alta, los tres cocineros de se ponen a preparar las órdenes y al finalizar los clientes pagan y los cocineros desaparecen.  Los estados de los tres cajeros cambian de “en espera” a “Atendiendo” y después a “orden entregada” |
| **Resultado Obtenido** | La aplicación muestró 3 clientes realizando una orden con los 3 diferentes cajeros. Los tres cocineros de se aparecieron para preparar las órdenes y al finalizar los clientes pagaron el monto y los cocineros desaparecieron.  Los estados de los tres cajeros cambiaron de “en espera” a “Atendiendo” y después a “orden entregada” conforma iban realizando las actividades |

|  |  |
| --- | --- |
| **ID Caso de Prueba** | **CP-03** |
| **Fecha** |  |
| **Título** | Agregar diez clientes |
| **Descripción** | El usuario que utiliza la aplicación agrega diez clientes de manera consecutiva. |
| **Condiciones de Entrada** | El usuario da diez clics consecutivos en menos de 3 segundos en el botón comprar. |
| **Resultado Esperado** | La aplicación muestra 3 clientes realizando una orden con el cajero 1, cajero 2 y cajero 3, respectivamente. Ya que la demanda es alta, los tres cocineros de se ponen a preparar las órdenes  Hay otros 4 clientes haciendo fila atrás de los que están pidiendo su orden.  En la lista que está implementada aparecen 3 líneas de texto diciendo “cliente en espera”. Conforme van despachando las órdenes los trabajadores, primero desaparecen las líneas de texto en la lista y van desapareciendo los clientes en espera conforme se van retirando los clientes con sus órdenes. |
| **Resultado Obtenido** | La aplicación mostró 3 clientes realizando una orden con el cajero 1, cajero 2 y cajero 3, respectivamente. Aparecieron los tres cocineros a preparar las órdenes  Había otros 4 clientes haciendo fila atrás de los que estaban pidiendo su orden.  En la lista que está implementada aparecieron 3 líneas de texto diciendo “cliente en espera” y fueron desapareciendo conforme los clientes se retiraban con sus órdenes.  Después de desaparecer las líneas de texto empezaron a desaparecer los clientes que estaban en la fila, pues desaparecían porque pasaban a pedir su orden. |

|  |  |
| --- | --- |
| **ID Caso de Prueba** | **CP-04** |
| **Fecha** |  |
| **Título** | Agregar veinticinco clientes |
| **Descripción** | El usuario que utiliza la aplicación agrega veinticinco clientes de manera consecutiva. |
| **Condiciones de Entrada** | El usuario da veinticinco clics consecutivos en menos de 6 segundos en el botón comprar. |
| **Resultado Esperado** | La aplicación muestra una gran serie de clientes haciendo su pedido, otros 4 más en la fila y los demás en la lista, la cual muestra un Scroll para poder visualizar todos los clientes en texto. Los cocineros hacen su trabajo de manera habitual. |
| **Resultado Obtenido** | La aplicación mostró los 25 clientes, tres pidiendo sus órdenes, otros 4 más en la fila y los demás en la lista, la cual mostró un Scroll para poder visualizar todos los clientes en texto. Los cocineros trabajaron de manera correcta. |

|  |  |
| --- | --- |
| **ID Caso de Prueba** | **CP-05** |
| **Fecha** |  |
| **Título** | Verificar producción de cocinero |
| **Descripción** | El usuario que utiliza la aplicación verifica que cada vez que aparezca un cliente en el mostrador, el cocinero cocine su orden. |
| **Condiciones de Entrada** | El usuario tiene 1 minuto para hacer 5 clics en el botón comprar ya sea de manera consecutiva o de manera espontánea. |
| **Resultado Esperado** | Cada que un cliente aparezca en mostrador, un cocinero debe aparecer para producir la orden. Cuando el estado del pedido sea “orden entregada” y el cliente se vaya, el cocinero puede desaparecer. |
| **Resultado Obtenido** | Los cocineros aparecieron cada que un cliente aparecía y desaparecían cada que el cliente se iba, solo que en algunos momentos desaparecían de la nada los clientes y el cocinero, aún cuando el estado era “Atendiendo”. Esto después de haber hecho las pruebas de los 25 clientes.  Se cree que hubo una sobrecarga de memoria por lo cuál pasó esto. Así que se prosiguió a reiniciar la PC y se volvió a hacer la prueba. Todo corrió con normalidad. |

|  |  |
| --- | --- |
| **ID Caso de Prueba** | **CP-06** |
| **Fecha** |  |
| **Título** | Sumar el monto total de ventas |
| **Descripción** | El usuario verifica que las ventas del día sean la suma de todos los cajeros. |
| **Condiciones de Entrada** | Haber hecho por lo menos 2 ordenes y haberlas terminado con éxito. |
| **Resultado Esperado** | El usuario que utiliza la aplicación tiene medio minuto para agregar los clientes que sea. Y al terminar de despachar todas las órdenes, los montos que tienen cada cajero, la suma debe ser igual al monto mostrado en el apartado “Ventas del día” |
| **Resultado Obtenido** | Al termino de todas las órdenes, las cuales fueron 13, se mostró en “ventas del día” la cantidad de los 3 cajeros sumada. |

## **Evidencias de las pruebas**

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

1 - Agregando un cliente

Una captura de pantalla de una red social

Descripción generada automáticamente

2 - Agregando 3 clientes

Una captura de pantalla de una red social

Descripción generada automáticamente

3 - Agregando 10 clientes

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

- Agregar 25 clientes

Interfaz de usuario gráfica

Descripción generada automáticamente

5 - Verificando que los cocineros preparen las ordenes

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

6 - Verificando el total acumulado de las ventas

# **Conclusiones**

El trabajar en este sistema nos dimos cuenta como esta realizadas las aplicaciones de productor consumidor con esto debemos recalcar que son de ayuda porque nos ayudan a automatizar proceso los cuales nos llevaría mucho tiempo realizarlos de manera manual y nos permite almacenar datos los cuales serán de ayuda más adelante para realizar evaluaciones de todos los datos guardados .

Además de que nos sirvió de repaso para recordar todo lo que vimos en el curso y poder integrar todas las tecnologías en una sola aplicación fue un gran reto, pero nos quedamos con el aprendizaje adquirido.

# **Referencias**

*Java*. (s. f.). java. Recuperado 13 de junio de 2021, de <https://www.java.com/es/download/help/whatis_java.html>

*Wikipedia*. (s. f.). Wikipedia. Recuperado 13 de junio de 2021, de <https://es.wikipedia.org/wiki/NetBeans>

*Lenguaje java*. (s. f.). lenguaje. Recuperado 13 de junio de 2021, de <https://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n>)

*Aplicacion de escritorio. (s. f.). wikipedia. Recuperado 13 de junio de 2021,* [*https://es.wikipedia.org/wiki/Aplicaci%C3%B3n\_de\_escritorio#:~:text=Una%20aplicaci%C3%B3n%20de%20escritorio%20es,de%20la%20red%20o%20internet*](https://es.wikipedia.org/wiki/Aplicaci%C3%B3n_de_escritorio#:~:text=Una%20aplicaci%C3%B3n%20de%20escritorio%20es,de%20la%20red%20o%20internet)