Inicio

**UNIVERSIDAD TECNOLÓGICA DEL PERÚ**

FACULTAD DE INGENIERÍA

INGENIERÍA DE SISTEMAS E INFORMÁTICA

**DESARROLLO DE UN APLICATIVO PARA CALCULAR LA CANTIDAD DE HORAS AUMENTAR A CADA OPERARIO DE PRODUCCIÓN DE PALTA EN LA PROCESADORA TORRE BLANCA.**

DOCENTE: NUÑEZ TARRILLO ERICK RAMON

CURSO DE INTEGRADOR I PARA SOFTWARE Y SISTEMAS

ALUMNA: SAMARITANO AYALA LESLY VERONICA

1628112

**LIMA-PERÚ**

**2019**

**CONTENIDO**

[**1.** **CAPITULO I** 3](#_Toc9437288)

[**ASPECTOS GENERALES** 3](#_Toc9437289)

[**1.1 Definición del problema.** 3](#_Toc9437290)

[**1.2** **Definición de objetivos.** 5](#_Toc9437291)

[**1.3** **Alcances y limitaciones.** 6](#_Toc9437292)

[**1.4** **Justificación.** 6](#_Toc9437293)

[**1.5** **Estado del arte.** 7](#_Toc9437294)

[**2.** **CAPITULO II** 7](#_Toc9437295)

[**MARCO TEÓRICO** 7](#_Toc9437296)

[**2.1** **Fundamento Teórico.** 7](#_Toc9437297)

[**3.** **CAPITULO III** 12](#_Toc9437298)

[**DESARROLLO DE LA SOLUCIÓN** 12](#_Toc9437299)

[**3.1 Descripción a la solución.** 12](#_Toc9437300)

[**3.2 Diagrama de clases.** 18](#_Toc9437301)

[**3.3 Estructura del proyecto.** 18](#_Toc9437302)

[**3.4 Contenido del proyecto.** 21](#_Toc9437303)

[**3.5 Implementación.** 24](#_Toc9437304)

[**4.** **CAPITULO IV** 24](#_Toc9437305)

[**RESULTADOS** 24](#_Toc9437306)

[**4.1 Prueba del aplicativo.** 24](#_Toc9437307)

[**4.2 Resultados Esperados.** 25](#_Toc9437308)

[**4.3 Pruebas del Aplicativo.** 27](#_Toc9437309)

[**4.4 Resultados según objetivos.** 28](#_Toc9437310)

[**4.5** **Costos y Beneficios.** 28](#_Toc9437311)

[**5.** **CAPITULO V** 30](#_Toc9437312)

[**CONCLUSIONES Y RECOMENDACIONES** 30](#_Toc9437313)

[**5.1 Conclusiones.** 30](#_Toc9437314)

[**5.1 Recomendaciones.** 30](#_Toc9437315)

[**6.** **REFERENCIAS BIBLIOGRÁFICAS** 31](#_Toc9437316)

[**7.** **ANEXOS** 31](#_Toc9437317)

[**ANEXO 1: LEAN CANVAS** 31](#_Toc9437318)

[**ANEXO 2: TABLA DE LAS VARIABLES DE LA CLASE PEDIDO** 33](#_Toc9437319)

# **CAPITULO I**

## **ASPECTOS GENERALES**

### **1.1 Definición del problema.**

En la procesadora de frutas y hortalizas Torre Blanca todos los días se presentan cartas de renuncias al área de Recursos Humanos. Las renuncias son de un intervalo entre 5-8 renuncias por la línea operativa, siendo el de mayor cantidad el área de empaquetado con intervalo de 3-5 renuncias por día. Esto se debe al sobretiempo que realizan los operarios diariamente y a una distribución no óptima para los procesos. A la falta de trabajadores y una correcta distribución de horas en la planta se ocasiona tardanzas en entregar las cajas de frutas requeridas, así como los plazos de entrega son largos cuando más cajas se solicitan.

#### **Descripción del problema**.

El problema que detallar es que se ocasionan renuncias por parte de los trabajadores en dicha procesadora Torre Blanca siendo el de mayor porcentaje el área de empaquetado. Las renuncias en toda la línea operativa se deben a las horas extras que se exige y extiende a lo largo del día. Es decir, si el supervisor de planta observa que no se va a finalizar con el número de cajas de las frutas (paltas) requeridas, prolonga el tiempo a todos los operarios de la línea operativa. Y esto no se encuentra establecida en el contrato, se indica en ella que son 8 horas diarias, pero se sale de la procesadora de finalizar sus labores 4 horas después, siendo la cantidad real 12 horas de la jornada laboral; si aun así no se termina con las cajas de paltas pedidas, entonces en ese momento las horas que se trabajan en adelante se considera “extras” y se remunera con 105% en cada hora normal. Por ende, al no contar con la cantidad requerida mínima de operarios en los procesos de recepción, selección, empaquetado y pesado se generan retrasos en entregar las cajas de frutas solicitadas.

Por otra parte, las renuncias también se dan a una mala distribución de horas totales a los operarios, ya que los días son más largos cuando la cantidad de cajas es mayor, y no se busca minimizar los días y dependiendo de ello contratar más personal en un área o aumentar horas extras en toda la línea operativa.

Los procesos que se toma en cuenta con sus respectivos detalles a continuación, cantidad mínima de personal en dichos procesos son 2 en el de recepción más un supervisor, 40 en selección, 40 en empaquetado más un supervisor, 6 en pesado de las cajas de frutas y un supervisor general de toda la línea. En el pesado el personal que se requiera dependerá de la cantidad de fajas habilitadas, son 9 fajas con las que dispone esta procesadora. En el caso de los retrasos es el doble de tiempo por operario, lo requerido es: 2 min en la recepción de paltas, 1 min en seleccionarlas se separa según a la madurez de la fruta (verde, semi-verde, o por madurar), 1 min en empaquetar en las cajas 6 u 8 frutas en cada una de ellas, varía según el tamaño calibrada, y por último 1 min en pesar las cajas como registrarlas. En conclusión, no se cumple con el objetivo de entregar las frutas demandas en esta procesadora.

### **Definición de objetivos.**

#### **Objetivos generales**.

* Implementar un aplicativo que calcule la cantidad de horas extras aumentar a cada operario en los procesos de recepción, empaquetado, selección y pesado, según a la cantidad de pedido de cajas para la exportación e importación de la palta en la procesadora Torre Blanca con la menor cantidad de días posible.

#### **Objetivos específicos.**

* Desarrollar un aplicativo que calcule el tiempo total que requiere una cantidad de cajas solicitadas con respecto a la variabilidad que hay en el proceso de recepción, selección, empaquetado y pesado de la procesadora aplicando la determinación del tamaño de muestra cuando se conoce la población con estadística inferencial.
* Calcular las horas que se tiene que aumentar a cada operario que se tiene en ese instante para la línea operativa. Para ello, se necesitan como datos la cantidad máxima de minutos que realiza el operario, así como el número de trabajadores que se demande. Y la población de la cantidad de cajas producidas en un día.

### **Alcances y limitaciones.**

Este aplicativo al desarrollarse tendrá como alcances que, en primer lugar, el código está orientado a la programación de objetos permitiendo de esta manera la reutilización. En segundo lugar, calculará el tiempo total que se requiere en la procesadora. Y, por último, el aplicativo será responsiva, se podrá usar ya sea en una computadora de escritorio, tablet o móvil. Por otro lado, las limitaciones que tiene este aplicativo son.

* El aplicativo estará diseñado para la cantidad mínima de 88 operarios en total de los procesos de recepción, selección, empaquetado y pesado.
* El aplicativo no calculara el tiempo que se requiere en cada uno de los procesos que se toma en cuenta de recepción, selección, empaquetado y pesado.
* El aplicativo no calculara los requerimientos que necesita cada operario ya que no cuenta con una base de datos de 88 operarios.

### **Justificación.**

Una herramienta tecnológica en una empresa, procesadora o cualquier otra entidad es una ventaja competitiva ya que, permite optimizar procesos, predecir resultados, determinar los valores de las variables para la mejora continua; en este caso determinar las horas extras de que se necita para entregar de cajas de fruta con la menor cantidad de días.

### **Estado del arte.**

En la actualidad existen softwares similares y con el mismo objetivo de optimizar el tiempo en todos los procesos del sector industrial, por ejemplo:

I68: ofrece producto y servicios con el objetivo de resolver la gestión de grupos empresariales, cubriendo de esa manera la gestión de la empresa aportando una visibilidad 360° y gestionando un dato único. Ofrece distintos softwares como de gestión ERP, gestión en el área de fabricación, gestión en el área de logística y en gestión de recursos humanos.

SCADA. - es un software basado en web que captura datos de operarios, sensores de máquina, PLCs (controladores lógicos programables) y varios dispositivos dentro de una planta, incluso en ubicaciones remotas. SCADA tiene la capacidad de presentar una planta 3D de todas las máquinas, operarios y procesos en tiempo real, lo que permite reducir el tiempo improductivo. (SisTrade, 2019)

# **CAPITULO II**

## **MARCO TEÓRICO**

### **Fundamento Teórico.**

#### **2.1.1 Las áreas de actividades recepción, selección, empaquetado y pesado.**

En primer lugar, se va detallar los procesos que se realizan en esta procesadora de frutas Torre Blanca que se encuentra ubicada en Chancay. Esta información a sido recopilado por un ex trabajador de este planta.

Las frutas que se procesan es el mango y la palta; para este proyecto se tomara en cuenta la palta. Asì mismo, tener en cuenta que dichos procesos para este trabajo solo son 4, el de recepciòn de frutas, selecciòn, empaquetado y pesado que acontinuaciòn se dictan: El proceso se incia cuando se recepcionan las cajas de platas, previamente en la madrugada ya se dejan las cajas para que en la mañana puedan ser recepcionados para el àrea de calidad. En el àrea de recepciòn se toman las cajas y se va colocando fruta por fruta en transportadora de cadena, este es una faja que lleva las frutas a los siguientes actividades a realizar, el tiempo maximo a realizar esta tarea es de 3 minutos. Luego, es llevado a la àrea de calidad en donde es lavado y desinfectado y nuevamente se colca en la faja transportadora. Una vez en la fajas los operarios seleccionan la palta, se elige entre semi maduro, maduro y verde. En dònde los verdes pasan a ser empaquetadas, los maduros y semi maduros son retirados para otra àrea. El siguiente actividad a realizar una vez que se tiene las paltas verdes, la calibradora elige las frutas según al tamaño, diviendolas en otras fajas transportadoreas. Ahora el siguiente actividad es empaquetar las paltas, en donde las paltas son colocadas en una caja entre 6 u 8 unidades, lo maximmo a realizar esta tarea es de 1 minuto. Y finalmente, se pesan las cajas para verificar si se posee el peso adecuado, aquì finaliza lo que es la lìnea operativa. En la planta hay supervisores uno en la recepciòn de frutas que se encarga de contabilizar las cajas ingresadas y enviadas a procesar; un supervisor en empaquetado que monitorea a los operarios, un supervisor en el pesado que se encaraga de verificar que las cajas pesadas tengan etiquetas, y un supervisor de toda la lìnea operativa quièn realiza los pedidos recibidos del ingeniero.

Por otro lado, para desarrollar este aplicativo que me calcule las horas extras aumentar a cada operario segùn a los pedidos que se realizan, primero se debe tener en cuenta la capacidad de operarios que se tiene en planilla. Para calcular estas operaciones se aplicara la determinación de tamaño de muestra con población conocida y la fración molar esto es posible gracias a laEstadística Inferencial esta información se obtiene de Violeta Alicia Nolberto Sifuentes y

María Estela Ponce Aruneri definen lo siguiente:

1. *La Estadística Inferencial.*

Rama de la Estadística que estudia el comportamiento y propiedades de las muestras y la posibilidad, y límites, de la generalización de los resultados obtenidos a partir de aquellas alas poblaciones que representan. Esta generalización de tipo inductivo, se basa en laprobabilidad.

1. *Población.*

Es la totalidad de elementos sobre los cuales recae la investigación. A cada elemento se le llama unidad estadística, ésta se le observa o se le somete a una experimentación.

1. *Muestra.*

Los autores hacen referencia Sierra Bravo (1991) anota que: “Una muestra en general, es toda parte representativa de lapoblación, cuyas características debe reproducir en pequeño lo más exactamente posible.” Para determinar el tamaño de muestra se usa la siguiente fórmula:

Ecuación 1: Tamaño de muestra

En dónde: es la varianza poblacional, Z la Z de la distribución normal

el error que se genera. Equivale a 0.05

1. *Fracción muestral.*

Es aquella proporción entre el tamaño de muestra y el tamaño de la poblacion elegida.

El autor Arrondo Manzano Vicente define lo que siginifica el error de precisión y el nivel de confianza. Y su fórmula es la siguiente:

Ecuación 2: Fórmula de fracción de muestra

En dónde n: es el tamaño de muestra y N es la población.

1. *Error de presición (E).*

Es la cantidad que se resta y se suma al estadístico o estimador para construir el intervalo de confianza en cuyo interior esperamos que se encuentre el parámetro. La máxima diferencia que cabe esperar que exista entre el estadístico y el parámetro, dada una confianza concreta. A mayor error de precisión, menos precisión (más imprecisión). Lo ideal es buscar la máxima precisión posible (el mínimo valor para el error de precisión).

1. *Nivel de confianza.*

Expresa la certeza de que realmente el dato que buscamos esté dentro del margen de error. Y además representa el porcentaje de intervalos que incluirían el parámetro de población.

Según la autora Mayra cueva Lopez define a la varianza poblacional como:

1. *Varianza poblacional.*

El análisis de varianza es una técnica que se puede utilizar para decidir si las medias de dos o más poblaciones son iguales.La prueba se basa en una muestra única,obtenida a partir de cada población.El análisis de varianza puede servir para determinar si las diferencias entre las medias muestrales revelan las verdaderas diferencias entre los valores medios de cada una de las poblaciones, osilas diferencias entre los valores medios de la muestra son más indicativas de una variabilidad de muestreo. Y su fórmula para hallar la varianza poblacional es la siguiente:

Ecuación 3: Varianza Poblacional

En dónde: es la varianza poblacional, Xi s el término de conjuntos de datos

media de la población, N: tamaño de la población

Por otra parte, para realizar el aplicativo web se usa la programaciòn orientado a objetos asi como el lenguaje orientado a objetos. Así lo define el autor Luis Izquierdo en su documento electrónico *Introducción a la Programación*.

1. *Progrramación orientada a objetos.*

El lenguaje de programaciòn orientada a objetos (POO) el elemento basico no es la funciòn, sino un ente denominado precisamente objeto. Un objeto es la representación en un programa de un concepto, y contiene toda la información necesaria para abstraerlo: datos que describen sus atributos y operaciones que pueden realizarse sobre los mismos. Ahora, se debe tener claro que no es lo mismo progrmaciòn orientada a objetos que un lenguaje orientado a objetos. La programación orientada a objetos es una “filosofía”, un modelo de programación, con su teoría y su metodología. (Alvarez, s.f.) Un lenguaje orientado a objetos es un lenguaje de programación que permite el diseño de aplicaciones orientadas a objeto.

En la POO el objeto es la clase, en el cual se plasma los atributos del objeto y los métodos asi lo especifica el autor Alvarez Miguel Angel en su pagina web.

« Una clase son declaraciones de objetos, también se podrían definir como abstracciones de objetos. Esto quiere decir que la definición de un objeto es la clase. Cuando programamos un objeto y definimos sus características y funcionalidades en realidad lo que estamos haciendo es programar una clase» (Alvarez, 2001).

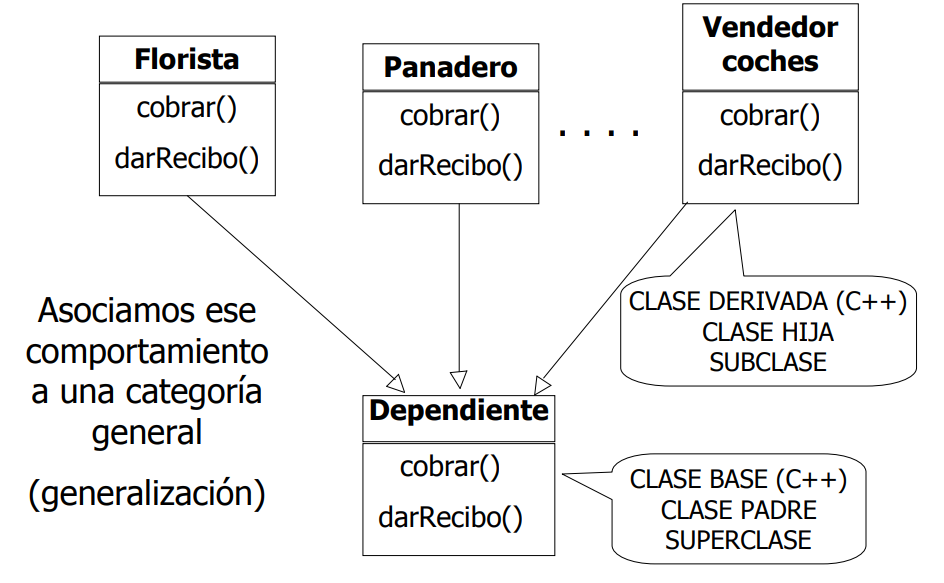
Las cualidades que se usa para este trabajo es la herencia y la encapsulación. La herencia es la más «importante de un sistema POO, la que nos dará mayor potencia y productividad, permitiéndonos ahorrar horas y horas de codificación y de depuración de errores» y la encapsulación busca organizar los datos y metodos del objeto de manera que no se pueda acceder a ellos desde otro medio distinto a la que se hayan especificado. Los datos pueden ser privados o públicos a través de la encapsulación se podra acceder a estos directamente. Una ventaja de la programación orientada objetos es que te permite reutilizar codigos y evitar la redundacia de códigos.

La autora Cristina Cachero, Pedro J. Ponce de León define herencia como:

1. *Herencia*

Es una propiedad que permite que los objetos sean creados a partir de otros ya existentes, obteniendo características (métodos y atributos) similares a los ya existentes. Es la relación entre una clase general y otra clase más especifica. Es un mecanismo que nos permite crear clases derivadas a partir de clase base, Nos permite compartir automáticamente métodos y datos entre clases subclases y objetos. Por ejemplo.

Figura1: Diagrama de clases aplicando herencia.



Según el autor Daniel Lara en su página web Laraveles define al encapsulamiento de datos en la programación orientada a objetos como:

1. *Encapsulamiento de datos.*

Es el proceso que consiste en organizar los datos y operaciones (métodos) de una clase que constituyen su estructura y su comportamiento con el fin de evitar el acceso a datos por cualquier otro medio que no sea el especificado y por esto el encapsulamiento de datos asegura la integridad de los datos que contiene el objeto.

El lenguaje de programación a utilizar para realizar este aplicativo web a realizar es JAVA. La fuente se ha obtenido de los autores Dra. María del Carmen Gómez Fuentes y Dr. Jorge Cervantes Ojeda en su libro Introducción a la Programación Web con Java: JSP y Servlets, JavaServer Faces. Y asi mismo también detallan sobre Java Server Page (JSP) y Lenguaje Marcado de Hipertexto (HTML).

1. *Lenguaje de Programación Java.*

Este lenguaje como caracteristicas importantes es que se construyo especificamente orientado a objetos, y que una vez compilado el codigo se puede llevar a otra maquina para ser nuevamente ejecutado sin modificación alguna; esto es posible ya que el código se ejecuta en una maquina virtual. El aplicativo es web, al utilizar el lenguaje JAVA se necesita de un JSP y un Servlet. EL JSP es una tecnología que otorga la facilidad de combinar el código HTML estatico con código generado dinamicante en un solo archivo. Un Servlet es una clase Java que corre en el servidor, estos actuan como capa intermediaria entre petición del navegador web o algún otro HTTP con la base de datos o aplicaciones en el servidor HTTP (Cervantes Ojeda & Gómez Fuentes, 2017). Sin embargo, para el proyecto no se desarrollara servlets sino que se genera directamente con el motor JSP.

1. *Página Java Server Page.*

Una página JSP (JavaServer Page) es una página HTML a la que se le incrusta código Java. El código Java se incrusta entre los siguientes indicadores <% y %>.

1. *Las Directivas.*

Son instrucciones procesadas por el motor JSP y cuando la página JSP se traduce a un *Servlet.* Las instrucciones están encerradas entre etiquetas <%@ y %>. (Ver figura 4)

1. Los Scriptlets.

Permite impregnar partes de código java. Este código se inserta directamente al Servlet generado cuando ejecutamos la página, estas van encerradas en <% y %>. (Ver igura 5)

1. *Variables predefinidas*

Un scriplet puede usar las siguientes: sesión, request, response, out e in. En este proyecto no se ha usado la última variable predefinida. Request es un objeto HttpServle6Request, response un objeto HttpServletResponse, out un objeto Print Writer, e in un objeto BufferedReader. Tener presente que estas variables también las usa los *Servlets*, pero no se declara.

1. *Las Expresiones.*

Permite recuperar dinámicamente o calcular valores a insertar directamente en la página JSP.

1. *Los Objetos Implícitos.*

El motor de JSP ofrece instancias de un conjunto de clases, estos objetos ya se encuentran establecidos que solo se usan dentro del código java. En nuestro aplicativo se ha usado los siguientes:

Tabla 1: Funciones de los Objetos implícitos

|  |  |  |
| --- | --- | --- |
| Objeto Implícito | Objeto | Funciones |
| request  Datos de la petición. | *javax.servlet.http.HttpServletRequest*  El *HttpServletRequest* está asociada con la petición | Permite mirar los parámetros de la petición mediante getParameter. |
| response  Datos de la respuesta. | *javax.servlet.http.HttpServletResponse* | El *HttpServletResponse* es el asociado con la respuesta del cliente. |
| out  Flujo de salida para el contenido de la página. | *javax.servlet.http.jsp.JSPWriter* | El *JSPWriter* se usa para enviar la salida al cliente. |
| exception | *java.lang.Throwable* | Se muestra errores o excepciones no capturadas. |

Esta tabla especifica los objetos implícitos que se han insertado en el aplicativo web desarrollado.

Por otro lado, también se usaron ciertas librerías y clases que proporción Java como:

1. *Clase Date.*

Son instancias que proporciona y almacenan fecha y hora de su librería.

1. *Java Collections Framework.*

En el array que se crea permite tener un conjunto de objetos indexados por un número, que vendría ser la posición. Se ha usado el array para almacenar los datos ingresados de los objetos.

1. *Interfaz Collection.*

Esto especifica funciones para manejar grupos de objetos, conocidos como elemento.

1. *Interfaz List.*

Este especifica las operaciones de una colección ordenada. Y proporciona un control sobre en dónde insertas los elementos. Una de sus implementaciones es el ArrayList. Este implementa como un array. Cuando el array se llena, se reserva otro más grande y se desplazan los elementos de array del antiguo al nuevo.

1. *Bucles sobre colecciones.*

Se usa el ***for*** para iterar un objeto de una colección de Datos.

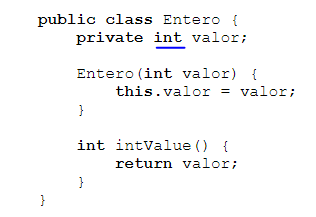
1. *Lenguaje Marcado de HiperTexto (Html).*

HTML es el acrónimo de las siglas del inglés: HyperText Markup Language (lenguaje de marcas de hipertexto). Las páginas de hipertexto se codifican con HTML. Los navegadores de internet están preparados para interpretar los comandos de HTML y desplegar la información en forma de una página de internet con texto e imágenes. El hipertexto de las páginas HTML está definido por tags, que son los comandos. Los tags comienzan con el carácter < y terminan con el carácter >. La información del tag va dentro de estos dos caracteres. Los navegadores arman y dan forma a las páginas de internet interpretando estos comandos.

1. *Clases envoltorio.*

En ocasiones es muy conveniente poder tratar los datos primitivos (int, boolean, etc.) como objetos. Por ejemplo, los contenedores definidos por el API en el package java.util (Arrays dinámicos, listas enlazadas, colecciones, conjuntos, etc.) utilizan como unidad de almacenamiento la clase Object. Dado que Object es la raíz de toda la jerarquía de objetos en Java, estos contenedores pueden almacenar cualquier tipo de objetos. Pero los datos primitivos no son objetos, con lo que quedan en principio excluidos de estas posibilidades. Como se muestra en la siguiente figura:

Figura 2: ejemplo del uso clase envoltorio.



1. *Diagrama de Clases.*

Un diagrama de clase según el autor Manuel Cillero lo define que el diagrama de clases recoge las clases de objetos y sus asociaciones. En el diagrama se representa la estructura y el comportamiento de cada uno de los objetos del sistema y sus relaciones con los demás objetos, pero no muestra información temporal. Con el fin de facilitar la comprensión del diagrama, se pueden incluir paquetes como elementos del mismo, donde cada uno de ellos agrupa un conjunto de clases.

# **CAPITULO III**

## **DESARROLLO DE LA SOLUCIÓN**

### **3.1 Descripción a la solución.**

En el desarrollo de este proyecto como solución al problema planteada se propone implementar una aplicación web. Esta aplicación funciona dentro de la internet, así como el cliente puede visualizar el contenido a través de su navegador en el internet. La página web se desarrolla con HTML (Lenguaje de Hipertexto Marcado) y JSPS (Java Server Pages). Las paginas JSPS se ejecutan en una máquina virtual de Java, en ella se encuentra todo el HTML que el cliente podrá interactuar, permitiendo desarrollar una página web dinámica, esto es posible a las etiquetas de HTML otorgada.

Las aplicaciones web se almacenan en un servidor ya que corren dentro de ella es decir se ejecutan. Y para este proyecto se usará *Apache Tomcat* y por se necesita también el *GlassFish*. Tener presente que el servidor envía las respuestas de una solicitud que se realice con la aplicación.

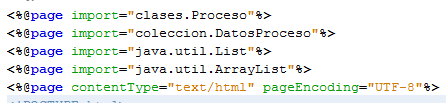
La lógica del negocio será desarrollada en el lenguaje Java por medio de las clases java, así mismo el IDE que se usa es *Neatbeans*, Java web proporciona la facilidad de desarrollar aplicaciones del lado servidor ya que, contiene Servlets, así como proporciona diferentes librerías.

El uso del JSP es muy importante en este proyecto ya que cuando se ejecuta por primera vez, el motor JSP genera un *Servlet* correspondiente a la página, esto se le llama instancia. La instancia de la JSP se guarda en la memoria del servidor. Y como el JSP contienen código Java, se inicia a lo que se denomina un hilo o *thread* y luego cada vez que hay una solicitud de la página, se usa la instancia del JSP y así sucesivamente se va creando un hilo por cada solicitud. (Cervantes Ojeda & Gómez Fuentes, 2017) Esto sucede cuando el JSP contiene código Java para procesar datos, en esta aplicación web la comunicación se dará entre JSP y Clases Java. Para realizar el código Java en una JSP es necesario conocer lo siguiente:

* + 1. **Las Directivas.**

Se usa las directivas para importar cualquier paquete o archivo que se requiera para el proyecto, así como las librerías que proporciona java, se ha utilizado en todos los JPS y clases estos se encuentran entre <%@ y %>. Algunos ejemplos a continuación:

Figura 3: Directivas de guardar datos pedido JSP



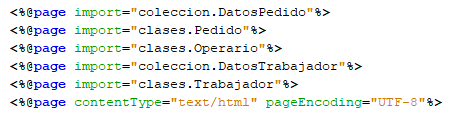
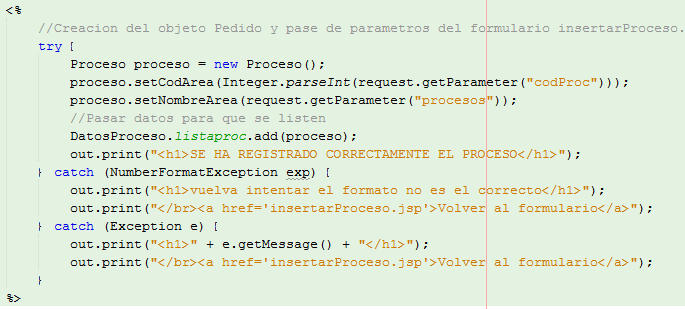


Figura 4: Directivas de guardar datos trabajador JSP

* + 1. **Los Scriptlets.**

Con este indicador se ha colocado el código java en todos los JPS, solo en aquellos que se realiza un proceso; como guardar datos del formulario ingresado y listar datos que provienen del JPS guardar datos y siempre están encerradas en <% y %>. Por ejemplo en la siguiente figura se muestra el scriptlet del JSP guardar proceso.

Figura 5: scriptlets del guardar proceso



* + 1. **Variables predefinidas.**

Unas de las variables predefinidas se muestran como ejemplo a continuación, en dónde request muestra los parámetros enviados del formulario insertar Trabajador JSP. Y el out sirve para imprimir en pantalla.

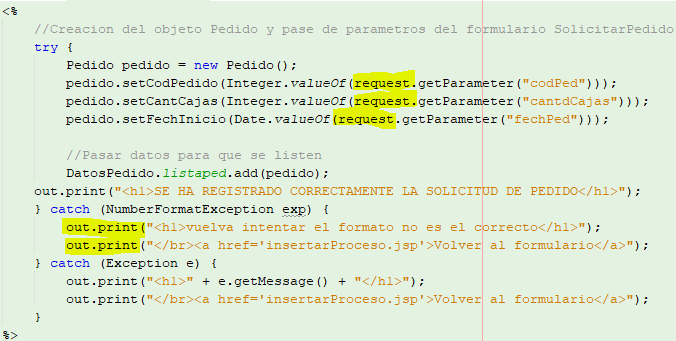
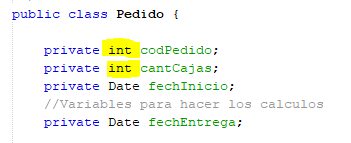


Figura 6: JSP guardar solicitud pedido

* + 1. **Clases envoltorio.**

Son Integer, Double que se ha usado para este aplicativo, esto se genera por cada tipo básico int y double respectivamente.

Figura 7: Clase pedido



* + 1. **Clase Date.**

Se usó para ingresar en el JSP pedido fecha de inicio y se halla también la fecha de entrega y se usa el tipo básico Date, por ende, usa la clase Date. Proviene de la librería útil Date.

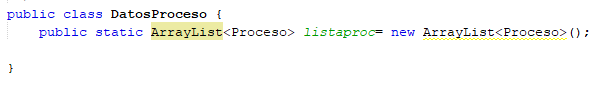


Figura 8: java útil date

* + 1. **Interfaz List.**

La creación de una lista facilita el alamcena de los objetos, primero importando el java.util.ArrayList. La creación de una lista es de esta manera:

Figura 9: java útil array list



* + 1. **Bucles sobre colecciones.**

El bucle que se usa es el for para listar la información almacenada en la clase y while para realizar un cálculo en el método de la clase Pedido.

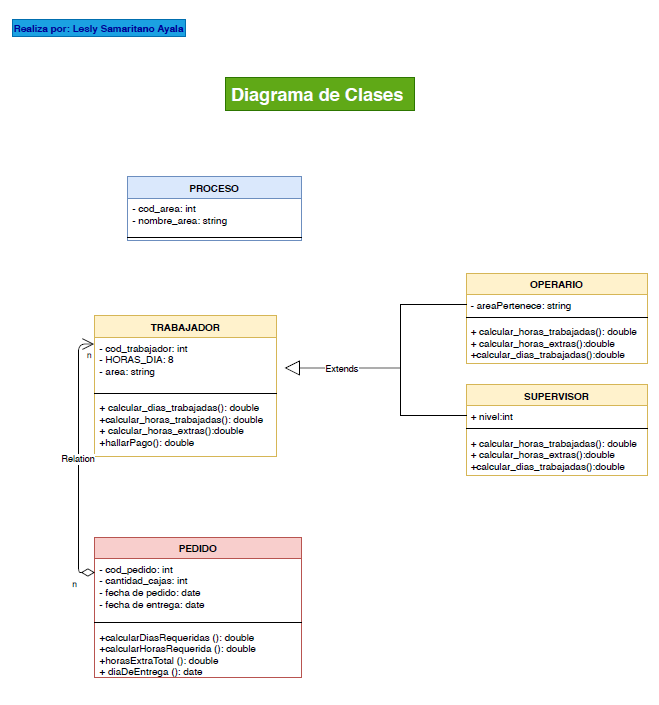
* + 1. **El framework de diseño.**

Se usa Bootstrap versión 3.7. Este es el marco de código abierto con HTML, CSS y JS (java script) ya que, se adapta en cada dispositivo fácilmente.

### **3.2 Diagrama de clases.**

Las clases de este proyecto son: el trabajador que hereda a supervisor y operario; proceso y pedido.

Figura 2: Diagrama de clases del aplicativo web



### **3.3 Estructura del proyecto.**

El proyecto se encuentra estructurado de la siguiente manera:

* Folder Web Pages: se encuentran todos los JSP.
  + index. – es el inicio de la aplicación web, a la cual me enviara a las otras páginas JPS.
  + SolicitarPedido. – se podrá realizar un pedido de cantidad de cajas que han sido solicitadas por la procesadora por medio de un formulario que envía a *guardarDatosPedido*.
  + insertarProceso. – un formulario para ingresar procesos y existan para enviar a guardarla en *guardarSolicitudPedido*.
  + insertarTrabajador. – llenar un formulario con los datos solicitados para buscar a un operario de alguno de los procesos.
  + guardarDatosProcesos. – los datos ingresados de *insertarProceso* se guardarán en este JSP. Y si se realizó satisfactoriamente mostrar un mensaje de éxito si no se tendrá que volver ingresar los datos.
  + guardarDatosTrabajador. - los datos ingresados de *insertarTrabajador* se guardarán en este JSP para ser buscado posteriormente. Y si se realizó satisfactoriamente mostrar un mensaje de éxito si no se tendrá que volver ingresar los datos.
  + guardarSolicitudPedido. – los datos que se hayan insertado en *SolicitudPedido* se guardaran aquí. Y si se realizó satisfactoriamente mostrar un mensaje de éxito si no se tendrá que volver ingresar los datos.
  + listarPedido. – se mostrará una tabla de los datos capturados con sus respectivos requerimientos que necesita la procesadora.
  + listarProcesos. – se mostrará una tabla de los datos capturados de *insertarProcesos*.
  + lisarTrabajador. - se mostrará una tabla de los datos capturados con sus respectivos requerimientos que necesita la procesadora.
* Folder Source Package: se encuentran los paquetes creados para almacenar las *clases.java*.
* Paquete clases: aquí está la lógica del negocio.
  + Pedido. – se almacena los datos guardados de la JSP *guardarDatosPedido,* así como los cálculos más importantes del proyecto, para así mostrar los requerimientos que se necesitan.
  + Trabajador. – se encuentra el cálculo secundario para los operarios, además almacenar los datos provenientes del JSP *guardarDatosTrabajador*.
  + Supervisor. – almacenar los datos provenientes del JSP *guardarDatosTrabajador*. Esta clase hereda a la clase *Trabajador*.
  + Operario. – almacenar los datos provenientes del JSP *guardarDatosTrabajador*. Esta clase hereda a la clase *Trabajador*.
  + Proceso. – almacenar los datos capturados del JSP *guardarDatosPocesos*.
* Paquete colección: clase para almacenar la lista de un objeto.
  + DatosPedido. – recibe los datos de la clase *pedido* y los almacena a un arrayList.
  + DatosTrabajador. – recibe los datos de la clase *trabajador* y los almacena a un arraList.
  + DatosProceso. – recibe los datos de la clase *proceso* y los almacena a un arrayList *listproc*.

### **3.4 Contenido del proyecto.**

#### **3.4.1 Clases.**

A continuación, se mostrará las clases más importantes del proyecto desarrollado.

##### *Clase Pedido.*

##### *Clase Trabajador.*

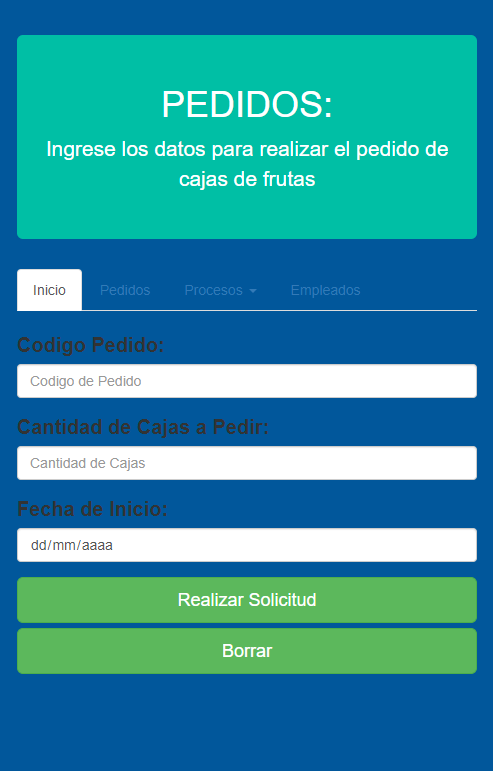
##### *Clase DatosPedido.*

##### *Clase DatosTrabajador.*

#### **3.4.1 Vistas.**

1. Inicio.
2. Pantalla realizar solicitud de pedido. Este envía los datos al JSP guardarDatosPedido. Se usará cuando el supervisor de la procesadora desea ingresar una determinada cantidad de cajas.

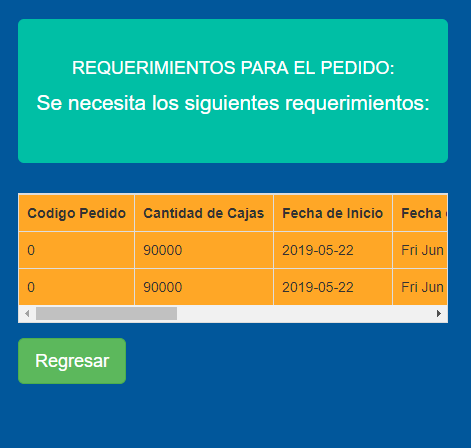
Figura 3: vista de la solicitud de pedido



La imagen representa un formulario, en el cual se ingresarán los siguientes datos para realizar un pedido. En la parte superior se muestra las pestañas a los que me llevan a Inicio si se hace clic mostrara el inicio de la página web; en la pestaña pedidos

1. Listar solicitud del pedido, aquí se muestra la lista de los datos capturados en el formulario de solicitud pedido, con sus respectivos requerimientos.

Figura 4: vista de listar pedido



1. Buscar trabajador, en esta vista se buscará a un determinado operario por su código y área.

Figura 5: vista de insertar trabajador



1. Vista listar trabajador, mostrara la información que posee un operario de la procesadora.

Figura 6: vista de listar trabajador



### **3.5 Implementación.**

#### **3.5.1 Recomendaciones para ambiente de desarrollo**

* Programar en un lugar con luz enfocada solo para la persona que desarrolla el aplicativo, pero si solo es una persona.
* Usar diferentes equipos para facilitar la visión, es decir usar dos maquina una para desarrollar el aplicativo y otra para buscar información u otras necesidades. Además, tener diferentes espacios para colocar tus libros, así no hay algo alrededor que estorbe tu espacio de trabajo.

#### **3.5.2 Recomendaciones para ambiente de prueba**

* Revisa si tienes todas los softwares o herramientas que necesitas para realizar la prueba.
* En caso de tener un problema con el IDE Neatbeans tener un plan de contingencia, ya sea un video, prototipo o tener el proyecto almacenado en un repositorio. En caso de realizar este se ha usado GitHub.

# **CAPITULO IV**

## **RESULTADOS**

### **4.1 Prueba del aplicativo.**

Una procesadora de frutas determinada demora en finalizar con la producción de frutas, ya que no se tiene un control adecuado del tiempo en que se termine con la cantidad de cajas solicitada por la planta. La procesadora de frutas no desea contratar más personal, solo desea trabajar con la cantidad actual en planilla. Sin embargo, al día se produce una cantidad mínima de cajas y los días tienen que aplazarse. Cuenta con los procesos recepción con 3 trabajadores y 3 minuto por caja para enviar las cajas, selección con 40 operarios y 1minuto por caja igualmente para empaquetado y pesado con 5 operarios y 1 min de pesado por caja. Además, que la cantidad mínima que sale del pesado es 2400.

### **4.2 Resultados Esperados.**

La ciencia que se usa es la Estadística Inferencial y fórmula planteada para resolver este caso en este problema es:

Ecuación 2: Fórmula de fracción de muestra

En dónde n: es el tamaño de muestra y N es la población.

Primeramente, para llegar a tener el valor de “n” que es el tamaño de muestra, y será el mínimo de cantidad de cajas hay que seguir previamente unas más formulas y pasos. Se halla la variabilidad que hay en los procesos de recepción, selección, empaquetado y pesado, con la cantidad de operarios respecto al tiempo y se halla con la siguiente fórmula.

Ecuación 3: Varianza Poblacional

En dónde: es la varianza poblacional, Xi s el término de conjuntos de datos

media de la población, N: tamaño de la población

El tamaño poblacional es: 2400 cantidad de cajas de frutas.

Var: la variabilidad de cantidad de cajas que se realiza en los procesos según al tiempo máximo de los operarios en el día. X1 es la cantidad máxima de cajas que realizan en el proceso de recepción y así sucesivamente hasta pesado.

Tabla2: Resolución de la Varianza Poblacional

|  |  |  |  |
| --- | --- | --- | --- |
| xi | x | x-u | (x-u) al 2 |
| x1 | 480 | -9840 | 96825600 |
| x2 | 19200 | 8880 | 78854400 |
| x3 | 19200 | 8880 | 78854400 |
| x4 | 2400 | -7920 | 62726400 |
|  |  |  | 317260800 |

Resultado de la varianza:

Tabla3: Resultado de la varianza poblacional

|  |  |
| --- | --- |
| U | 10320 |
| N | 2400 |
| var | 132192 |

En segundo lugar, hallar la cantidad mínima caja que se produce al día y esto se calcula con la fórmula siguiente.

Ecuación 1: Tamaño de muestra

En dónde: es la varianza poblacional, Z la Z de la distribución normal

el error que se genera. Equivale a 0.05

Se realiza la prueba con un 95 de confiabilidad y 0.05 de des confiabilidad.

El resultado de es equivalente a n = 203131514.9 como se conoce la población se tiene que validar la muestra. Si la fracción de muestra es mayor 0.05 se sigue segmentado el n. Y para ello se comprueba con la división de la fracción molar. *(Ver ecuación 1).*

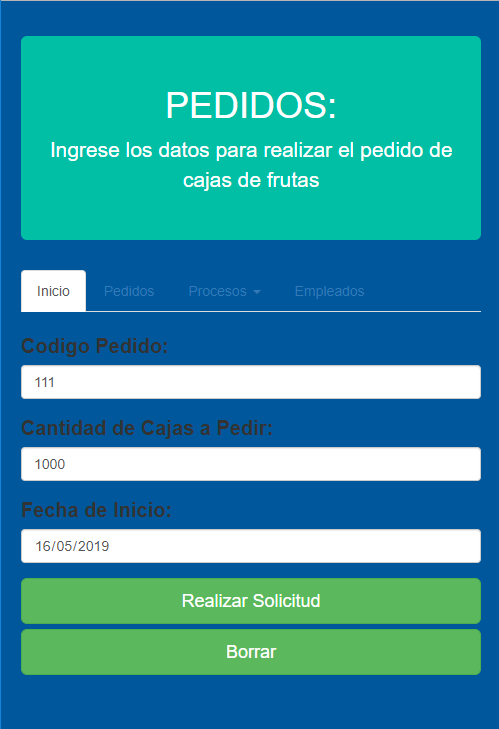
Después de fraccionar 3 veces se obtuvo un n de 1200. Esto es la cantidad mínima que se puede producir en cajas en el día con la variabilidad que hay. La cantidad mínima de cajas se usa para comparar con la cantidad que solicita la procesadora. Y se obtienen la cantidad de días que se requiere para finalizar, las horas totales, y horas extras. Así como la cantidad de horas aumentar.

Si se ingresa 1000 cajas a producir. El día mínimo es 1 con total de horas 6.66667 y no necesita horas extras.

### **4.3 Pruebas del Aplicativo.**

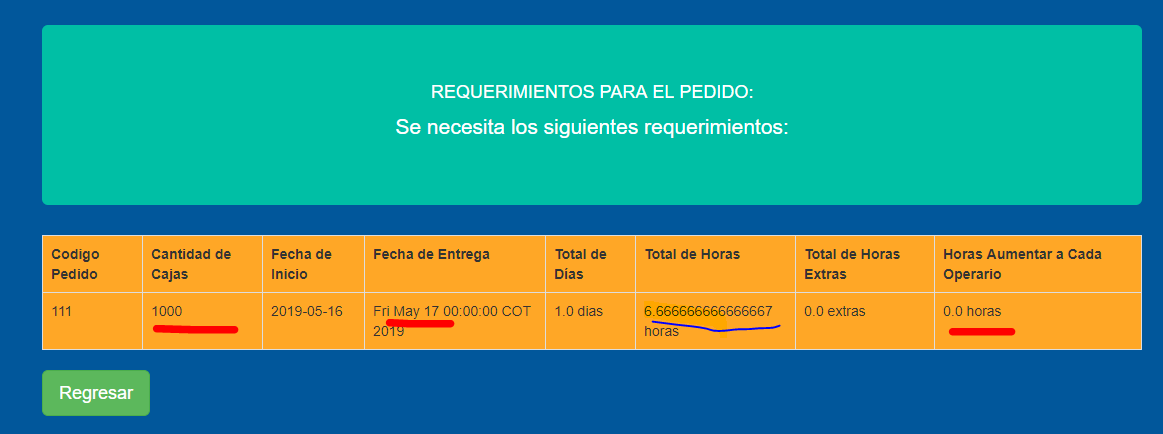
1. Se ingresa los datos. La prueba se hace con 1000 de cajas de frutas. Clic en el botón realizar solicitud

Figura 7: vista de realizar pedido



1. Regresas al index y te muestra los cálculos realizados.

Figura 8: vista de datos



### **4.4 Resultados según objetivos.**

El objetivo general es calcular las horas extras. Para el caso de 1000 de cajas de frutas no es necesaria, y el mínimo día es de 1, ya que los operarios estarían trabajando menos de 8 horas. Y de los objetivos específicos hallar el total de horas obtenido es de 6.67 horas totales.

### **Costos y Beneficios.**

#### **4.5.1 Presupuestos.**

Los gastos que se tiene y lo que se piensa ganar son los siguientes:

Tabla 3: Cuadro de Presupuestos

|  |  |
| --- | --- |
| Gasto | Cantidad en (s/.) |
| Diseño del aplicativo | 500.00 |
| Diseño de la conexión | 600.00 |
| Mantenimiento | 250.00 |
| Dominio | 50.00 |
| Hosting | 150.00 |
| Servidor | 3000.00 |
| Sueldos | 1200.00 |
| Implementación | 2000.00 |
| Computadoras | 3000.00 |

#### **4.5.2 Costos**

Este cuadro de gastos, se detallan la cantidad aproximada de inversión.

Tabla 4: Cuadro de Ingresos

|  |  |
| --- | --- |
| Ingresos | Cantidad en (s/.) |
| Plan anual | 3000.00 |
| Marketing | 100.00 |
| Tienda online | 4000.00 |
| Reparación a Maquinas | 2000.00 |

El plan anual es por el servicio del aplicativo.

El marketing es por la publicidad en la página web por la cantidad de cookies instaladas. Así como el beneficio si compran el producto lanzado.

Una tienda online es un cargo fijo de agregar a una procesadora y las ganancias de ventas es un monto no exacto.

El servicio de máquina será por contacto con algunos socios, y ofrecer la reparación.

# **CAPITULO V**

## **CONCLUSIONES Y RECOMENDACIONES**

### **5.1 Conclusiones.**

Respecto al primer objetivo específico **“**Desarrollar un aplicativo que calcule el tiempo total que requiere una cantidad de cajas solicitadas”, el presente proyecto empleo la fórmula *de determinación de tamaño de muestra* que ofrece la Es dística Inferencial y la *fracción de una muestra*. Se pudo concluir que lo siguiente:

* Se requiere de otra clase que actualice los datos que se van registrando y así usarlas en otras clases de objetos para otros intereses.
* La población de la cantidad de cajas es variante para una procesadora por lo cual debería ser un dato ingresado para los cálculos.

Respecto al segundo objetivo específico “Calcular las horas que se tiene que aumentar a cada operario que se tiene en ese instante para la línea operativa” se concluye lo siguiente:

* Se requiere como dato a ingresar la cantidad de los operarios que se encuentren laborando en planilla actualmente. Para que no se tenga que modificar el código fuente.
* Los cálculos que se realizan en cada clase son independiente por lo cual no se puede utilizar los datos que se ingresan de un JSP diferente y se envié a uno especifico por medio del formulario.

### **5.1 Recomendaciones.**

* Trabajar con algún patrón de tal modo que organice a la aplicación en segmentos independientes. Ya que no es una buena práctica incluir código javo en un JSP para que los datos sean procesados.
* Al realizar array es más sencillo con una lista que proporciona la librería de java, denominado java útil ya que, cuando se añade un objeto lo almacene directamente a una lista y no a un arreglo.
* Almacenar el código fuente en un repositorio para este proyecto se ha usado GitHub, su función es alojar proyectos con código fuente de programas de computadora.

# **REFERENCIAS BIBLIOGRÁFICAS**

Alvarez, M. A. (24 de julio de 2001). *desarrolloweb.com*. Obtenido de desarrolloweb.com: https://desarrolloweb.com/articulos/499.php

Cervantes Ojeda, D. J., & Gómez Fuentes, D. d. (2017). Introducción a la Programación Web con Java: JSP y Servlets, JavaServer Faces. (U. A. Metropolitana, Ed.) México, Guajimalca de Morelos. Obtenido de http://www.cua.uam.mx/pdfs/revistas\_electronicas/libros-electronicos/2017/java/Java.pdf

i68, G. (s.f.). *Software de gestión ERP para empresas innovadoras i68*. Obtenido de https://www.grupoi68.com/software-erp-gestion-estadisticas-procesos/

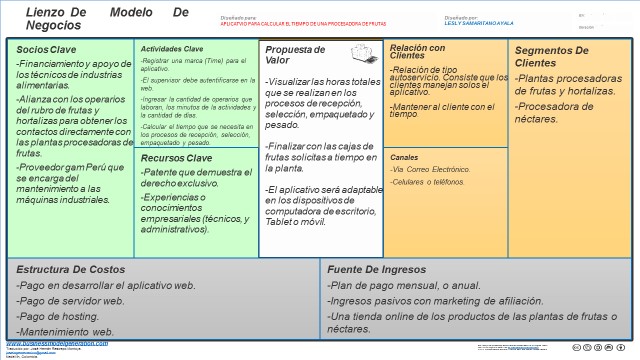
Izquierdo, L. R. (s.f.). *Introducción a la Programación.* Obtenido de http://luis.izqui.org/resources/ProgOrientadaObjetos.pdf

SisTrade. (2019). *SisTrade Software Consulting* . Obtenido de http://www.sistrade.com/es/soluciones/gestion-produccion/industria-40/

Zapata, C. M. (2017). *GestioPolis*. Obtenido de https://www.gestiopolis.com/programacion-lineal-en-la-investigacion-de-operaciones/

# **ANEXOS**

## **ANEXO 1: LEAN CANVAS**

****

## **ANEXO 2: TABLA DE LAS VARIABLES DE LA CLASE PEDIDO**

|  |  |  |
| --- | --- | --- |
| Variable | Tipo | Significado |
| codPedido | Int | Es el código de solicitud |
| cantCajas | Int | Cantidad de cajas a solicitar |
| fechInicio | Date | Es la fecha en que se desea iniciar la producción |
| fechEntrega | Date | Fecha en que se finaliza la solicitud de pedido |
| Date ParseFecha(String fecha) | String | Método que convierte un Date a String |
| sumaXU | Double | Metodo para hallar la suma entre las xi y la media poblacional |
| x1  x2  x3  x4 | x1= cantd de cajas en el recepciòn  x2= cantd de cajas en el selecciòn  x3= cantd de cajas en el empaquetado  x4= cantd de cajas en el pesado | Guardar el tiempo total de un día en cada proceso. |
| u | Int | La media poblacional |
| varianzaPoblacional() | Doublé | Metodo para hallar la varianza |
| tamañoCajas() | Int | Tamaño de cajas minimo en un día |
| calcularDiasRequeridas() | Double | Halla los días que se necesitan según a la cantidad de cajas que solicita. |
| calcularHorasRequerida() | Double | Halla los horas que se necesitan según a la cantidad de cajas que solicita. |
| horasExtraTotal() | Double | Halla los horas extras total de un día que se necesitan. |
| diaDeEntrega() | Date | Metodo para hallar la fecha del último día que se produce. |
| calcular\_horas\_aumentar() | Double | Calcula las horas que se debe aumentar a cada operario |