Conteo Físico de Inventario

Brayan Aponte, Carlos Jiménez, Santiago Londoño

No. de Equipo Trabajo: 21

I. INTRODUCCIÓN

El presente documento aborda el proyecto de software "Conteo físico de inventario" orientado a registrar el conteo realizado de manera física a un inventario específico.

En este informe se presenta su justificación, descripción de usuarios, funcionalidades, requerimientos, entornos de desarrollo y operación, y de su interfaz de usuario (la cual se apoya de un mockup) además de incluir un enlace a su repositorio en GitHub y presentar resultados obtenidos luego de someter operaciones significativas del software (en cuanto a complejidad computacional) a grandes cantidades de datos.

II. DESCRIPCIÓN DEL PROBLEMA A RESOLVER

Muchas empresas hoy día manejan el control de sus inventarios con diferentes soluciones de software que se encuentran en el mercado, las cuales les brindan la posibilidad de controlar los productos que entran y salen. No obstante, es sorprendente ver que la gran mayoría de estas empresas cuando desean realizar un conteo físico de su inventario, es decir, una validación física de sus productos, optan por realizarlo en archivos excel u hojas de papel. Lo que genera perdida de tiempo y posibilita el error humano.

Este tipo de registros manuales, no solo genera gran posibilidad de error en los datos, sino que no permite ver a detalle observaciones del conteo físico como la hora en la que se registró un producto, que usuario lo hizo, si se realizo algún ajuste posteriormente, entre otros. Es así que se genera la necesidad de tener un software que permita registrar estos datos con facilidad, seguridad y rapidez.

III. USUARIOS DEL PRODUCTO DE SOFTWARE

Todas las empresas que actualmente manejan inventarios de productos se encuentran en la necesidad de revisarlos periódicamente para validar que no existan diferencias entre los datos registrados en sus sistemas y los productos reales. Los usuarios que utilizarán el sistema se dividen principalmente de dos grupos: aquellos que tendrán control total sobre el inventario que se está contando (creación, modificación, eliminación de productos) y aquellos usuarios que únicamente podrán realizar registros a dicho inventario (cantidad de un producto específico que se contó). Para su uso no se requiere ningún nivel de experiencia y puede ser utilizado por prácticamente cualquier persona que haya usado un teléfono móvil.

.

IV. REQUERIMIENTOS FUNCIONALES DEL SOFTWARE

AUTENTICACIÓN

Descripción:

Permite ingresar a la aplicación con los datos correctos de usuario y contraseña.

Acciones iniciadoras y comportamiento esperado:

El usuario deberá ingresar su usuario y su contraseña, las cuales una vez ingresadas, el sistema se encargará de corroborar si son correcta, si es así, el usuario podrá ingresar, de lo contrario, su ingreso será restringido y dirigido nuevamente para ingresar el nombre de usuario y contraseña.

Requerimientos funcionales:

- Búsqueda de usuario y contraseña ingresados
- Validación de los datos

ADMINISTRACIÓN DE USUARIOS

Descripción:

Creación, modificación y eliminación de los usuarios que podrán acceder a la aplicación.

Acciones iniciadoras y comportamiento esperado:

Si el usuario cuenta con permiso de administrador ingresará a una pantalla donde podrá consultar todos los usuarios de la aplicación, filtrar uno especifico, realizar modificaciones a usuarios existentes, así como su eliminación y la creación de nuevos usuarios. El sistema se encargará de realizar la búsqueda de los usuarios solicitados y mostrarlos en pantalla, de igual forma, recibirá los datos ingresados por el usuario y realizará la acción pertinente (creación, modificación, eliminación de un usuario)

Requerimientos funcionales:

- Creación de Usuarios
- Consulta de Usuarios
- Actualización de Usuarios

- Eliminación de Usuarios
- Almacenamiento de Usuarios

CREACIÓN DE INVENTARIO

Descripción:

Permite la creación de un nuevo conteo de inventario.

Acciones iniciadoras y comportamiento esperado:

El usuario podrá crear un nuevo inventario vacío o la creación de uno desde un archivo Excel. El sistema deberá crear un nuevo inventario y en caso de que se cargue desde un archivo, deberá crear todos los artículos contenidos en él, con sus respectivos atributos.

Requerimientos funcionales:

- Creación de Inventario
- Creación de Artículos
- Almacenamiento de Inventario
- Almacenamiento de Artículos

CONSULTA DE ARTÍCULOS

Descripción:

Permite consultar los artículos creados para el inventario seleccionado.

Acciones iniciadoras y comportamiento esperado:

El usuario podrá consultar todos los artículos registrados para el inventario actual, al igual que filtrar uno específico desde la barra de búsqueda. El sistema se encargará de realizar la búsqueda solicitada por el usuario y mostrará en pantalla los artículos que coincidan con el criterio de búsqueda.

Requerimientos funcionales:

- Consulta de Artículos

ADMINISTRACIÓN DE ARTÍCULOS

Descripción:

Permite al usuario crear, modificar y eliminar artículos para el inventario seleccionado.

Acciones iniciadoras y comportamiento esperado:

El usuario podrá realizar modificaciones a artículos existentes, así como su eliminación y la creación de nuevos. El sistema se encargará de realizar la búsqueda de los artículos solicitados y mostrarlos en pantalla, de igual forma, recibirá los datos ingresados por el usuario y realizará la acción pertinente (creación, modificación, eliminación de un artículo)

Requerimientos funcionales:

- Creación de Artículos
- Consulta de Artículos
- Actualización de Artículos
- Eliminación de Artículos
- Almacenamiento de Artículos

REGISTRO DE CONTEO

DESCRIPCIÓN:

Permite ingresar el conteo de un producto específico.

Acciones iniciadoras y comportamiento esperado:

Desde el módulo de consultas, el usuario podrá seleccionar un artículo y para este registrar la cantidad que se ha contado. El sistema deberá recibir la cantidad registrada y sumarla a la existente para el artículo seleccionado.

Requerimientos funcionales:

- Creación de conteo
- Actualización de Artículo

EXPORTACIÓN DE INVENTARIO

DESCRIPCIÓN:

Permite exportar el inventario registrado.

Acciones iniciadoras y comportamiento esperado:

El usuario podrá exportar el conteo del inventario a un archivo Excel. El sistema deberá retornar todos los artículos registrados para el inventario con sus respectivas cantidades y guardar en un archivo Excel.

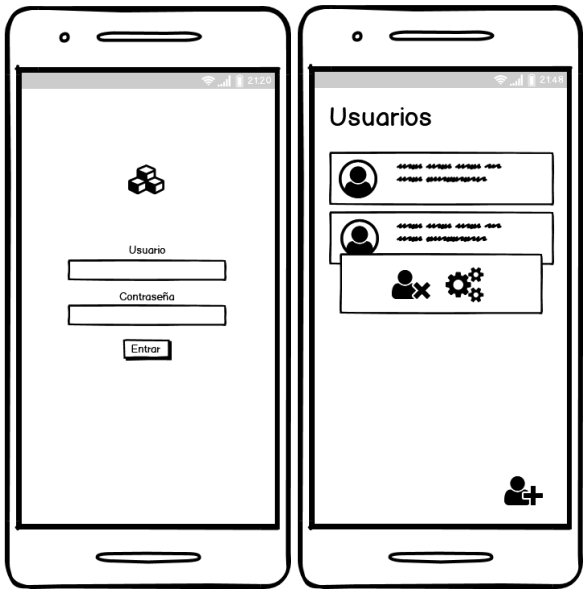
Requerimientos funcionales:

- Consulta de Artículos
- Almacenamiento de Artículos

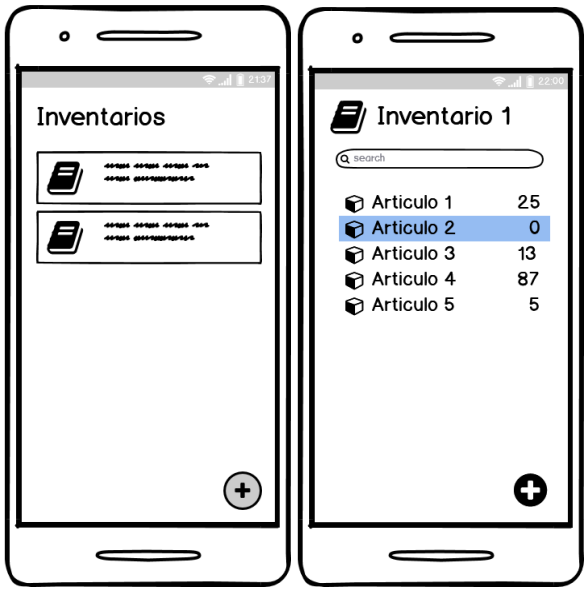
V. DESCRIPCIÓN DE LA INTERFAZ DE USUARIO PRELIMINAR

El sistema contará con tres módulos principales: Usuarios, Inventarios y Productos.

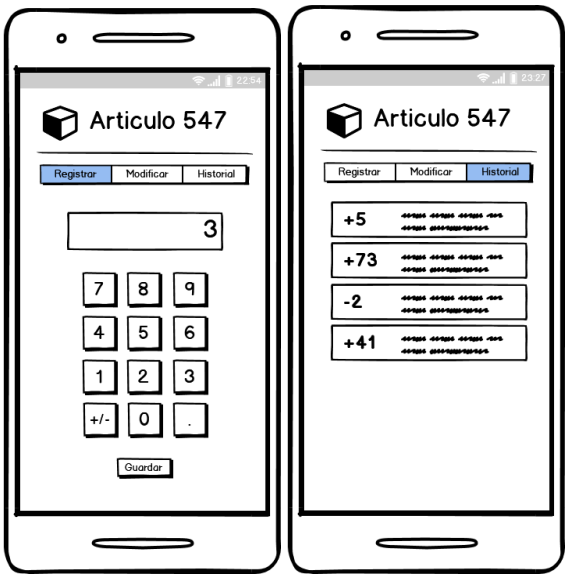
Una vez se accede al sistema el usuario deberá ingresar su nombre y su contraseña. Si cuenta con el perfil de administrador podrá consultar y modificar todos los usuarios de la aplicación.



En la pantalla principal se visualizarán los inventarios activos y se podrán crear nuevos. Cuando se selecciona un inventario, en la pantalla se visualizarán los artículos asociados. Desde aquí se podrá realizar la búsqueda de un articulo especifico.



Al seleccionar un artículo se ingresará a la pantalla desde la cual se podrán visualizar tres pestañas. En la primera se permite registrar la cantidad contada para el artículo. En la segunda pestaña se podrán realizar modificaciones al artículo seleccionado. Y desde la tercera se visualizarán todos los registros y cambios realizados sobre este artículo.



VI. ENTORNOS DE DESARROLLO Y DE OPERACIÓN

El software será escrito en Java y se desarrollará haciendo uso de Android Studio pues está pensado para ser un aplicativo móvil. Por lo ya mencionado este se ejecutará en dispositivos android (no necesariamente android puro).

VII. PROTOTIPO DE SOFTWARE INICIAL

Se ha realizado una primera versión de un prototipo de software inicial, en la cual hemos implementado algunas estructuras de datos que serán usadas posteriormente.

Además, el software desarrollado se registró en el siguiente repositorio de software Github[1]: https://git.io/JfYHB

En este prototipo fueron implementadas dos estructuras de datos lineales, es decir, una estructura en la cual cada dato tiene a lo sumo un antecesor y un sucesor, las estructuras implementadas fueron **Linked Lists** y **Dynamic Arrays** los cuales permiten realizar las siguientes funciones básicas, la lista completa de funcionalidades de cada estructura será expuesta posteriormente.

Linked Lists:

- Creación de la lista:
  Constructor de linkedListImp<E>();

- Inserción de un solo dato:
  pushFirst(E key)

[1] https://github.com/

- Añade al principio.

pushLast(E key)

- Añade al final.

- Actualización de un solo dato:
  changeFirst(E key)
  - Cambia el primer elemento por key.

  changeLast(E key)
  - Cambia el último elemento por key.

  change(E oldKey, E newKey)
  - Busca el elemento oldKey y si existe lo cambia por newKey

- Eliminación de un solo dato:
  popFirst()
  - Elimina el primer elemento.

  popLast()
  - Elimina el último elemento.

  delete(E key)
  - Busca el elemento key y si existe lo borra.

- Búsqueda de un solo dato:
  inList(E key)
  - Busca si el elemento key está en la lista

- Consulta de todos los datos:
  visit()
  - Imprime todos los elementos de la lista.

- Búsqueda parcial de datos:
  rangeSearch(int n)
  - Imprime los primeros n elementos de la lista.
- Ordenamiento:
  sort()
  - Ordena la lista.

  sorted()
  - Retorna una copia ordenada de la lista, no modifica la original.

- Almacenamiento de los datos

Dynamic Arrays:

- Creación de la lista:
  Constructor de dArrayImp<>()

- Inserción de un solo dato:
  add(E elem)
  - Añade elem al final del arreglo.

- Actualización de un solo dato:
  update(int index, E new_elem)
  - Cambia el elemento en la posición index por new_elem.

- Eliminación de un solo dato:
  delete(int index)
  - Elimina el elemento en la posición index y mueve una posición hacia atrás los elementos posteriores a esa posición.

  pop()
  - Elimina el último elemento.

- Búsqueda de un solo dato:
  get(int index)
  - Retorna el elemento en la posición index()

- Consulta de todos los datos:
  visit()
  - Imprime todo el arreglo.

- Búsqueda parcial de datos:
  rangeSearch(int i, int j)
  - Imprime el arreglo desde la posición i hasta la posición j.

- Ordenamiento:
  sort()
  - Ordena todo el arreglo.

  sortSubArr(int start, int end)
  - Ordena el arreglo desde la posición start hasta la posición end, modifica el arreglo original.

  sorted()
  - Retorna una copia ordenada del arreglo, no modifica el original.

- Almacenamiento de los datos

VIII. PRUEBAS DEL PROTOTIPO

La pruebas del prototipo se realizaron por separado para las linked lists y los Dynamic Arrays para las siguientes cantidades de datos:

- 10 mil datos,
- 100 mil datos,
- 500 mil datos,
- 1 millón de datos y
- 10 millones de datos

Para esta prueba usamos el método System.currentTimeMillis() para medir el tiempo de ejecución y el método Random.randInt() para llenar las estructuras, se usaron las funcionalidades que exigían un mayor costo computacional.

En las Linked Lists se usaron los métodos:

- sort(), O(nlogn).
- popLast(), O(n).
- inList(E key), O(n).

En los Dynamic Arrays se usaron los métodos:

- sort(), O(n^2).
- delete(int index), O(n).
- inList(E elem), O(n).

Se obtuvieron los siguientes resultados:

linkedList.sort()

Cantidad de elementos	Tiempo de ejecución

	(ms)
10.000	46 ms
100.000	rec. time exceeded
500.000	rec. time exceeded
1.000.000	rec. time exceeded
10.000.000	rec. time exceeded

linkedList.popLast()

Cantidad de elementos	Tiempo de ejecución (ms)
10.000	0 ms
100.000	1 ms
500.000	5 ms
1.000.000	7 ms
10.000.000	27 ms

linkedList.inList(E lastElement)

Cantidad de elementos	Tiempo de ejecución (ms)
10.000	0 ms
100.000	1 ms
500.000	2 ms
1.000.000	6 ms
10.000.000	28 ms

dynamicArray.sort()

Cantidad de elementos	Tiempo de ejecución (ms)
10.000	50 m s
100.000	350 ms
500.000	1318 ms
1.000.000	2508 ms
10.000.000	29930 ms

dynamicArray.delete(int index)

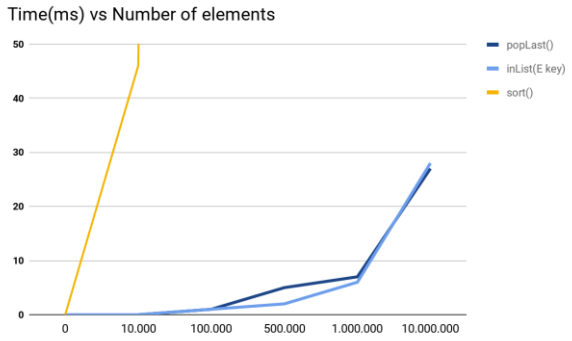
Cantidad de elementos	Tiempo de ejecución (ms)
10.000	0 ms
100.000	2 ms
500.000	6 ms
1.000.000	10 ms
10.000.000	97 ms

dynamicArray.inList(E lastElement)R

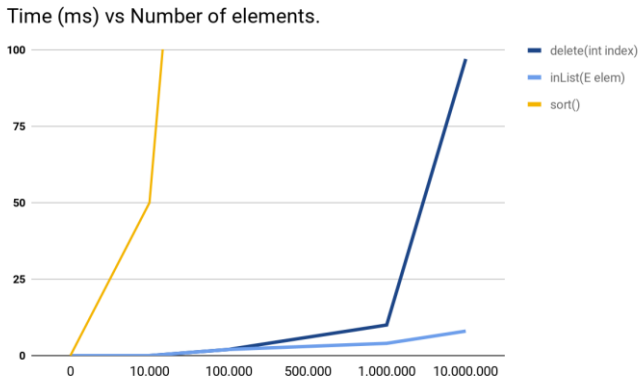
Cantidad de elementos	Tiempo de ejecución (ms)
10.000	0 ms
100.000	2 ms
500.000	3 ms
1.000.000	4 ms
10.000.000	8 ms

Gráfica comparativa

Linked Lists



Gráfica comparativa
Dynamic Arrays



I. ANÁLISIS DE COMPARATIVO

Linked List	
Method	**Big O**
isEmpty()	O(1)
len()	O(1)
pushFirst(E key)	O(1)
pushLast(E key)	O(1)
popFirst()	O(1)
popLast()	O(n)
changeFirst(E key)	O(1)
changeLast(E key)	O(1)
delete(E key)	O(n)
change(E oldKey, E newKey)	O(n)
rangeSearch(int n)	O(n)
visit()	O(n)
sort()	O(nlogn)
sorted()	O(nlogn)
inList(E key)	O(n)
setEmpty()	O(1)

Dynamic Array	
Method	**Big O**
rUp()	O(n)
rDown()	O(n)
add(E elem)	(rUp) ? O(n) : O(1)
delete(int index)	O(n)
pop()	(rDown) ? O(n) : O(1)
update(int index, E new_elem)	O(1)
get(int index)	O(1)
visit()	O(n)
rangeSearch(int i, int j)	O(n)
sortSubArr(int start, int end)	O(n^2)
sort()	O(n^2)
setEmpty()	O(1)
inList(E elem)	O(n)
getIndexOf(E elem)	O(n)
getLen()	O(1)
getSize()	O(1)
isEmpty()	O(1)

Explicación de notación:
(acción) ? a : b = si es necesario (acción) entonces a, si no b.

II. ROLES Y ACTIVIDADES

- Brayan Aponte
  Actividades:

- Mockups y descripción de la interfaz de usuario preliminar.
- Descripción del problema a resolver.
- Analisis requerimientos funcionales.
- Repositorio Github.

Roles:

- Investigador
- Observador
- Técnico

- Carlos Jiménez

  Actividades:

  - Aportes en implementación de Linked Lists y Dynamic Lists.
  - Corrección de errores en LL y DL.
  - Aporte en análisis comparativo y pruebas del prototipo en LL y DL.
  - Pruebas del prototipo y comparaciones entre estas.

  Roles:

  - Líder.
  - Experto.

- Santiago Londoño

  Actividades:

  - Aportes en implementación de Linked Lists y Dynamic Lists.
  - Corrección de errores en LL y DL.
  - Aporte en pruebas del prototipo en LL y DL.

  Roles:

  - Técnico.
  - Investigador.
  - Coordinador.

III. DIFICULTADES Y LECCIONES APRENDIDAS

La implementación de los java generics significó diversos problemas al momento de implementar los métodos por casteo de variables u objetos, sin embargo una vez solucionados estos errores aprendimos a utilizar mejor los java generics.

Además, presentamos problemas como grupo al momento de organizar las reuniones de trabajo, situación que se fue solucionando con el paso de los días.