Alberto Pickering

A01017746

Proyecto final

Se desarrolló una aplicación web utilizando Node.js, React y Neo4j para que un restaurante pueda administrar su inventario de materias primas y productos elaborados.

Manual de instalación

1. Descargar el proyecto de <https://github.com/alberto911/a01017746_bda18>.
2. Abrir una terminal, se debe tener instalado Docker y docker-compose.
3. Cambiarse al directorio raíz del proyecto, en donde se encuentra el archivo docker-compose.yml.
4. Ejecutar el comando:

docker-compose up

Se crearán dos contenedores:

* webapp: Basado en una imagen de Node, con el código y todos los módulos necesarios para que corra la aplicación.
* db: Basado en una imagen de Neo4j, carga la base de datos del directorio /neo4j/data.

La aplicación expone los siguientes puertos:

* + 8080: Web app
  + 7474: Servidor de Neo4j
  + 7687: Bolt

1. Abrir un navegador e ir a la dirección: localhost:8080 para acceder a la aplicación.
2. Al finalizar, correr docker-compose down para eliminar los contenedores.

Nota: La aplicación también está en <https://dry-coast-34430.herokuapp.com/>

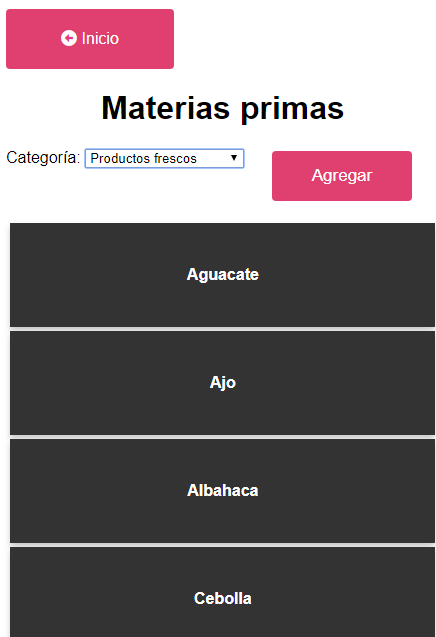
Manual de usuario

En la primera página de la aplicación se debe iniciar sesión. Los datos de la cuenta de prueba son:

Usuario: admin

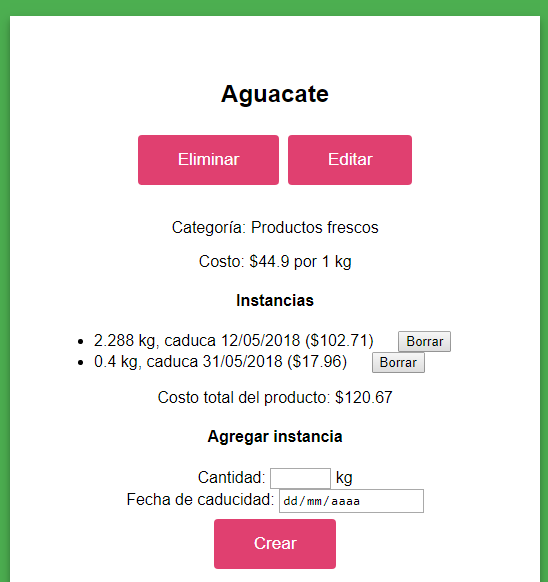
Contraseña: admin

Al ingresarlos correctamente, se muestra la página principal de la aplicación, en donde se selecciona cuál almacén se quiere visualizar. Analicemos primero el almacén de materias primas. En la parte izquierda se puede ver una lista con todas las materias primas que se han ingresado al sistema. Esta lista se puede filtrar según la categoría de los productos. Por ejemplo, en la siguiente imagen se muestran únicamente los productos frescos.



Si se hace clic en el botón Agregar, se desplegará la forma para crear una nueva materia prima en el sistema. Para volver a la página principal de materias primas, se debe oprimir el texto ‘Materias Primas’. Al seleccionar un elemento de la lista, se mostrarán todos sus detalles en el panel de la derecha. Por ejemplo, en la siguiente imagen se muestran los detalles del aguacate, incluyendo su precio, las unidades en que se mide y la categoría a la que pertenece. Además, se muestra una lista de sus instancias, es decir, de todos los aguacates que hay en el inventario.

De cada instancia se puede ver la cantidad, fecha de caducidad y el costo (calculado a partir del costo de la materia prima). El costo total del producto es la suma del de todas sus instancias. Si se hace clic en el botón de Borrar, se elimina esa instancia del inventario. En la parte inferior hay una forma para agregar nuevas instancias al inventario. Basta con indicar la cantidad y la fecha de caducidad, y hacer clic en Crear. Si se oprime el botón Eliminar, se borrará la materia prima y todas sus instancias del sistema. Sin embargo, cuando la materia prima aparece en una receta, sólo se borran sus instancias.

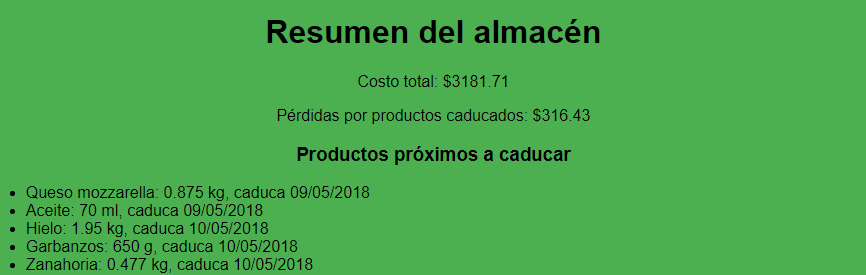


Si se desean modificar los datos de una materia prima, se hace clic en Editar. Por ejemplo, para cambiar el costo del aguacate, se muestra el formulario con los datos actuales y sólo se modifican el campo correspondiente (44.9 por 50 en este caso). Después de hacer clic en Enviar, se muestran nuevamente los detalles del producto con los cambios realizados.



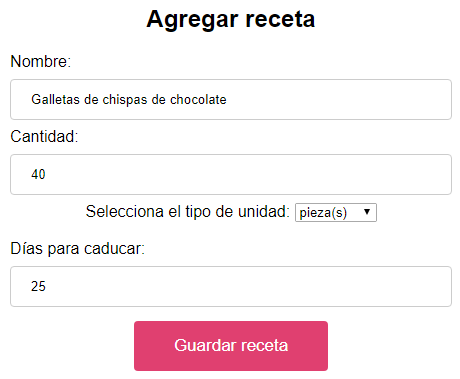
En la página principal de materias primas se muestra un resumen del almacén en el panel derecho. Este resumen tiene tres datos principales:

* Costo total del almacén: Calculado a partir del inventario disponible.
* Pérdidas por productos caducados: Cada día, a medianoche, el sistema elimina automáticamente del inventario todas las materias primas que caducaron el día anterior. Aquí se muestra el costo de todas las materias primas que se han tenido que tirar.
* Materias primas próximas a caducar: Se muestra una lista de las materias primas que caducarán en los próximos tres días, para que los administradores lo tomen en cuenta y las utilicen lo más pronto posible.



El almacén de productos elaborados sigue la misma lógica que el de materias primas. Aquí únicamente exploraremos las diferencias. Un producto elaborado es el equivalente a una instancia de una receta. Por lo tanto, para elaborar productos primero se necesita tener una receta.

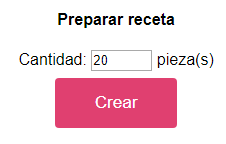
El formulario para agregar recetas se divide en dos secciones. En la primera, se introducen datos generales como el nombre, cuánto rinde y cuántos días dura después de prepararse. Por ejemplo, la siguiente imagen muestra cómo crear una receta de galletas. Se puede hacer clic en Guardar Receta después de llenar estos campos, pero aún no hemos agregado los ingredientes.



Para agregar ingredientes a la receta utilizamos la parte inferior del formulario. Se debe seleccionar una categoría, una materia prima y la cantidad que requiere la receta. Se pueden agregar tantos ingredientes como sea necesario. Si se desea eliminar un ingrediente de la receta, basta con hacer clic en Quitar. Al finalizar de insertar la lista, se puede guardar la receta completa.



Al visualizar los detalles de una receta, se muestran sus ingredientes y el costo calculado a partir de las cantidades y de los costos de las materias primas. Además, se muestra una lista de productos elaborados, con cantidad, costo y fecha de caducidad (calculada automáticamente a partir de la receta y la fecha en la que se creó). Para elaborar un producto, sólo es necesario introducir la cantidad en el formulario correspondiente. El sistema verifica que haya suficientes materias primas para producirlo. De ser así, reducirá el inventario de materias primas según las cantidades indicadas en la receta y la cantidad preparada. Por ejemplo, si la receta de galletas produce 40 piezas, pero sólo se quiere hacer medio tanto, se introduce 20 en el formulario, como se muestra en la siguiente imagen.



El resumen del almacén de productos elaborados es igual que el de materias primas. Para salir del sistema, se debe hacer clic en el botón Logout en la página de inicio.

Descripción del trabajo realizado

Primero se creó el esqueleto de la aplicación utilizando Node.js y Express y se decidió utilizar una arquitectura MVC. Se utilizó la librería Seraph que se conecta con Neo4j y permite ejecutar consultas desde javascript. Además, tiene la ventaja de que permite definir modelos y sus composiciones. Se definieron modelos con las siguientes características:

* Materia Prima: Son todos los nodos con la etiqueta :MateriaPrima, que además tienen otra etiqueta con la categoría (por ejemplo :Carne).
* Materia Prima Instance: Estos nodos tienen la etiqueta :MateriaPrimaInstance y contienen la cantidad y fecha de caducidad como propiedades. Existen relaciones (HAY) entre materias primas e instancias de la forma:

(:MateriaPrima)-[:HAY]->(:MateriaPrimaInstance)

* Receta y Producto Elaborado: Siguiendo el mismo patrón, es decir:

(:Receta)-[:HAY]->(:ProductoElaborado)

* CONTIENE: Es la relación entre recetas y materias primas. La cantidad del producto que requiere la receta se almacena como propiedad de la relación:

(:Receta)-[:CONTIENE {cantidad: 10}]->(:MateriaPrima)

* Usuario: Nodos para almacenar el nombre de usuario y la contraseña necesarios para la autenticación y autorización.
* Caducado: Dos nodos (uno por cada almacén) que van guardando el total de las pérdidas por productos caducados. El total se actualiza automáticamente a medianoche cuando se eliminan los productos caducados.

Después de definir los modelos, se creó una API para realizar las operaciones CRUD sobre los mismos. Se establecieron las rutas correspondientes a cada operación y se crearon controladores para responder en formato JSON a las peticiones HTTP del usuario. También se crearon las demás consultas, por ejemplo:

* Productos por caducar: Se buscan todas las instancias cuya fecha de caducidad es menor a la fecha actual más tres días.
* Costo de una receta: Se recorren todas las relaciones CONTIENE de una receta y se multiplica la cantidad por el costo de la materia prima. Al final se regresa la suma de estos productos.

Se creó el front end utilizando React y su esquema de componentes anidados. En cada almacén hay un componente principal llamado ProductList que despliega la lista con todos los productos y que se actualiza cuando el usuario agrega, edita o borra productos. Los detalles de un producto se muestran con el componente ProductDetail, que, a su vez, contiene un componente para mostrar la lista de instancias, y así sucesivamente. Todos los componentes son hijos del componente padre llamado App. Se utilizó React Router para manejar todas las rutas client side y evitar que la página se recargue con cada petición. Así, toda la aplicación parece correr en una sola página.

Para el deployment, se generó un bundle con Webpack para producción, el cual está disponible de manera estática en el servidor. Se creó un Dockerfile para tomar una imagen de Node e instalar todos los módulos necesarios, especificados en el archivo package.json del proyecto. Luego se copian todos los archivos del proyecto y se inicia el servidor. También se creó una imagen de Neo4j y se pobló la base de datos con recetas reales. Finalmente, se creó el archivo docker-compose.yml para conectar ambos contenedores.