



Sala Comunitaria de Elaboración de Productos

Java y Aplicaciones Avanzadas sobre Internet

Gabriel Di Marco - 1870/2



El problema

Se trabajó en el desarrollo de una aplicación web que permite gestionar el funcionamiento de la Sala Comunitaria de la Facultad de Ciencias Veterinarias de la Universidad Nacional de La Plata. La aplicación, utilizada por el personal de la sala, permite a los usuarios verificar el stock de los productos, agregar las compras de insumos y materias primas, verificar las familias productoras que proveen dichos insumos y materias primas, la elaboración y venta de productos, mostrar recetas que utilizan estos productos.

La aplicación se divide en el backend, diseñado con Java, y un frontend, diseñado con el framework Angular. Esta se conecta a una base de datos relacional, manejada mediante SQL.

El desarrollo de la aplicación se realizó en etapas, iniciando por el planteo de conceptos y bocetado, procediendo después al diseño del backend y la base de datos, mientras que el frontend se dejó para el final.

Etapas de diseño





Etapa 1: análisis, diseño y maquetado del sistema

Al inicio del proyecto, se diseñó un documento en el que se detallan las historias de usuario, es decir, una explicación general e informal de una función de software que debe proveer el sistema desde el punto de vista de un usuario que lo utilizará.

También se realizaron una serie de bocetos de las posibles vistas de la aplicación. Estas son ideas conceptuales que difieren del resultado final, se trata de una proposición inicial de la visualización de la aplicación.



Historias de usuario

0 of 0 points		1 • 12 - 18 Aug •  100%	
★	Ordenar stock	   	<input type="checkbox"/>
★	Actualizar perfil	   	<input type="checkbox"/>
★	Ver perfil	   	<input type="checkbox"/>
★	Mantener sesión	   	<input type="checkbox"/>
★	Restablecer contraseña	   	<input type="checkbox"/>
★	Registro	   	<input type="checkbox"/>
★	Buscar elementos	   	<input type="checkbox"/>
★	Eliminar elemento	   	<input type="checkbox"/>
★	Mostrar stock	   	<input type="checkbox"/>
★	Calcular requerimiento	   	<input type="checkbox"/>
★	Iniciar sesión	   	<input type="checkbox"/>
★	Cerrar sesión	   	<input type="checkbox"/>
★	Agregar elemento	   	<input type="checkbox"/>
★	Modificar elemento	   	<input type="checkbox"/>
★	Modificar stock	   	<input type="checkbox"/>

Vistas iniciales



[Cerrar sesión](#) [Perfil](#)

Datos personales	Datos de sesión
<input type="text" value="Nombre"/>	<input type="text" value="Usuario"/>
<input type="text" value="Apellido"/>	<input type="text" value="Contraseña"/>
<input type="text" value="Mail"/>	<input type="text" value="Confirmar contraseña"/>
<input type="button" value="Guardar cambios"/>	<input type="button" value="Convertir a administrador"/>
	<input type="button" value="Cambiar contraseña"/>



ACCESO A LA BASE DE DATOS

<input type="text" value="Usuario"/>
<input type="text" value="Contraseña"/>
<input type="button" value="Ingresar"/>



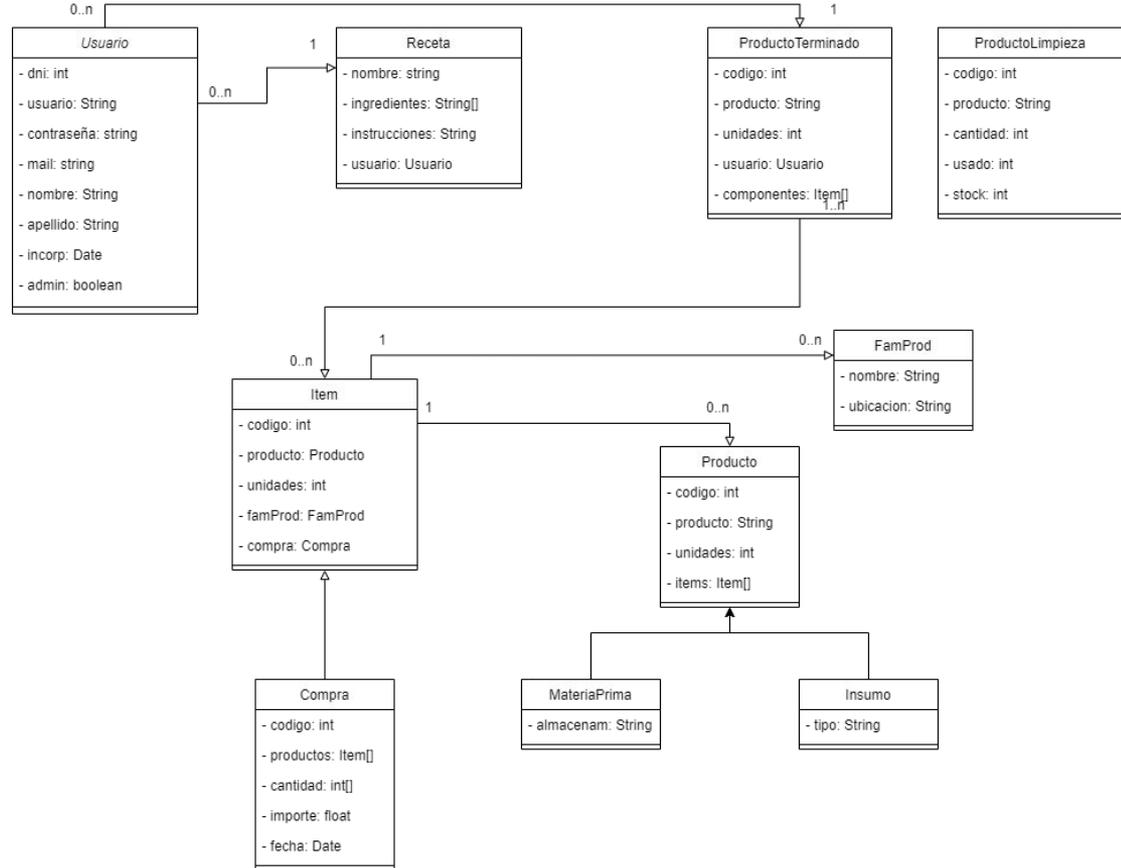
Etapa 2: definición de los objetos del modelo

En esta etapa se comenzó a trabajar en la construcción del software, diseñando los elementos que componen la aplicación y las relaciones entre estos.

Se inició realizando un diagrama de clases utilizando draw.io, en el cual se ven los distintos elementos (clases), sus atributos y las relaciones entre estos. El diagrama muestra una propuesta inicial del diseño, por lo que puede diferir del resultado final.

Posteriormente, se diseñaron las clases en Java utilizando Eclipse. Los archivos en los que se definen los objetos se guardaron en el paquete “model”

Diagrama de clases





Etapa 3: desarrollo de la capa de persistencia

En esta etapa se trabajó en la capa de persistencia de la aplicación, la cual cumple la función de mediador entre el sistema y la base de datos que preserva la información.

Esta se realizó mediante el patrón DAO, que proporciona una capa de abstracción entre capa de negocio y la capa de acceso a datos. Implementando una interfaz que encapsula la lógica de acceso a datos, así como métodos para crear, leer, actualizar y eliminar estos datos.

Se usó el framework Java Hibernate para relacionar los elementos de la base de datos con los objetos implementados en el backend.

Mediante Maven se implementan las dependencias necesarias a partir de este punto del proyecto.

Los archivos en los que se definen la interfaz y la implementación se guardaron en el paquete “dao”.



Etapa 4: desarrollo de servicios REST

En esta etapa se utilizó el framework Java Jersey para desarrollar una API REST, lo que permite realizar las operaciones en la base de datos mediante los métodos HTTP. Los recursos se identifican mediante URIs, usando las anotaciones de la API JAX-RS para asociar estas URIs a los métodos de Java.

La API REST se integró a la capa de persistencia mediante inyección de dependencias usando el framework HK2.

Usando Postman fue posible probar el correcto funcionamiento de las solicitudes HTTP.

Los archivos donde se definen las URIs y los métodos HTTP están en el paquete “resources”.



Etapa 5: desarrollo de la vista del proyecto

En esta etapa se trabajó en el frontend de la aplicación, la parte visible con la cual los usuarios pueden interactuar con el sistema. Se utilizó el framework Angular para construir una SPA (Single-page Application), la cual permite acceder a las distintas secciones de la aplicación mediante ruteo.

También se implementaron servicios que dan acceso a los datos. Inicialmente se implementaron datos estáticos creados en los propios servicios para realizar pruebas. Posteriormente se realizaría la conexión a la base de datos.

Durante las pruebas hubo dificultades a la hora de acceder a la base de datos debido a la política CORS. Se pudo solucionar incluyendo un filtro a las solicitudes HTTP que agrega al encabezado las autorizaciones necesarias.



Etapa 6: sistema completo

Finalmente, en esta etapa se realizó la conexión entre frontend y backend. Mediante los servicios de la API REST diseñada en la etapa 4, la aplicación es capaz de acceder a la base de datos y obtener, modificar o eliminar los elementos guardados y mostrarlos en pantalla. Por motivos de conservación de información, la aplicación no elimina realmente ningún elemento de la base de datos, sino que utiliza un atributo específico para “invalidar” el elemento, y que no se muestre a los usuarios.

La aplicación también implementa un sistema de inicio de sesión, que verifica la identidad de los usuarios directamente desde la base de datos para comprobar que los datos sean correctos. La sesión se mantiene mediante un token obtenido usando la librería JWT, y permite pasar la guardia, la cual de no tener el token, redirige a la página de inicio de sesión.

Organización del grupo



Organización del grupo

Al realizar el trabajo solo, se me asignó una carga de trabajo menor a la de los grupos de dos y tres integrantes. Se me pidió implementar las operaciones ABM (alta, baja y modificación) y la consulta de los usuarios de la aplicación, las materias primas y las familias productoras. Debido a esto hay gran diferencia con los bocetos de la etapa 1, ya que muchos de los elementos no están presentes en el resultado final.

Conclusiones



Conclusiones

Durante la materia se aprendió sobre el desarrollo de Aplicaciones Web Dinámicas en Java, incluyendo el uso de los Servlets, generalmente obsoletos, el manejo de sesiones, la persistencia de datos, la interacción entre frontend, backend y la base de datos. Al evitar usar Spring Boot y realizar la implementación manual de las funciones necesarias, fue posible aprender en detalle el funcionamiento de los frameworks y librerías que Spring automatiza.