Tema 3. Acceso a bases de datos con ORM

Boletín 1. Kanban

Gestión de tareas con Kanban

Desarrollar una aplicación Java de consola que permita a un usuario autenticarse y gestionar un tablero Kanban usando Hibernate (JPA). La aplicación debe persistir datos en PostgreSQL y trabajar con entidades relacionadas.

1. Requisitos funcionales

1.1. Autenticación

Al iniciar la aplicación, mostrará el siguiente menú:

```
******************
***    KANBAN    ***
******************
1) Iniciar sesión
2) Registrarse
0) Salir
```

El usuario se identifica con:

- email (único)
- password (se almacena el **hash*)

Condiciones:

- Registro: no permitir emails repetidos.
- Login: si usuario o contraseña incorrectos, mostrar error y reintentar.
- Tras login correcto, entrar al menú principal del Kanban.

Nota: Por motivos de seguridad, nunca debemos guardar en una base de datos las contraseñas. En su lugar guardaremos el resultado de aplicar una función hash al password original, que es el resultado de aplicar una función matemática que no tiene inversa. Así si la base de datos se viese comprometida, los atacantes no podrían saber el password de los usuarios. Para crear el hash, lo recomendado sería utilizar BCrypt pero no tiene soporte para Java, así que vamos a utilizar **PBKDF2 que sí soporte y es una buena alternativa.*

Obtención de un hash más seguro

SHA-256 es seguro criptográficamente, pero demasiado rápido para contraseñas por lo que:

- Un atacante con GPU puede probar millones de contraseñas por segundo.

- Puede usar diccionarios o tablas precalculadas.

Por tanto necesitamos un mecanismo que dificulte estos ataques.

Utilizando Salt

Un salt es un valor aleatorio que se añade a la contraseña antes de calcular el hash.

Caracteristicas de un salt:

- No es secreto
- Es distinto para cada usuario
- Se guarda junto al hash

¿Por qué necesitamos el salt?

- Evita que dos usuarios con la misma contraseña tengan el mismo hash.
- Impide el uso de tablas precalculadas.
- Obliga al atacante a atacar cada contraseña individualmente

Generando un salt de 16 bytes

```
SecureRandom random = new SecureRandom();
byte[] salt = new byte[16]; // 128 bits
random.nextBytes(salt);
```

¿Qué es PBKDF2?

PBKDF2 es una función derivadora de claves (KDF) diseñada para contraseñas.

Hace dos cosas clave:

- Aplica un hash muchas veces (iteraciones)
- Usa un salt

La importancia de las iteraciones

Las iteraciones indican cuántas veces se repite internamente el proceso de hash.

- Más iteraciones implican más tiempo de cálculo
- Login un poco más lento (milisegundos)
- Ataque masivo muchísimo más lento

Creando el hash de una contraseña

```
int iterations = 100_000;
int keyLength = 256;

PBEKeySpec spec = new PBEKeySpec(
        password.toCharArray(),
        salt,
```

```
        iterations,
        keyLength
);

SecretKeyFactory skf =
        SecretKeyFactory.getInstance("PBKDF2WithHmacSHA256");

byte[] hash = skf.generateSecret(spec).getEncoded();
```

¿Qué se guarda en la base de datos?

Nunca basta con guardar solo el hash.

Se debe guardar todo lo necesario para verificar la contraseña:

- algoritmo
- número de iteraciones
- salt
- hash

Ejemplo:

```
PBKDF2WithHmacSHA256:100000:saltBase64:hashBase64
```

Verificación de la contraseña

- Se recupera el salt y las iteraciones
- Se recalcula el hash
- Se comparan ambos hashes

```
boolean ok = MessageDigest.isEqual(hashCalculado, hashGuardado);
```

2.2 Menú principal (depués del login)

A) Gestión de Tableros

- Crear tablero
- Listar mis tableros
- Eliminar tablero (con sus columnas y tarjetas)

B) Gestión de Columnas (dentro de un tablero)

- Crear columna en un tablero
- Listar columnas del tablero
- Renombrar columna
- Eliminar columna (con sus tarjetas)

C) Gestión de Tarjetas (dentro de una columna)

- Crear tarjeta (título obligatorio, descripción opcional)
- Listar tarjetas por columna
- Mover tarjeta a otra columna del mismo tablero
- Editar tarjeta (título/descripcion)
- Eliminar tarjeta

D) Gestión de Etiquetas

- Crear etiqueta
- Listar etiquetas
- Asignar etiqueta a tarjeta
- Quitar etiqueta de tarjeta
- Listar tarjetas por etiqueta (solo del usuario autenticado)

E) Salir (logout)

- Cerrar sesión y volver al menú de login.

3. Modelo de datos (propuesta)

Usuario

- id, email (único), passwordHash

Tablero

- id, nombre, usuario_id

Columna

- id, nombre, tablero_id

Tarjeta

- id, titulo, descripcion, columna_id

Etiqueta

- id, nombre

Según este modelo de datos, las etiquetas son globales (compartidas por todos los usuarios).

4. Reglas de negocio mínimas

- Un tablero recién creado tendrá 3 columnas por defecto: “TODO”, “DOING”, “DONE”.
- No se permite mover una tarjeta a una columna de otro tablero.

- Al borrar tablero, se borran columnas y tarjetas asociadas (cascada).
- El nombre de una etiqueta no debe duplicarse (unicidad).

5. Entregables

- Fichero README con URL del repositorio
- Script SQL con datos de prueba (usuario, tableros, etc...)
- Contraseña del usuario de pruebas

6. Criterios de evaluación

- Modelo y relaciones (25%): mapeos correctos 1–N y N–N, claves, constraints.
- Autenticación (20%): registro/login correcto, hash de password.
- Funcionalidad de la aplicación Kanban (50%): CRUD + mover tarjeta + etiquetas.
- Organización y estructura del código (10%)