Standard Creación de test HorApp

Base de Datos para Test

Como lo indica el archivo application-test.properties el cual representa el ambiente de test, se debe crear en PgAdmin una base de datos llama horapp_test_db, la cual será la que se utilizara en la ejecución de los test para no interferir con horapp_db que es la db principal.

Estructura de carpetas

Por cada implementación de servicio se debe crear una carpeta con el mismo nombre del servicio con el sufijo Test, ejemplo: para el servicio ScheduleService se debe crear una carpeta en test con el nombre ScheduleServiceTest.

Scripts

Por cada entidad nueva a probar se debe crear su propio script de imports SQL en la carpeta resources/db_templates_test con el nombre EntidadUseCaseTestInserts.sql, ejemplo: para la entidad Schedule de debe crear un import llamado ScheduleUseCaseTestInserts.sql.

En este script se coloca cualquier tipo de sentencia SQL que se necesite para realizar los tests de integración para el caso de uso de esa entidad.

Se deben insertar datos independientes para cada test, es decir, no se deben usar los mismos registros insertados para testear dos métodos distintos (si se usan los mismos registros en distintos escenarios del mismo método por ej: caso positivo y negativo, pero no para métodos distintos). Separar de la siguiente manera con comentarios.

```sql
-- SAVE ----

INSERT INTO courses (id_course, course_name, deleted) VALUES
 (100000 ,'AM1', false);

INSERT INTO schedules (id_schedule, course_group, id_course) VALUES
 (100000, '1K1', 100000);

---------
-- FIND BY ID -------

INSERT INTO courses (id_course, course_name, deleted) VALUES
 (100001 ,'AM1', false);

INSERT INTO schedules (id_schedule, course_group, id_course) VALUES
 (100001, '1K1', 100000);

INSERT INTO days_and_times (id_day_and_time, end_time, start_time, id_schedule, day_of_week) VALUES
 (100000, '10:00', '08:00', 100001, 'MONDAY');
```

También se debe mantener actualizado el archivo DeleteAllTemplateSqlTest.sql, en el cual se tienen que eliminar todos los datos de la base de datos en orden para que no haya problemas de foreign keys.

Clase de Test

En cada clase de test se debe agregar la siguiente configuración:

```java
Cabaleiro Nicolas *
@SpringBootTest
@ActiveProfiles("test")
@TestInstance(TestInstance.Lifecycle.PER_CLASS)
@Sql(scripts = {"/db_templates_test/NombreDelScriptParaEsteCasoDeUso.sql"}, executionPhase = Sql.ExecutionPhase.BEFORE_TEST_CLASS)
@Sql(scripts = {"/db_templates_test/DeleteAllTemplateSqlTest.sql"}, executionPhase = Sql.ExecutionPhase.AFTER_TEST_CLASS)
class EntityTest {
```

De manera que se cargue el contexto de spring en los test, se utilice la configuración en el perfil de ‘test’ como la base de datos de test, para que se cree una sola instancia de la clase en todos los test de esa clase, para que se ejecute el script de insert antes de los test y que se ejecute el script de delete luego de la ejecución de los test

La injeccion de dependencias se debe realizar mediante la anotación @Autowired

Nombrado de métodos

Cada método de test se debe estar anotado con @Testy se debe nombrar con el nombre real del método seguido del sufijo Test y luego se puede agregar un guion bajo con una breve descripción del contexto de ese test, es decir, una aclaración de si es el test cuando sale bien, cuando sale mal, etc, ya que se deben incluir todo los casos posibles

Ejemplo: para el test del método findById() puedo crear los siguientes tests

findByIdTest_Successful() que seria cuando se encuentra la entidad.

findByIdTest_WhenEntityDoesNotExists() que seria un caso donde el test falla a propósito sin encontrar la entidad.

Alcance

Todos los métodos que interactúen con dependencias externas mediante inyección de dependencias que no sea un Repositorio se deben probar de manera **UNITARIA**, es decir que se debe mockear o simular el comportamiento de esa dependencia sin realizar una interacción real.

Si un método interactúa directamente con un Repositorio, este mismo debe tener un test de **INTEGRACIÓN**, probando la comunicación real con la base de datos mediante los scripts SQL previamente mencionados.