

Borrador

Acceso a Datos

Práctica 1

Curso 2025-2026

Observaciones

- Componentes. Nº máximo de personas para desarrollar la práctica: 2.
- Las fechas de entrega y su posterior defensa serán las indicadas en clase y/o plataforma.
- No se permitirá el envío de la práctica pasada la fecha de entrega.
- Se establecerá una fecha de defensa donde se procederá a la calificación utilizando la rúbrica indicada en el presente documento.

En el caso de que un alumno no esté presente en el momento de la defensa la práctica no estará superada.

- El proyecto deberá ser exportable, permitiendo ser ejecutado en cualquier dispositivo. Es necesario que, en el código, se manejen rutas relativas y que partan de la ubicación del proyecto.
- Se utilizará Java como lenguaje de programación.

Objetivos

Desarrollar una aplicación de acceso a datos usando Modelo guiado por dominio (DDD), usando el patrón Repositorio [1] y ficheros como persistencia de datos:

Instrucciones de desarrollo de la práctica

Desarrolle una aplicación de acceso a datos usando:

- Modelo guiado por dominio (DDD)
- Modelo de dominio
- Raíces agregadas
- Objetos valor
- Repositorio el patrón repositorio bajo el modelo
- Ficheros como persistencia para los datos.

Existen dos interfaces que deberán implementar los repositorios (no es posible modificarlos):

```
public interface IRepository<T, ID> {  
    /**  
     * Devuelve el número de entidades.  
     * @return número de entidades
```

Práctica 1. Acceso a datos

```
    */

    long count();

    /**
     * Borra la entidad con identificador id.
     * Si no se encuentra no realiza ninguna acción
     *
     * @param id      identificador de la entidad. No debe ser nulo
     *
     * @throws IllegalArgumentException En caso de ser id nulo
     */
    void deleteById(ID id);

    /**
     * Borra todas las entidades del repositorio.
     */
    void deleteAll();

    /**
     * Devuelve true si existe la entidad con identificador id.
     * @param id      Identificador de la entidad
     * @return        true si existe la entidad con el identificador
id
     *
     * @throws IllegalArgumentException En caso de ser id nulo
     */
    boolean existsById(ID id); //

    /**
     * Devuelve la entidad T con identificador id.
     * @param id      Identificador de la entidad
     * @return        Entidad que tiene como identificador id.
     *
     * @throws IllegalArgumentException En caso de ser id nulo
     */
    T findById(ID id);

    /**
     * Devuelve todas las instancias
     * @return Todas las instancias
     */
    Iterable<T> findAll();

    /**
     * Guarda la entidad entity.
     * @param entity  entidad a guardar. No debe ser nulo
     * @return        entidad guardada
     *
     * @throws IllegalArgumentException En caso de ser entity nulo
     */
    <S extends T> S save(S entity);
}

public interface IRepositoryoExtend<T, ID> extends
    IRepositoryo<T, ID> {
```

Práctica 1. Acceso a datos

```
/**
 * Devuelve la entidad T con identificador id.
 *
 * @param id      Identificador de la entidad
 * @return        Entidad que tiene como identificador id u
Optional#empty() si no se encuentra
 *
 * @throws IllegalArgumentException En caso de ser id nulo
 */
Optional<T> findByIdOptional(ID id);

/**
 * Devuelve todas las instancias de tipo T
 *
 * @return        todas las instancias.
 * */
List<T> findAllToList();
}
```

- Implemente una aplicación donde se use el patrón repositorio [1], incluyendo raíces agregadas, y como persistencia de datos, ficheros, usando las interfaces anteriores para poder usar operaciones CRUD.
- Cada repositorio deberá, además de implementar las interfaces anteriores, al menos un método propio que pertenezca semánticamente a dicho repositorio.
- La elección de tipo de ficheros y estructura del contenido será el que desee.
- La práctica deberá tener, en su análisis conceptual de clases UML, al menos dos clases de entidades de dominio, con al menos una relación uno-a-muchos. En caso de usar más, todas las clases deberán estar relacionadas.

Rúbrica

Consulta de registros	<p>No emplea, no comprende o contiene errores en la consulta de los registros.</p> <p>(0 puntos)</p>	<p>Utiliza, usando dos clases de entidad, los siguientes métodos de la interfaz IRepository en todos los repositorios:</p> <ul style="list-style-type: none"> - long count(); - boolean existsById(ID id); - T findById(ID id); - Iterable<T> findAll(); <p>(1 punto)</p>	<p>Utiliza, usando tres o más clases de entidad, los siguientes métodos de la interfaz IRepository en todos los repositorios:</p> <ul style="list-style-type: none"> - long count(); - boolean existsById(ID id); - T findById(ID id); - Iterable<T> findAll(); <p>(1 punto)</p>
Consulta de registros de IRepositoryExtend<T, ID>		<p>Utiliza, usando dos clases de entidad, los siguientes métodos de la interfaz IRepository en todos los repositorios:</p> <ul style="list-style-type: none"> - Optional<T> findByIdOptional(ID id); - List<T> findAll(); <p>(1 punto)</p>	<p>Utiliza, usando tres o más clases de entidad, los siguientes métodos de la interfaz IRepository en todos los repositorios:</p> <ul style="list-style-type: none"> - Optional<T> findByIdOptional(ID id); - List<T> findAll(); <p>(1 punto)</p>

Práctica 1. Acceso a datos

Inserción de registros	<p>No emplea, no comprende o contiene errores en la inserción de registros</p> <p>(0 puntos)</p>	<p>Utiliza, usando dos clases de entidad, el siguiente método de la interfaz IRepository en todos los repositorios:</p> <ul style="list-style-type: none"> - <S extends T> S save(S entity); <p>(Se recuerda que una actualización de un objeto consiste en buscarlo y, posteriormente, guardarlo).</p> <p>(1 punto)</p>	<p>Utiliza, usando tres o más clases de entidad, el siguiente método de la interfaz IRepository en todos los repositorios:</p> <ul style="list-style-type: none"> - <S extends T> S save(S entity); <p>(1 punto)</p>
Borrado de registros	<p>No emplea, no comprende o contiene errores el borrado de registros usando ficheros.</p> <p>(0 puntos)</p>	<p>Utiliza, usando dos clases de entidad, el siguiente método de la interfaz IRepository en todos los repositorios:</p> <ul style="list-style-type: none"> - void deleteById(ID id); <p>(2 puntos)</p>	<p>Utiliza, usando tres o más clases de entidad, el siguientes método de la interfaz IRepository en todos los repositorios:</p> <ul style="list-style-type: none"> - void deleteById(ID id); <p>(2 puntos)</p>

Instrucciones de entrega

- Comprima, en formato zip, el proyecto con el código fuente respetando los directorios.
- El nombre de cada fichero será:
 - En el caso de realizar el proyecto una sola persona:
`25-26_ad_practical_apellidosAlumno1_nombreAlumno1.zip`
 - En caso contrario:
`25-26_ad_practical_apellidosAlumno1_nombreAlumno1#apellidosAlumno2_nombreAlumno2.zip`
- Suba a la plataforma los ficheros antes de la fecha de finalización.

Referencias

[1] Modelo repositorio.

<https://aulavirtual35.educa.madrid.org/ies.lapaloma.madrid/mod/book/view.php?id=131520&chapterid=3748>