Examen

Devuelve	Método y Descripción
List<T>	findAll()
List<T>	findAll(Sort sort)
List<T>	findAllById(Iterable<ID> ids)
void	flush() Flushes all pending changes to the database.
T	getReferenceById(ID id) Returns a reference to the entity with the given identifier. *(sustituye a getById y getOne, que están deprecated)*
<S extends T> List<S>	saveAll(Iterable<S> entities)
<S extends T> List<S>	saveAllAndFlush(Iterable<S> entities) Saves all entities and flushes changes instantly.
<S extends T> S	saveAndFlush(S entity) Saves an entity and flushes changes instantly.

Devuelve	Método y Descripción
long	count() Returns the number of entities available.
void	delete(T entity) Deletes a given entity.
void	deleteAll() Deletes all entities managed by the repository.
void	deleteById(ID id) Deletes the entity with the given id.
boolean	existsById(ID id) Returns whether an entity with the given id exists.
Iterable<T>	findAll() Returns all instances of the type.
Optional<T>	findById(ID id) Retrieves an entity by its id.
<S extends T>	save(S entity) Saves a given entity.
<S extends T> Iterable<S>	saveAll(Iterable<S> entities) Saves all given entities.

resultados. Pueden ser varios unidos por "And" o "Or". Por convención, debemos "camelcase", es decir, el principio de cada palabra (incluyendo los nombres de empezar por mayúscula y el resto en minúscula. Ejemplos:

```
List<Empleado> findByNombre (String nombre);
List<Empleado> findByNombreAndEmail (String nombre, String email);

List<Empleado> findFirst3ByNombre (String nombre);
```

Obviamente la consulta puede recibir parámetros empleado el símbolo la interrogación "?" y el número de parámetro. Ejemplo:

```
@Query("select e from Empleado e where e.nombre=?1 and e.email=?2")
Empleado obtenerEmpleadoPorNombreYEmail (String nombre,String email);
```

```java
@Service
public class EmpleadoServiceImplBD implements EmpleadoService {
    @Autowired
    EmpleadoRepository repositorio;

    public Empleado añadir (Empleado empleado) {
           return repositorio.save (empleado);
    }

    public List<Empleado> obtenerTodos() { return repositorio.findAll ();  }

    public Empleado obtenerPorId (long id) {
        return repositorio.findById (id).orElse(null);
        // findById de JpaRepository devuelve un Optional. Para simplificar,
        // y que el servicio siga devolviendo Empleado y no Optional<Empleado>
        // hacemos que si no lo encuentra devuelva null.
    }

    public Empleado editar (Empleado empleado) {
           return repositorio.save (empleado);
    }

    public void borrar(Long id) {
        repositorio.deleteById (id);
    }
}
```

agregar dependencias h2
agregar @Entity, @Id, @Data @NoArgsConstructor @AllArgsConstructor
@EqualsAndHashCode(of="id") @GeneratedValue en el campo que corresponda

añade una interfaz = public interface productoRepository extends JpaRepository <Producto, Long>{}

crear el servicio ServiceEmpleadoImplBD

@Service
public class EmpleadoServiceImplBD implements EmpleadoService {
@Autowired
EmpleadoRepository repositorio;
public Empleado añadir (Empleado empleado) {
return repositorio.save (empleado);
}
public List<Empleado> obtenerTodos() { return repositorio.findAll (); }
public Empleado obtenerPorId (long id) {
return repositorio.findById (id).orElse(null);
// findById de JpaRepository devuelve un Optional. Para simplificar,
// y que el servicio siga devolviendo Empleado y no Optional<Empleado>
// hacemos que si no lo encuentra devuelva null.
}
public Empleado editar (Empleado empleado) {
return repositorio.save (empleado);

}
public void borrar(Long id) {
repositorio.deleteById (id);
}
}
@manytoOne

```java
@Entity
public class Empleado {
@Id
@GeneratedValue
private Long id;
@ManyToOne
private Departamento departamento;   //Departamento debe ser también  @Entity
//resto de atributos, constructores, getters, setter, y resto de métodos.
//Si tenemos constructores no Lombok deben incluir ahora el departamento
```

@manyToOne

```java
@Entity
public class Categoria {
    @Id
    @GeneratedValue
    private Long id;
    @NotEmpty
    private String nombre;
    @OneToMany(fetch = FetchType.EAGER, cascade = CascadeType.REMOVE)
    private List<Empleado> empleados = new ArrayList<>();
}
```

bidireccional

```java
@Entity
public class Categoria {
    @Id
    @GeneratedValue
    private Long id;

    @NotEmpty
    private String nombre;

    @OneToMany(fetch = FetchType.EAGER, cascade = CascadeType.REMOVE,
               mappedBy="categoria")                    // orphanRemoval = true
     private List<Empleado> empleados = new ArrayList<>();
}
```

```java
@Entity
public class Empleado {
@Id
@GeneratedValue
private Long id;

@ToString.Exclude
@ManyToOne
@JoinColumn (name="CATEGORIA_ID")    //opcional
private Categoria categoria;
```

Paginacion

```
@Service
public class EmpleadoService {

    @Autowired
    EmpleadoRepository empleadoRepository;

    private final Integer pageSize = 10;

    public List<Empleado> getEmpleadosPaginados(Integer pageNum)  {
        Pageable paging = PageRequest.of(pageNum, pageSize, Sort.by("nombre"));
        Page<Empleado> pagedResult = empleadoRepository.findAll(paging);
        if (pagedResult.hasContent())  return pagedResult.getContent();
        else return null;
    }
}
```

```
@Controller
public class EmpleadoController {
    @Autowired
    public EmpleadoService empleadoService;

    @GetMapping("/")
    public String showList(@RequestParam(required = false) Integer pag, Model model) {
        int ultPag = empleadoService.getTotalPaginas() - 1;
        if (pag == null || pag < 0 || pag > ultPag) pag = 0;
        Integer pagSig = ultPag > pag ? pag + 1 : ultPag;
        Integer pagAnt = pag > 0 ? pag - 1 : 0;
        model.addAttribute("listaEmpleados", empleadoService.getEmpleadosPaginados(pag));
        model.addAttribute("pagSiguiente", pagSig);
        model.addAttribute("pagAnterior", pagAnt);
        model.addAttribute("pagFinal", ultPag);
        return "listView";
    }
}
```

```
<body>
  <h1>Listado de empleados</h1>
  <table>
     <thead><th>ID</th><th>Nombre</th><th>Email</th><th>Salario</th></thead>
     <tbody><tr th:each="empleado : ${listaEmpleados}">
                <td th:text="${empleado.id}">Id</td>
                <td th:text="${empleado.nombre}">nombre</td>
                <td th:text="${empleado.email}">email@gmail.com</td>
                <td th:text="${empleado.salario}">0</td>
            </tr>
     </tbody>
  </table>
  <a th:href="@{/?pag=0}">Pág inicial</a> | 
  <a th:href="@{/(pag=${pagAnterior})}">Pág ant</a> | 
  <a th:href="@{/(pag=${pagSiguiente})}">Pág sig</a> | 
  <a th:href="@{/(pag=${pagFinal})}">Utl.pág</a>
</body>
```