DGTIC UNAM
DIRECCIÓN GENERAL DE CÓMPUTO Y DE TECNOLOGÍAS DE INFORMACIÓN Y COMUNICACIÓN

REDEC UNAM

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
DIRECCIÓN GENERAL DE CÓMPUTO Y DE TECNOLOGÍAS
DE INFORMACIÓN Y COMUNICACIÓN

DIPLOMADO

Desarrollo de sistemas con tecnología Java

Módulo 10

API RESTful con Spring Boot

M. en C. Jesús Hernández Cabrera

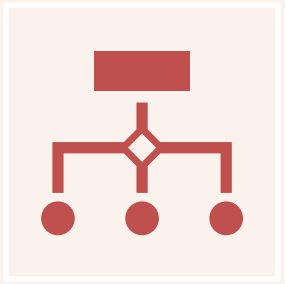
Capa de Servicio

Normalmente, el controlador es el último de la cadena de dependencias.

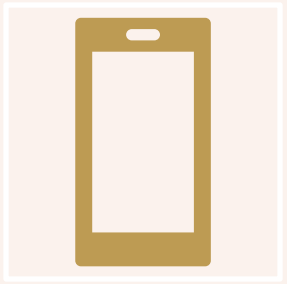
Recibe solicitudes HTTP del controlador frontal Spring (DispatcherServlet) y simplemente las delega a una **capa de servicio**.

El propósito de tener una capa de servicio (***Service***) en una aplicación REST, y en la arquitectura de aplicaciones en general, es separar la lógica de negocio de la lógica de presentación y de acceso a datos.

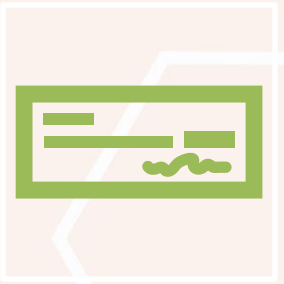
Capa de Servicio



Separación entre la lógica de negocio, la presentación y el acceso a datos.



Parte central de la arquitectura de una aplicación.

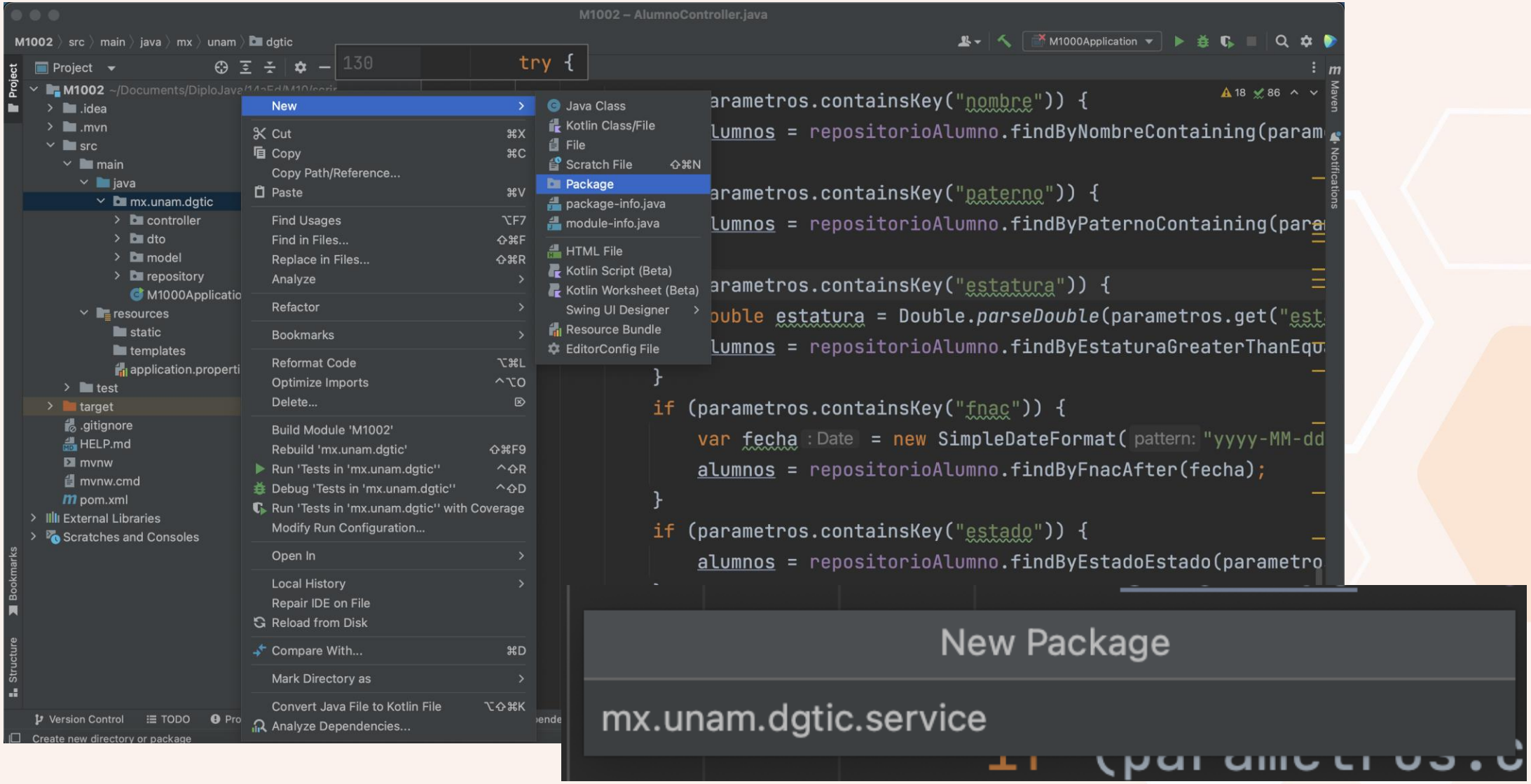


Encapsula las operaciones de negocio y reglas específicas.

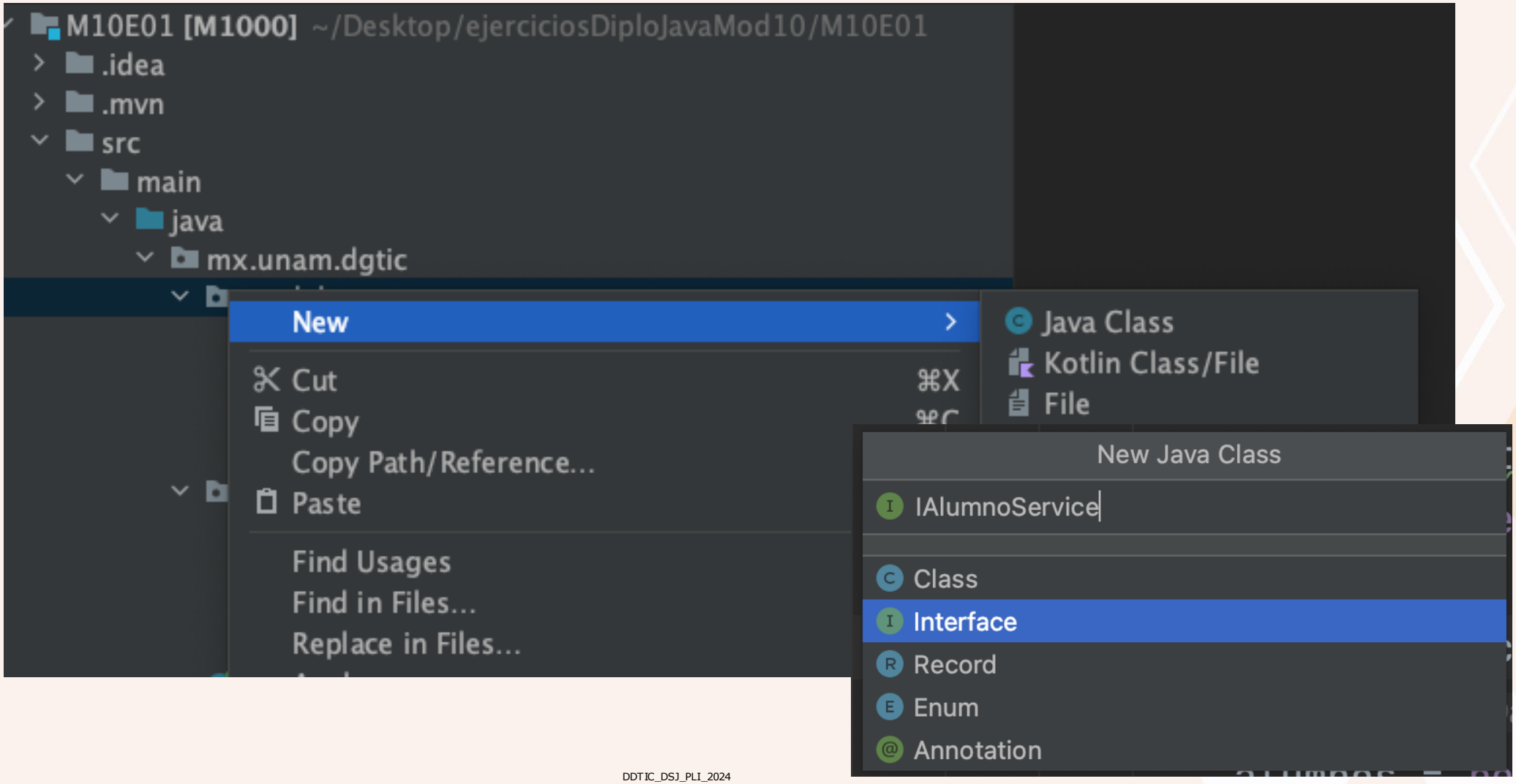
Capa de Servicio

Abstracción de la lógica de negocio.	• Centraliza la lógica de negocio, separándola del acceso a datos y la presentación. • Permite cambios en la lógica de negocio sin afectar otras capas.
Facilita la reutilización de código.	• Reduce la duplicación de lógica de negocio. • Permite compartir funcionalidades comunes entre diferentes partes de la aplicación.
Simplifica las pruebas unitarias y de integración.	• Permite pruebas aisladas de la lógica de negocio.
Promueve el desacoplamiento y la modularidad.	• Separa las preocupaciones de negocio, datos y presentación. • Facilita la mantenibilidad y escalabilidad de la aplicación.

Service


M1002 – AlumnoController.java
M1002 src main java mx unam dgtic
M1000Application
Project
M1002 ~/Documents/DiploJava/14aEd/M10/scri
.idea
.mvn
src
main
java
mx.unam.dgtic
controller
dto
model
repository
M1000Applicatio
resources
static
templates
application.properti
test
target
.gitignore
HELP.md
mvnw
mvnw.cmd
pom.xml
External Libraries
Scratches and Consoles
New
Cut ⌘X
Copy ⌘C
Copy Path/Reference...
Paste ⌘V
Find Usages ⌥F7
Find in Files... ⇧⌘F
Replace in Files... ⇧⌘R
Analyze
Refactor
Bookmarks
Reformat Code ⌥⌘L
Optimize Imports ^⌥O
Delete...
Build Module 'M1002'
Rebuild 'mx.unam.dgtic' ⇧⌘F9
Run 'Tests in 'mx.unam.dgtic'' ^⇧R
Debug 'Tests in 'mx.unam.dgtic'' ^⇧D
Run 'Tests in 'mx.unam.dgtic'' with Coverage
Modify Run Configuration...
Open In
Local History
Repair IDE on File
Reload from Disk
Compare With... ⌘D
Mark Directory as
Convert Java File to Kotlin File ⌥⇧⌘K
Analyze Dependencies...
Java Class
Kotlin Class/File
File
Scratch File ⇧⌘N
Package
package-info.java
module-info.java
HTML File
Kotlin Script (Beta)
Kotlin Worksheet (Beta)
Swing UI Designer
Resource Bundle
EditorConfig File
130
try {
arametros.containsKey("nombre")) {
lumnos = repositorioAlumno.findByNombreContaining(param
arametros.containsKey("paterno")) {
lumnos = repositorioAlumno.findByPaternoContaining(para
arametros.containsKey("estatura")) {
ouble estatura = Double.parseDouble(parametros.get("est
lumnos = repositorioAlumno.findByEstaturaGreaterThanEqu
}
if (parametros.containsKey("fnac")) {
var fecha : Date = new SimpleDateFormat(pattern: "yyyy-MM-dd
alumnos = repositorioAlumno.findByFnacAfter(fecha);
}
if (parametros.containsKey("estado")) {
alumnos = repositorioAlumno.findByEstadoEstado(parametro
18 86
Maven
Notifications
Bookmarks
Structure
Version Control
TODO
Create new directory or package
New Package
mx.unam.dgtic.service

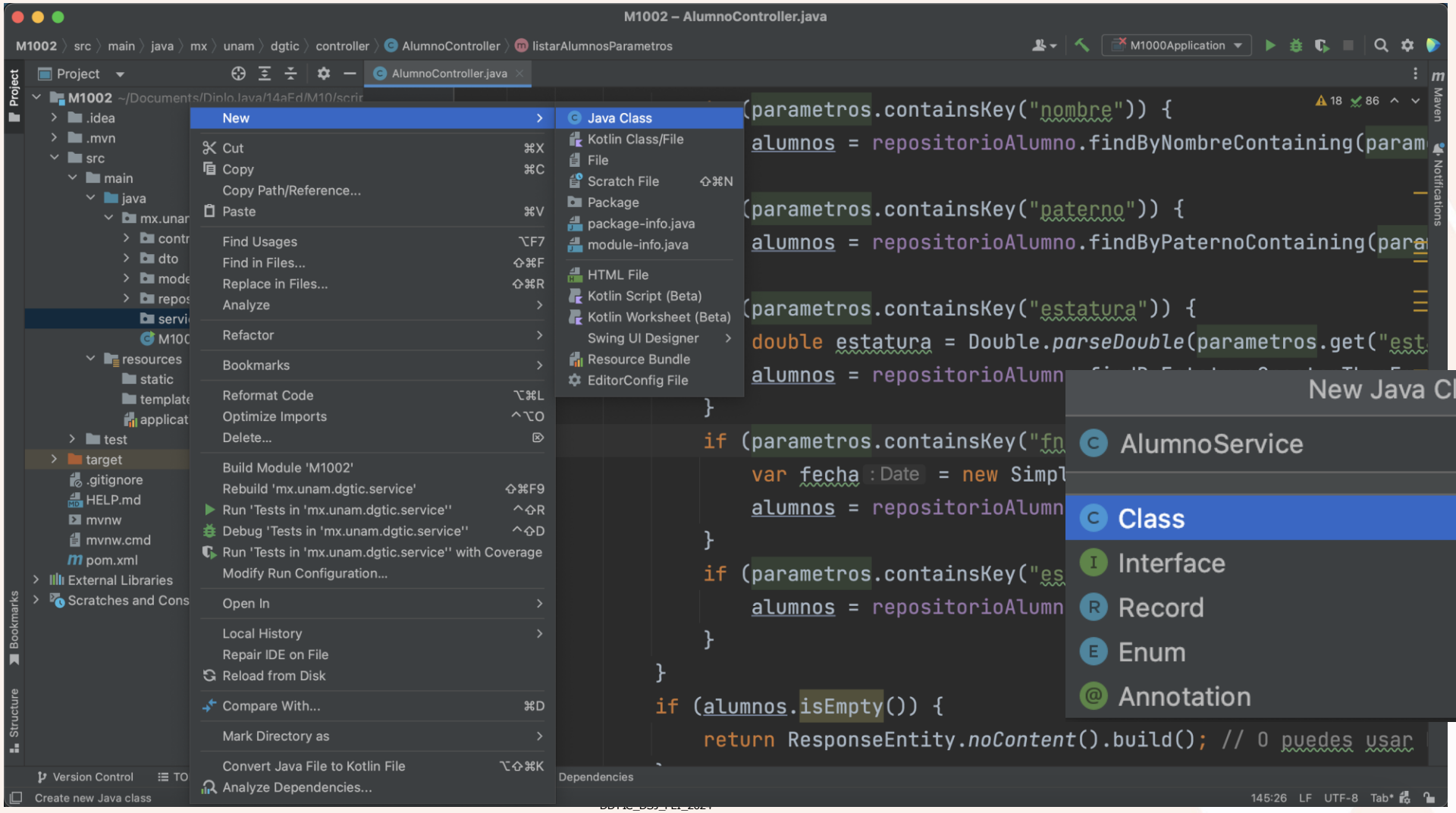
Service


M10E01 [M1000] ~/Desktop/ejerciciosDiploJavaMod10/M10E01
.idea
.mvn
src
main
java
mx.unam.dgtic
New
Cut ⌘X
Copy
Copy Path/Reference...
Paste
Find Usages
Find in Files...
Replace in Files...
Java Class
Kotlin Class/File
File
New Java Class
IAlumnoService
Class
Interface
Record
Enum
Annotation

Métodos abstractos

```java
public interface IAlumnoService {

    List<Alumno> getAlumnosList();
    Alumno updateAlumno(Alumno alumno);
    Alumno createAlumno(Alumno alumno);
    void deleteAlumno(String matricula);
    Optional<Alumno> getAlumnoById(String matricula);
    List<Alumno> findAlumnosByEstado(String estado);
}
```

Service


M1002 – AlumnoController.java
New
Java Class
Kotlin Class/File
File
Scratch File
Package
package-info.java
module-info.java
HTML File
Kotlin Script (Beta)
Kotlin Worksheet (Beta)
Swing UI Designer
Resource Bundle
EditorConfig File
Cut
Copy
Copy Path/Reference...
Paste
Find Usages
Find in Files...
Replace in Files...
Analyze
Refactor
Bookmarks
Reformat Code
Optimize Imports
Delete...
Build Module 'M1002'
Rebuild 'mx.unam.dgtic.service'
Run 'Tests in 'mx.unam.dgtic.service''
Debug 'Tests in 'mx.unam.dgtic.service''
Run 'Tests in 'mx.unam.dgtic.service'' with Coverage
Modify Run Configuration...
Open In
Local History
Repair IDE on File
Reload from Disk
Compare With...
Mark Directory as
Convert Java File to Kotlin File
Analyze Dependencies...
New Java Class
AlumnoService
Class
Interface
Record
Enum
Annotation

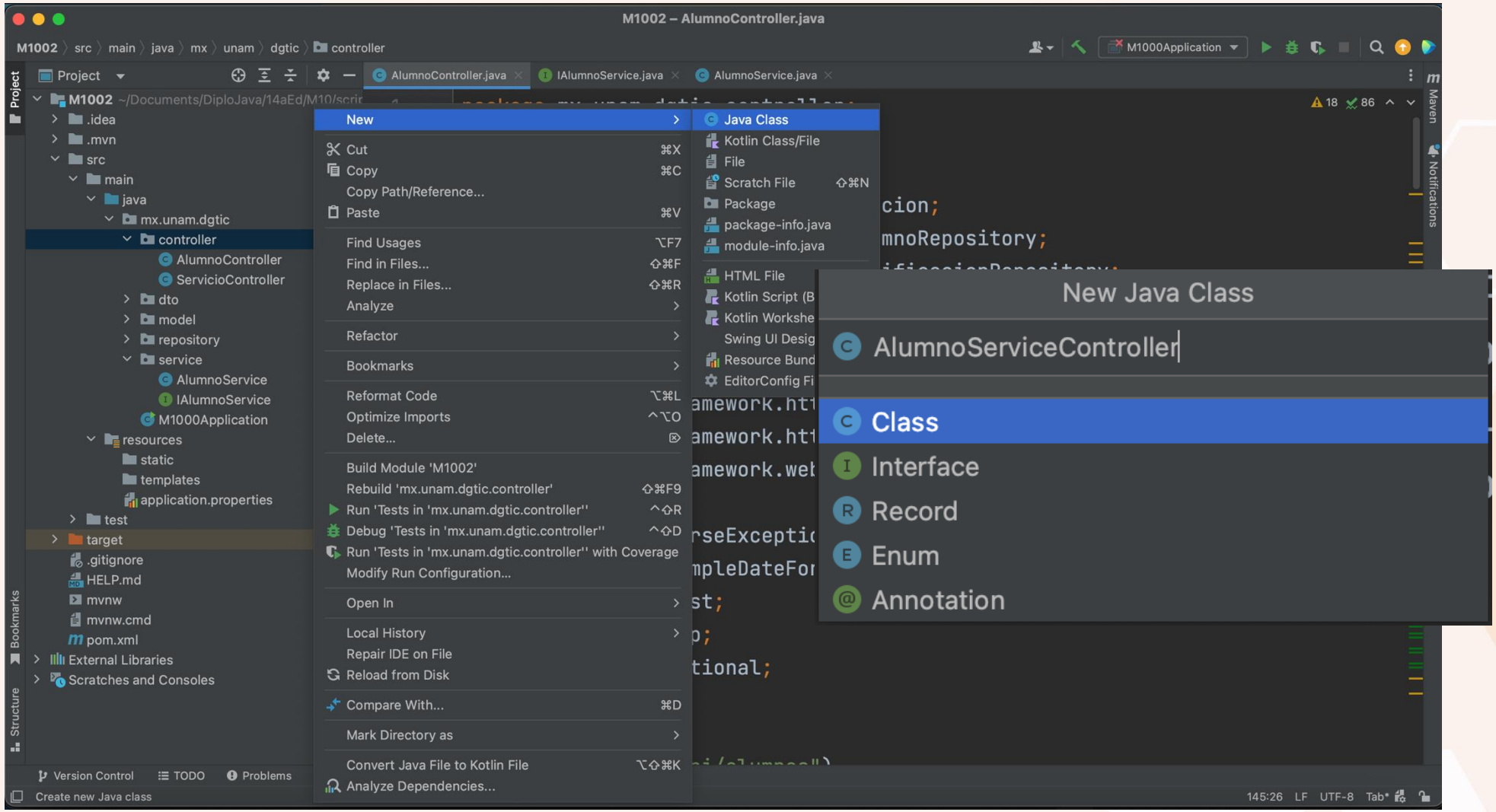
AlumnoService

```java
import java.util.Optional;

@Service
public class AlumnoService implements IAlumnoService{

    @Autowired
    private AlumnoRepository alumnoRepository;


    @Override
    public List<Alumno> getAlumnosList() {
        return alumnoRepository.findAll();
    }
```

Crear el controlador

- Crear el paquete controller.
- Crear el controlador RESTful.

Services en Controller



AlumnoServiceController

```java
@RestController
@RequestMapping(path = "/api/alumnos",
                produces = MediaType.APPLICATION_JSON_VALUE)
public class AlumnoServiceController {
    @Autowired
    AlumnoService alumnoService;

    @GetMapping(path = "/")
    public ResponseEntity<List<Alumno>> getAlumnos( ){
        return ResponseEntity.ok(
                alumnoService.getAlumnosList() List<Alumno>
                        .stream() Stream<Alumno>
                        .collect(Collectors.toList())
        );
    }

}
```

ModelMapper

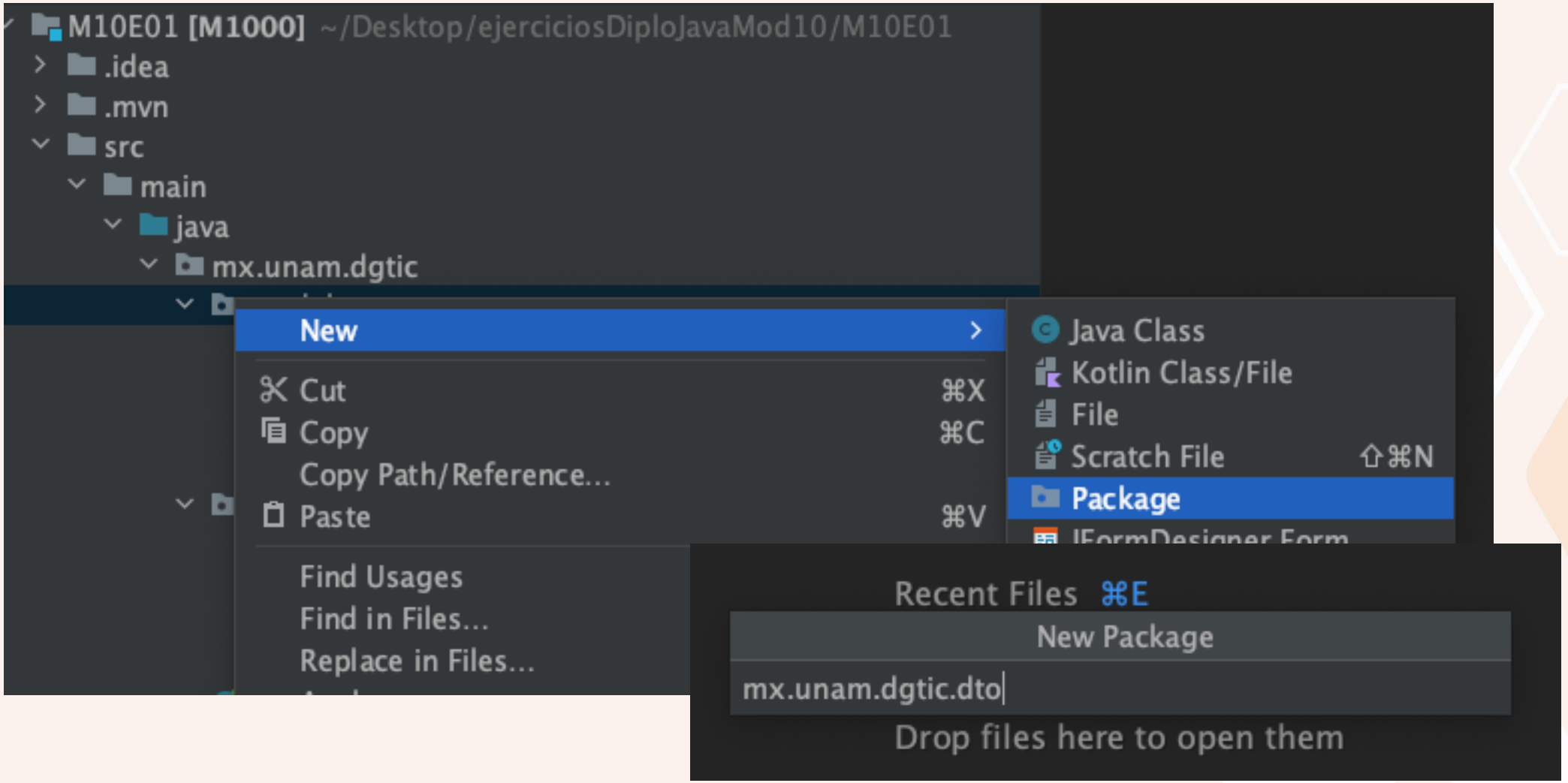
Se usa para simplificar el proceso de convertir un objeto de un tipo a otro.

Esto es particularmente útil cuando se separan los modelos de dominio (entidades) de los modelos de vista o DTOs, permitiendo una clara separación de la lógica de negocio de la presentación o transferencia de datos.

ModelMapper

- Ventajas
  - Reducción de Código Repetitivo (Boilerplate)
    - Minimiza la necesidad de escribir manualmente métodos de mapeo para cada transformación de objeto.
  - Flexibilidad
    - ModelMapper ofrece configuraciones avanzadas para manejar casos complejos de mapeo.
  - Mejora la Mantenibilidad
    - Al centralizar el mapeo de objetos, cualquier cambio en los objetos de origen o destino requiere ajustes en un solo lugar.

Crear AlumnoDto



Crear AlumnoDto

```
public class AlumnoDto {
    private String matricula;
    private String nombre;
    private String paterno;
    private String fnac;
    private double estatura;
    private String estado;

    public AlumnoDto() {
```

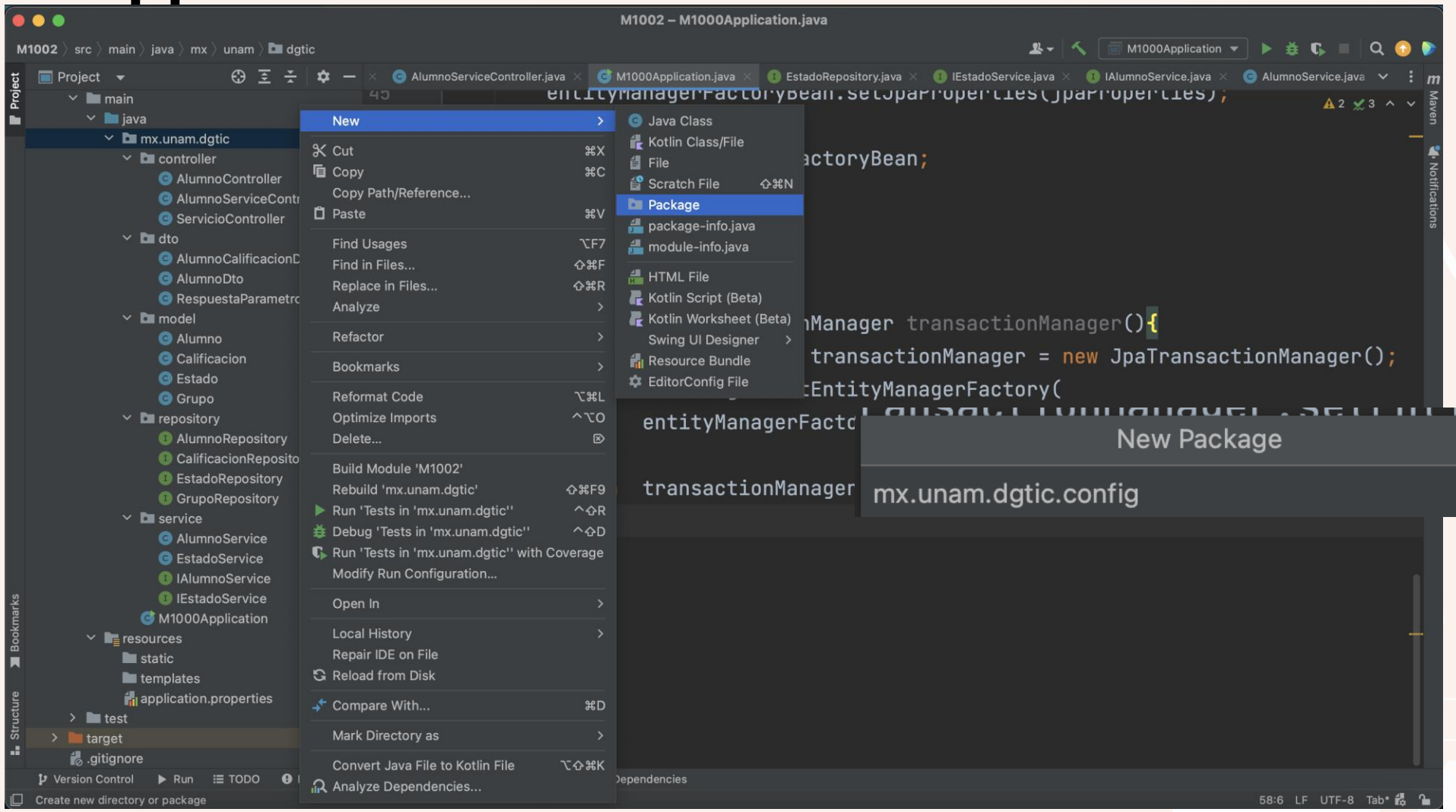
ModelMapper

- Agregar la dependencia ModelMapper a pom.xml

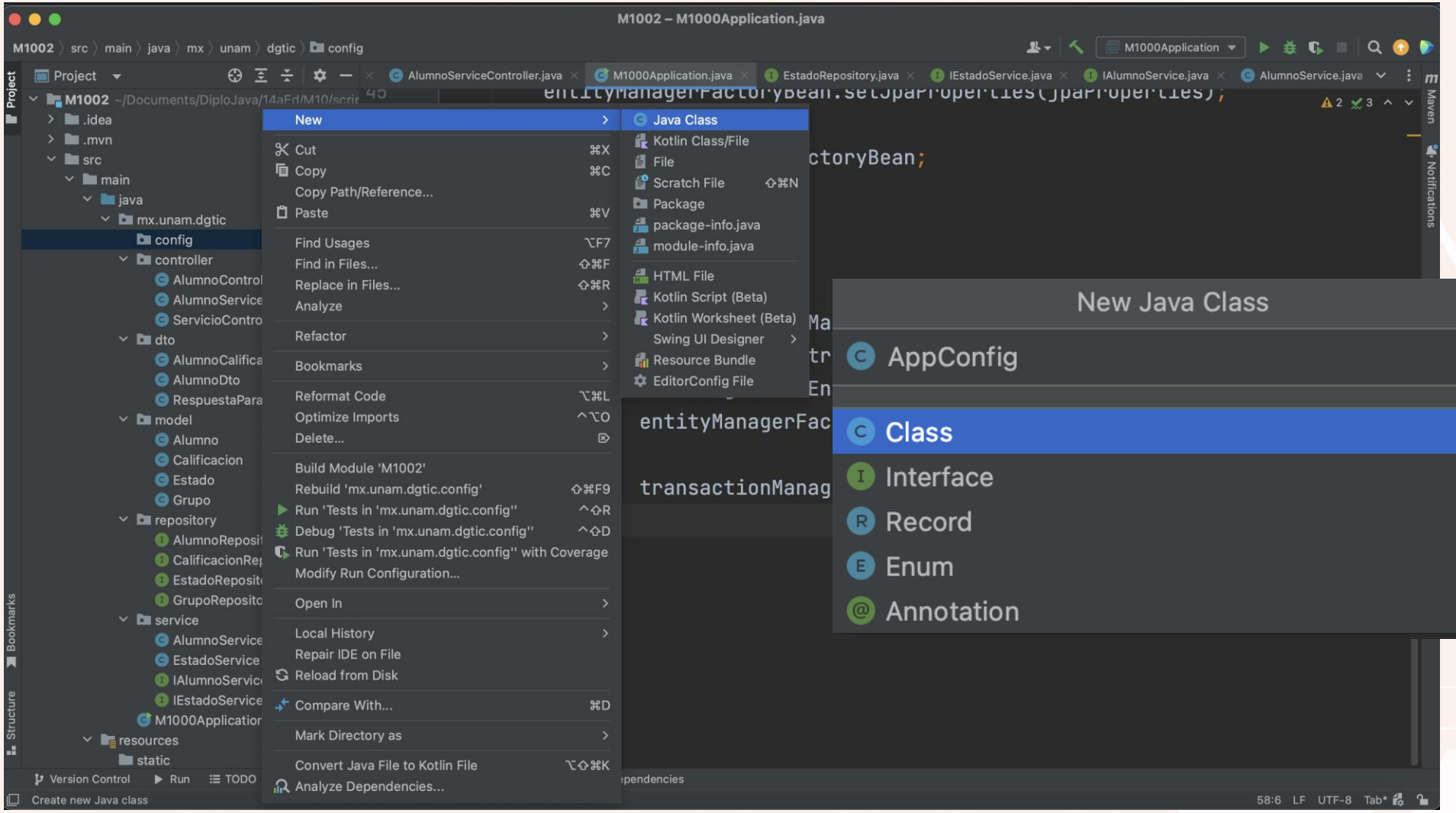
```
<dependency>
   <groupId>org.modelmapper</groupId>
   <artifactId>modelmapper</artifactId>
   <version>3.2.0</version>
</dependency>
```

- Configurar ModelMapper como un Bean de Spring
  - Puede usarse una clase de configuración @***Configuration***
  - Puede agregarse como un @***Bean*** en la clase principal anotada con @***SpringBootApplication***

ModelMapper



ModelMapper



```java
public interface IAlumnoService {

    List<AlumnoDto> getAlumnosList();

    AlumnoDto createAlumno(AlumnoDto alumno) throws ParseException;
    void deleteAlumno(String matricula);
    Optional<AlumnoDto> getAlumnoById(String matricula);
    List<AlumnoDto> findAlumnosByEstado(String estado);
}
```

Contacto

M. en C. Jesús Hernández Cabrera

Profesor de carrera

jesushc@unam.mx

Redes sociales:

www.linkedin.com/in/hcjesus