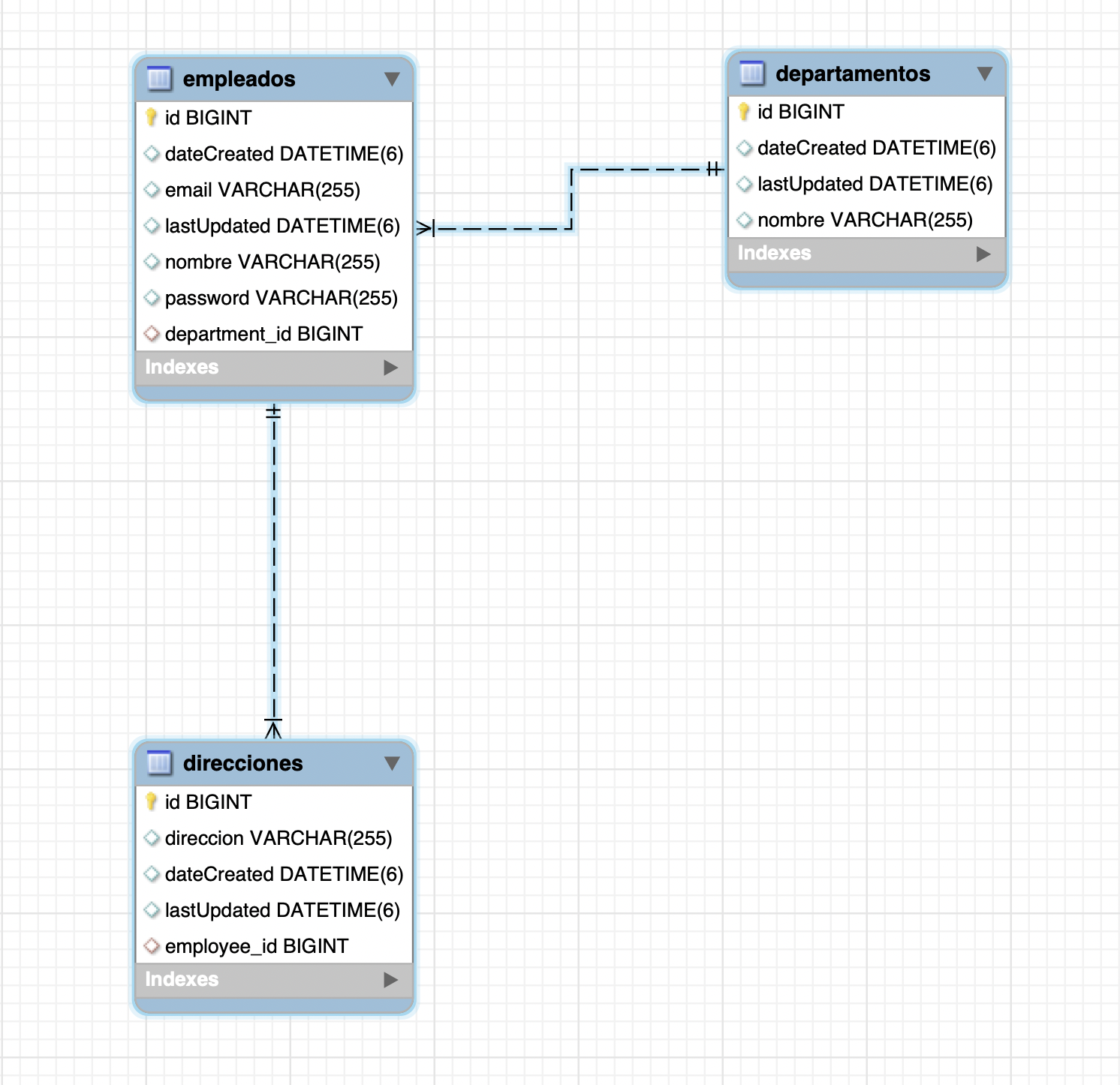
EJERCICIO MANY TO ONE

Vamos a realizar un ejercicio llamado EjercicioManyToOne, será un proyecto de Spring Boot y tendrá las siguientes dependencias:

* Spring Web.
* Spring Boot Dev Tools.
* Thymeleaf.
* MySQL Driver.
* Spring Data JPA.

Lo primero que vamos a hacer es crearnos una base de datos en MySql Workbench llamada **mi\_empresa**.



El funcionamiento de la aplicación será el siguiente: Al ejecutar la

aplicación, debemos ir a la URL: <http://localhost:8080/department/list>

donde se nos mostrara una tabla con los departamentos guardados en la base de datos, más adelante se describe lo que debe tener cada página web.

La aplicación debe tener lo siguiente:

* Paquete controller: Tendremos 2 controladores:
  + DepartmentController: Conteniendo los siguientes métodos:
    - list: Mapeado a la URL “/list”. Este método llamará al servicio departmentService y recuperará todos los departamentos de la base de datos y mostrará la página web list.html.
    - show: Método GET que recuperará el departamento asociado al id que le pasemos, mostrándolo en la pagina show.html.
    - create: Método GET que mostrará la página web create.html.
    - save: Método POST, que nos permitirá guardar un departamento, redirigiendo después a la pagina show.html, pasando el id del departamento que acabamos de guardar.
    - update: Método PUT, que nos permitirá actualizar el departamento en la base de datos, redirigiendo después a la pagina show.html, pasando el id del departamento que acabamos de guardar.
    - delete: Método DELETE, que nos permitirá eliminar el departamento en la base de datos, redirigiendo después a la pagina “/department”.
  + EmployeeController: Conteniendo los siguientes métodos:
    - list: Método GET. Este método llamará al servicio employeeService y recuperará todos los empleados de la base de datos y mostrará la página web list.html.
    - list: Método GET, mapeado a la URL: "/list/{departmentId}", este método recuperará el departamento asociado al id que le pasemos, mostrándolo en la pagina list.html.
    - add: Método GET que nos mostrara la pagina add.html de empleado.
    - save: Método POST, que nos permitirá guardar un empleado, redirigiendo después a la pagina show.html, pasando el id del empleado que acabamos de guardar.
    - show: Método GET que recuperará el empleado asociado al id que le pasemos, mostrándolo en la pagina show.html.
    - edit: Método GET que mostrará la página web edit.html, pasando el id que recibamos por parámetro.
    - update: Método POST, que nos permitirá actualizar el empleado en la base de datos, redirigiendo después a la pagina show.html, pasando el id del departamento que acabamos de guardar.
    - delete: Método DELETE, que nos permitirá eliminar el empleado en la base de datos, redirigiendo después a la pagina “/employee/list”.
  + Paquete entities: Tendremos 3 clases:
    - Adress: Tendrá las siguientes propiedades:

@Id

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

**private** **long** id;

@Temporal(TemporalType.***TIMESTAMP***)

**protected** Date dateCreated = **new** Date();

@Temporal(TemporalType.***TIMESTAMP***)

**protected** Date lastUpdated = **new** Date();

@Column(name = "direccion")

**private** String address;

@OneToOne(fetch = FetchType.***LAZY***)

**private** Employee employee;

Constructores

Getter/Setter

toString

* + - Department: Tendrá las siguientes propiedades:

@Id

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

**private** **long** id;

@Temporal(TemporalType.***TIMESTAMP***)

**protected** Date dateCreated = **new** Date();

@Temporal(TemporalType.***TIMESTAMP***)

**protected** Date lastUpdated = **new** Date();

@Column(name = "nombre")

**private** String name;

@OneToMany(cascade = CascadeType.***ALL***, mappedBy = "department", fetch = FetchType.***LAZY***)

**private** List<Employee> employees; Constructores

Getter/Setter

toString

* + - Employee: Tendrá las siguientes propiedades:

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

protected long id;

@Temporal(TemporalType.TIMESTAMP)

protected Date dateCreated = new Date();

@Temporal(TemporalType.TIMESTAMP)

protected Date lastUpdated = new Date();

@Column(name = "email")

private String email;

@Column(name = "nombre")

private String name;

@Column(name = "password")

private String password;

@OneToOne(mappedBy = "employee", optional = false, cascade = CascadeType.ALL)

private Address address;

@ManyToOne(fetch = FetchType.EAGER)

private Department department;

Constructor

Getter/Setter

* Paquete repository: Tendremos dos interfaces:
  + DepartmentRepository: la cual extenderá de JpaRepository.
  + EmployeeRepository: la cual extenderá de JpaRepository y tendrá los siguientes métodos:
    - Employee findById(**long** id);
    - List<Employee> findByDepartmentId(Long departmentId);
* Paquete services: Contendrá lo siguiente:
  + Clase DepartmentService, con lo siguiente:

@Autowired

private DepartmentRepository departmentRepository;

public List<Department> list() {}

public Object delete(){}

public Department update(Department department) {}

public Department save(Department department) {}

public Department get(long id) {}

* Clase EmployeeService, con lo siguiente:

@Autowired

private EmployeeRepository employeeRepository;

@Autowired

private DepartmentRepository departmentRepository;

public Object list() {}

public Employee save(Employee employee) {}

public Employee get(long id) {}

public Employee update(Employee employee) {}

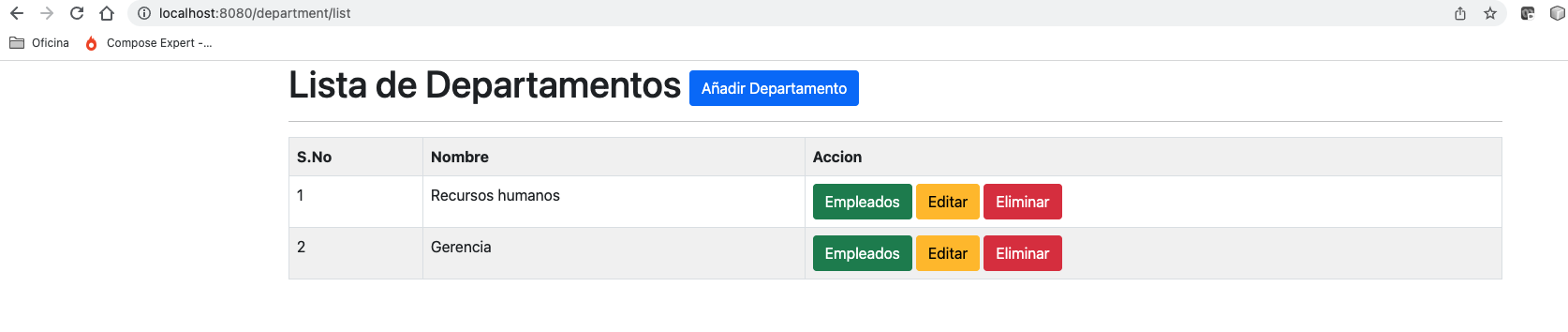
public void delete(long id) {}

public List<Employee> list(long departmentId) {}

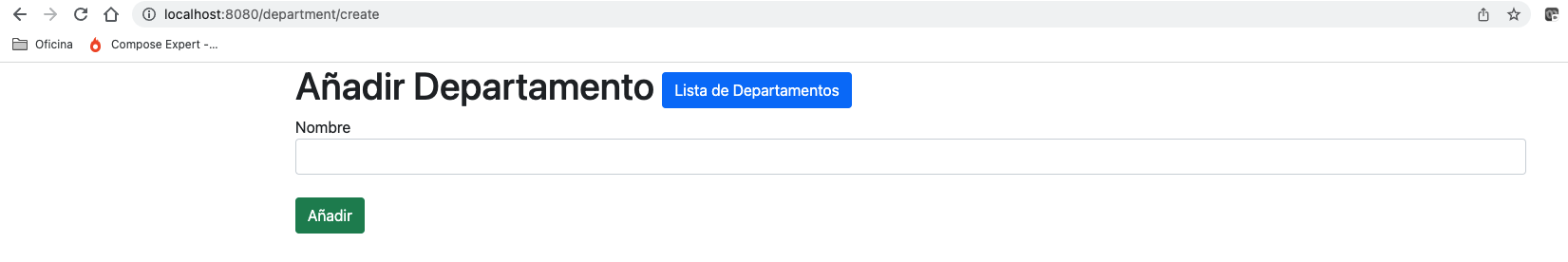
* Directorio templates: Contendrá las carpetas department y employee, con las siguientes páginas maquetadas con Bootstrap, estas paginas se entregarán ya maquetadas:
  + Carpeta department:
    - create.html
    - list.html
    - show.html
  + Carpeta employee:
    - add.html
    - edit.html
    - list.html
    - show.html

**Ejemplo de funcionamiento**

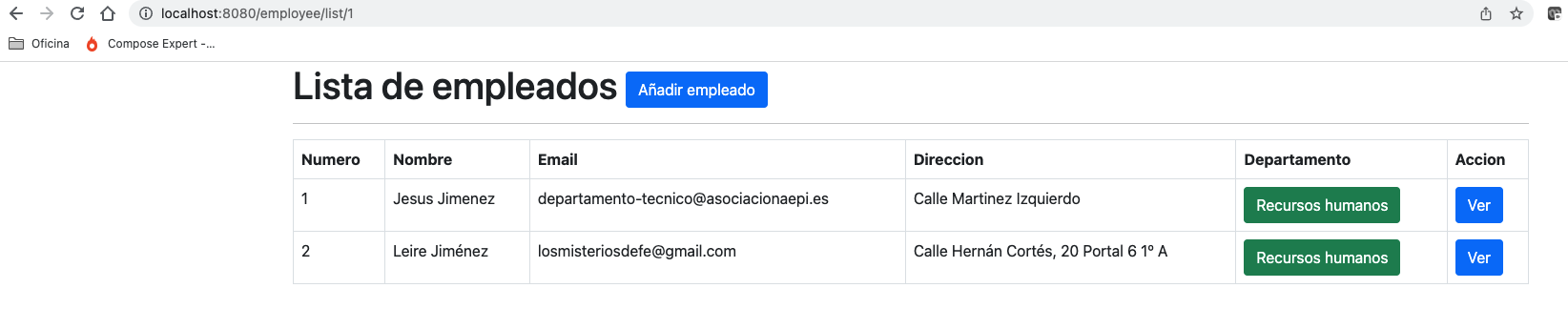
Al ejecutar la aplicación se nos muestra la siguiente pantalla



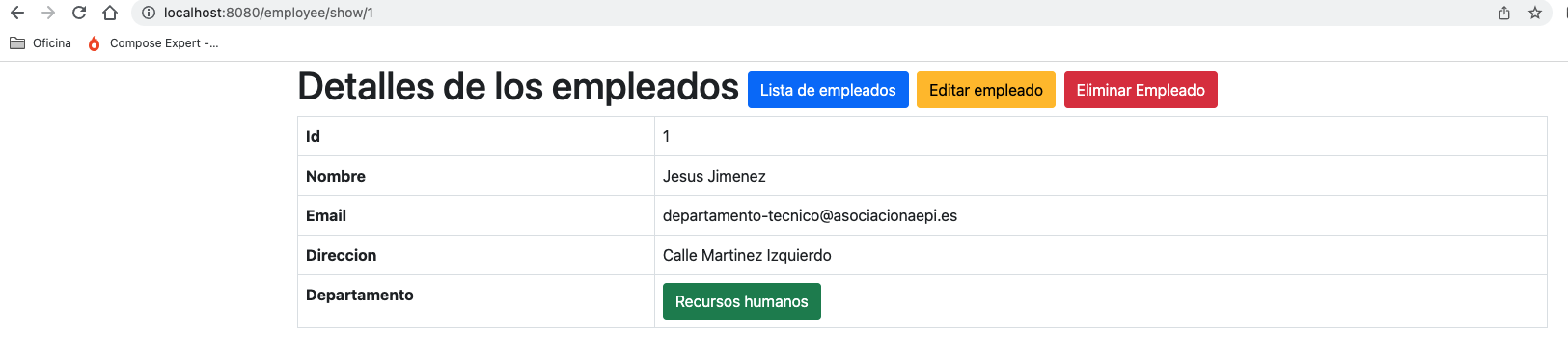
Al pulsar en Añadir Departamento, se nos muestra la siguiente pantalla



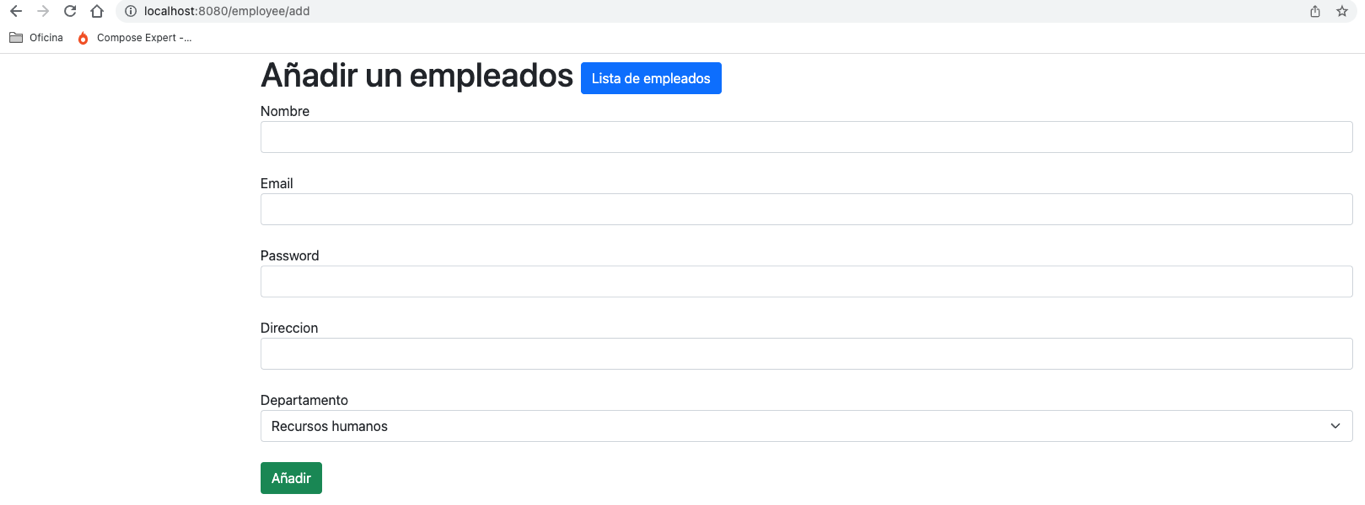
Al pulsar en el botón verde Empleados de la primera página, se nos muestra la siguiente pantalla



Al pulsar en el botón Ver de cada empleado, se nos muestra la siguiente pantalla



Al pulsar en Añadir empleado, de la pagina de lista empleados se nos muestra la siguiente pantalla



Al pulsar en el botón verde de la pagina Detalles de los empleados, se nos muestra la siguiente pantalla

