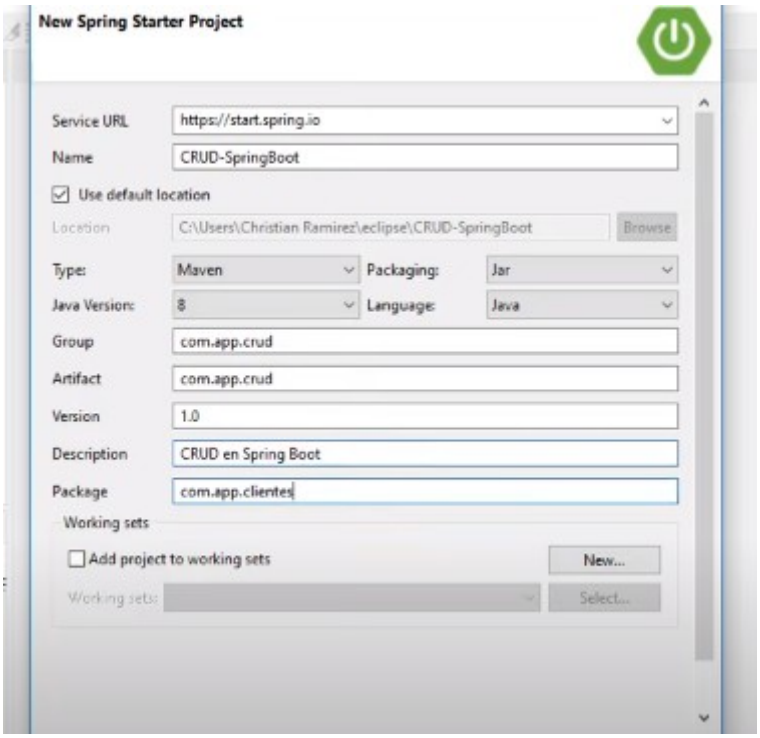
Anotaciones – Crud Spring Boot, Thymeleft y Bootstrap fecha: 23/08/23

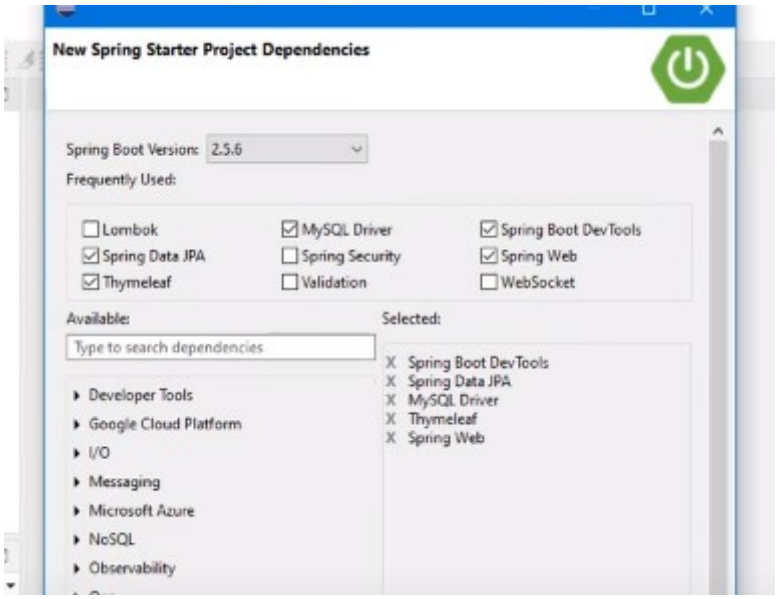
url guia: https://www.youtube.com/watch?v=oF3XmiHgT-I&ab_channel=LaTecnolog%C3%ADaAvanza

---

- Inicializar proyecto con eclipseIDE o STS, configuraciones:



- Añadir las librerias:



- Crear base de datos a utilizar

“bd_crud”

= -Configurar el properties -=

```
spring.datasource.url = jdbc:mysql://localhost:3306/bd_crud
spring.datasource.username = root
spring.datasource.password = 5246

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
spring.jpa.hibernate.ddl-auto = create

spring.jpa.show-sql = true
spring.jpa.properties.hibernate.format_sql = true

## server.port = 8090
```

= - Creacion de la entidad → objecto de la tabla en base de datos -=

- se importan varios atributos:

```
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.Table;
```

- se crearon los campos, id, nombre, apellido y email

```
@Entity
@Table(name = "estudiante")
public class Estudinate {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "nombre", nullable = false, length = 50)
    private String nombre;

    @Column(name = "apellido", nullable = false, length = 50)
    private String apellido;

    @Column(name = "email", nullable = false, length = 50, unique = true)
    private String email;
```

- se crean el cosntructor, seters y geters del mismo

= -Creacion de repositotio (DAO)-=

```java
package com.app.web.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import com.app.web.entidad.Estudinate;

@Repository
public interface EstudianteRepository extends JpaRepository<Estudinate, Long>{

}
```

-se extiende desde la clase “JpaRepository”, a esta se le pasa por parametros la clase a la cual se extenderan los metodos clasicos de un Crud
- esta clase servira para manejar la capa de interaccion entre la applicacion y la base de datos, cabe mencionar que los metodos extendidos ya estaran programados

= -Creacion de Service - =

```java
package com.app.web.service;

import java.util.List;

import com.app.web.entidad.Estudinate;

public interface EstudianteService {

    public List<Estudinate> listarEstudiantes();
}
```

- la interfaz nos servira para declarar los metodos disponibles que se haran uso a lo largo de la aplicación
- esta a su vez se implementa en su clase hija

= -Creacion de ServiceImpl-=

```java
package com.app.web.service;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.app.web.entidad.Estudinate;
import com.app.web.repository.EstudianteRepository;

@Service
public class EstudianteServiceImpl implements EstudianteService{

    @Autowired
    private EstudianteRepository repo;

    @Override
    public List<Estudinate> listarEstudiantes() {
        return repo.findAll();
    }

}

```

- servira como capa de funcionamiento, podra escoger entre los metodos del repositorio cual se usuara en cada funcion, aquí se podra manejar logica mas avanzada y hacer uso de algunas validaciones
- al usar JpaRepository se es mas facil hacer funciones basicas de crud ya programadas

= -Creacion de Controlador (endpoint)- =

```java
package com.app.web.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;

import com.app.web.service.EstudianteService;

@Controller
public class EstudianteController {

    @Autowired
    private EstudianteService service;

    @GetMapping("/estudiantesList")
    public String listarEstudiantes(Model modelo) {
        modelo.addAttribute("estudiantesList", service.listarEstudiantes());
        return "estudiantesList";
    }
}

```

- servira para definir los puntos de acceso del API
- se conctara con los servicios que previamente definamos en el proceso, es un enrutamiento
- en la linea 18, se puede hacer uso del objeto de respuesta del service.listarEstuantes() desde el archivo de respuesta HTML, estudianteList

= -Creacion del front con thymeleaf y boostrap- =

- ventana de lectutra o tabla

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
 <meta charset="UTF-8">
 <title>Estudiante</title>
 <!-- Bootstrap 4 css -->
 <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.2/dist/css/bootstr
<body>

    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
      <a class="navbar-brand" th:href="@{/}">Home</a>

      <div class="collapse navbar-collapse" id="navbarSupportedContent">
        <ul class="navbar-nav mr-auto">
          <li class="nav-item">
            <a class="nav-link" th:href="@{/estudiantes}">Estudiantes</a>
          </li>
        </ul>
      </div>
    </nav>

    <div class="container">
        <div class="row">
            <h1>Lista de estudiantes</h1>
        </div>
        <table class="table">
          <thead class="thead-dark table-striped table-bordered">
```

- en la etiqueda de html del inicio del docto, se anexa thymeleaft y su acronimo a usar, las etiquetas que empiecen con th seran mismas funciones de thymeleaft

```
    </thead>
    <tbody>
      <tr th:each="estudiante : ${estudiantesLista}">
        <th th:text="${estudiante.id}" scope="row"> Id </th>
        <th th:text="${estudiante.nombre}" scope="col"> Nombre </th>
        <th th:text="${estudiante.apellido}" scope="col"> Apellido</th>
        <th th:text="${estudiante.email}" scope="col"> Email </th>
      </tr>
    </tbody>
    <tfoot class="thead-dark">
```

-padazo de codigo para hacer bucle for con thymeleaft
- ventana de lista o lectura de datos en db

== ventana de creacion de estudiante, formulario

```
<div class="container">
    <div class="col-md-6 card container justify-content mt-3">
      <h1 class="text-center">Registro de estudiantes</h1>
      <div class="card-body">
        <form th:action="@{/estudianteSave}" th:object="${estudiante}" method="post">
          <div class="form-group">
            <label for="exampleInputPassword1">Nombre:</label>
            <input type="text" class="form-control" id="" placeholder="Ingrese nombre" th:field="*{nombre}">
          </div>
          <div class="form-group">
            <label for="exampleInputPassword1">Apellido:</label>
            <input type="text" class="form-control" id="" placeholder="Ingrese Apellido" th:field="*{apellido}">
          </div>
          <div class="form-group">
            <label for="exampleInputEmail1">Email:</label>
            <input type="email" class="form-control" id=""  placeholder="Ingrese email" th:field="*{email}">
          </div>
          <div class="form-group text-center">
             <button type="submit" class="btn btn-primary">Guardar</button>
          </div>
        </form>
      </div>
    </div>
```

- con la ayuda de thymeleaf se setea a los campos con los nombres de los atributos del objeto para ir a la db

== ventada de edicion, lgica en controlador y html

```
@GetMapping("/estudiantes/editar/{id}")
public String estudianteEditarForm(@PathVariable Long id, Model modelo) {
    modelo.addAttribute("estudiante", service.obtenerEstudianteById(id));
    return "editar_estudiante";
}
```

- se obtiene el objeto relacionado con el id pasado desde el siguiente boton

```
<tbody>
  <tr th:each="estudiante : ${estudiantesLista}">
    <th th:text="${estudiante.id}" scope="row"> Id </th>
    <th th:text="${estudiante.nombre}" scope="col"> Nombre </th>
    <th th:text="${estudiante.apellido}" scope="col"> Apellido</th>
    <th th:text="${estudiante.email}" scope="col"> Email </th>
      <th scope="col" >
          <a th:href="@{/estudiantes/editar/{id}(id=${estudiante.id})}" class="btn btn-info">Editar</a>
          <a th:href="@{/estudiantes/eliminar/{id}(id=${estudiante.id})}" class="btn btn-danger">Eliminar</a>
      </th>
  </tr>
</tbody>
```

- se anexan los botones en la plantilla, poniendo el id de cada registro en el boton correspondiente

```
<div class="container">
    <div class="col-md-6 card container justify-content mt-3">
      <h1 class="text-center">Edicion de estudiante</h1>
      <div class="card-body">
        <form th:action="@{/estudiantes/{id}(id=${estudiante.id})}" th:object="${estudiante}" method="post">

          <div class="form-group">
            <label for="exampleInputPassword1">Nombre:</label>
            <input type="text" class="form-control" id="" placeholder="Ingrese nombre" th:field="*{nombre}">
          </div>
          <div class="form-group">
            <label for="exampleInputPassword1">Apellido:</label>
            <input type="text" class="form-control" id="" placeholder="Ingrese Apellido" th:field="*{apellido}">
          </div>
          <div class="form-group">
            <label for="exampleInputEmail1">Email:</label>
            <input type="email" class="form-control" id=""  placeholder="Ingrese email" th:field="*{email}">
          </div>
          <div class="form-group text-center">
             <button type="submit" class="btn btn-primary">Actualizar</button>
          </div>
        </form>
      </div>
    </div>

</div>
```

- se setea la informacion obtenida desde el anterior punnto, mandando la info obtenida en los campos correspondiente, la accion del fromulario cambia ala nueva url de edicion

== funcionamiento de eliminacion

- el boton de eliminar se anexo en la tabla dentro del bucle, poniendo en el boton el id de cada registro en el boton correspondiente, se enlaza con el url de eliminacion dewl controller

```
    <th scope="col" >
        <a th:href="@{/estudiantes/editar/{id}(id=${estudiante.id})}" class="btn btn-info">Editar</a>
        <a th:href="@{/estudiantes/eliminar/{id}(id=${estudiante.id})}" class="btn btn-danger">Eliminar</a>
    </th>
</tr>
```

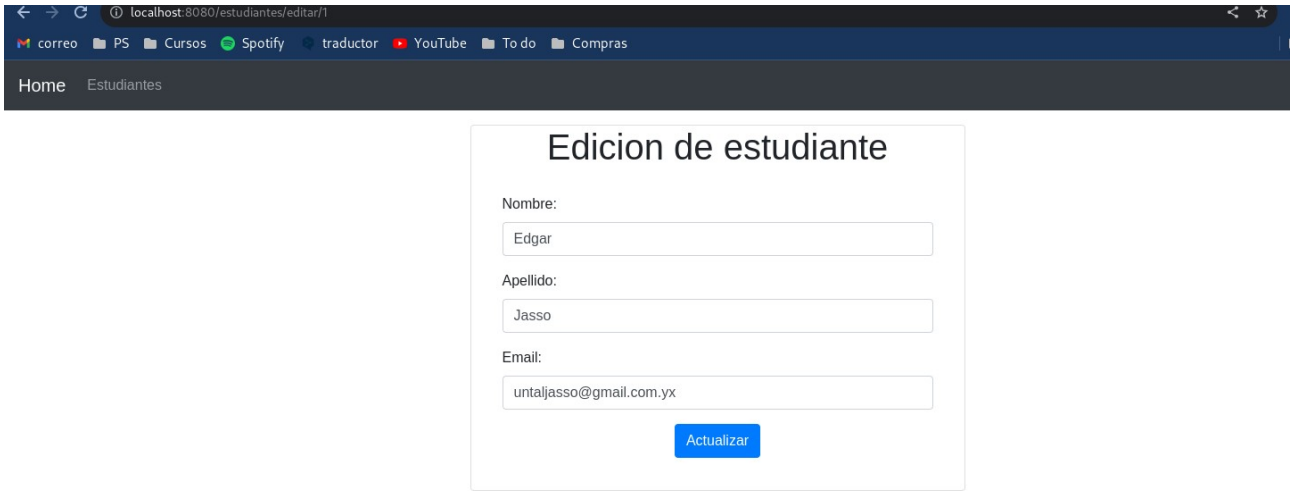
```
@GetMapping("/estudiantes/eliminar/{id}")
public String estudianteEliminar(@PathVariable Long id) {
    service.eliminarEstudiante(id);
    return "redirect:/estudiantes";
}
```

==========

tutorial terminado

==========

ventanas finales



==========

- mejoras disponibles hay muchas