10-11-2021

BACKEND

LABORATORIO

EVALUACIÓN 1 Proyecto “Movies”

María Navarro - Javier Sañudo - Héctor Oviedo

San valero –desarrollo de aplicaciones web 2º

# INDICE

Contenido

[INDICE 1](#_Toc87386021)

[PROYECTO MOVIES 2](#_Toc87386022)

[1 - Crea un nuevo proyecto web usando el framework Spring boot 2](#_Toc87386023) [2](#_Toc87386024)

[2 - Prepara el entorno según la metodología git-flow. – 2](#_Toc87386025)

[3 - Crea un controlador REST 2](#_Toc87386026)

[4 - Permite filtrar ese listado de películas por año y/o título 4](#_Toc87386027)

[5 - Añade el resto de las operaciones CRUD al controlador REST. 5](#_Toc87386028)

[6 - Crea el fichero de recursos de traducción (i18n) 5](#_Toc87386029)

[7 - Añade un método al controlador principal que devuelva en HTML 6](#_Toc87386030)

[8 - Crea una página web con un desplegable para seleccionar el idioma de la web. 7](#_Toc87386031)

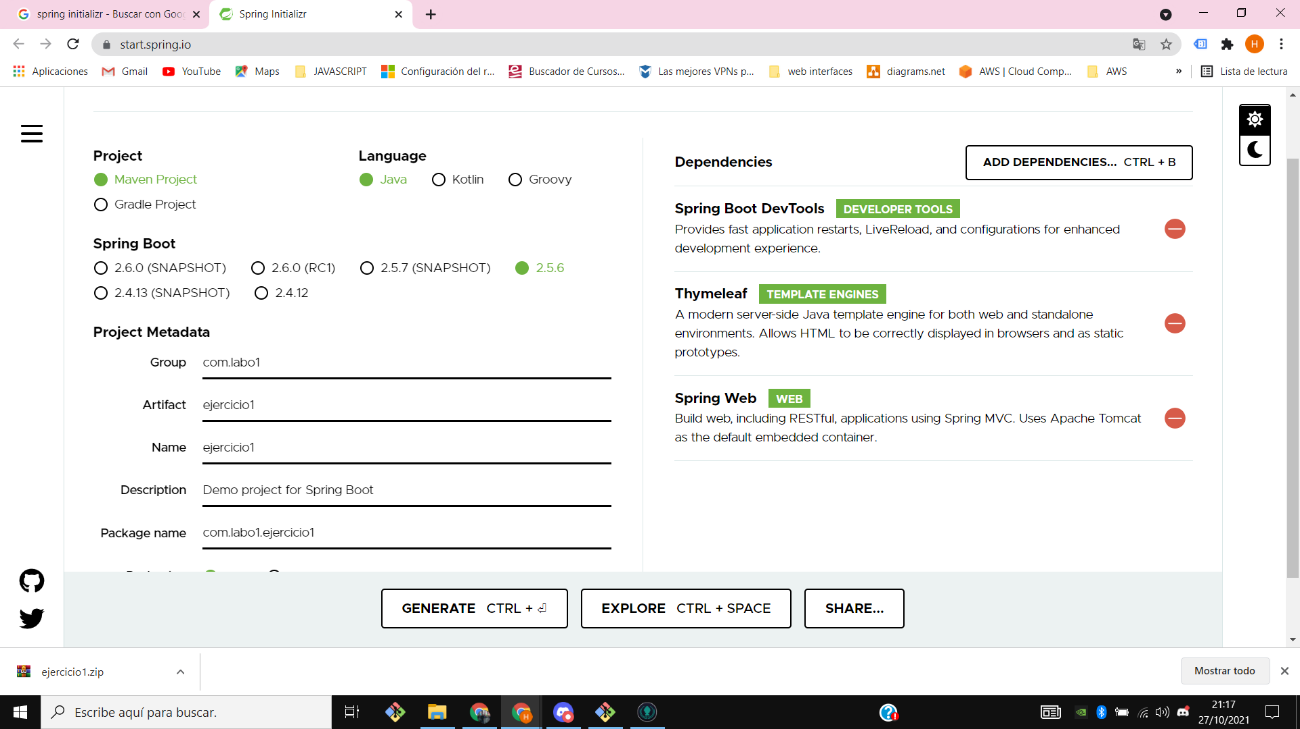
[9 - La API debe guardar el último idioma elegido 7](#_Toc87386032)

[10 - Defensa individual del proyecto en un video 8](#_Toc87386033)

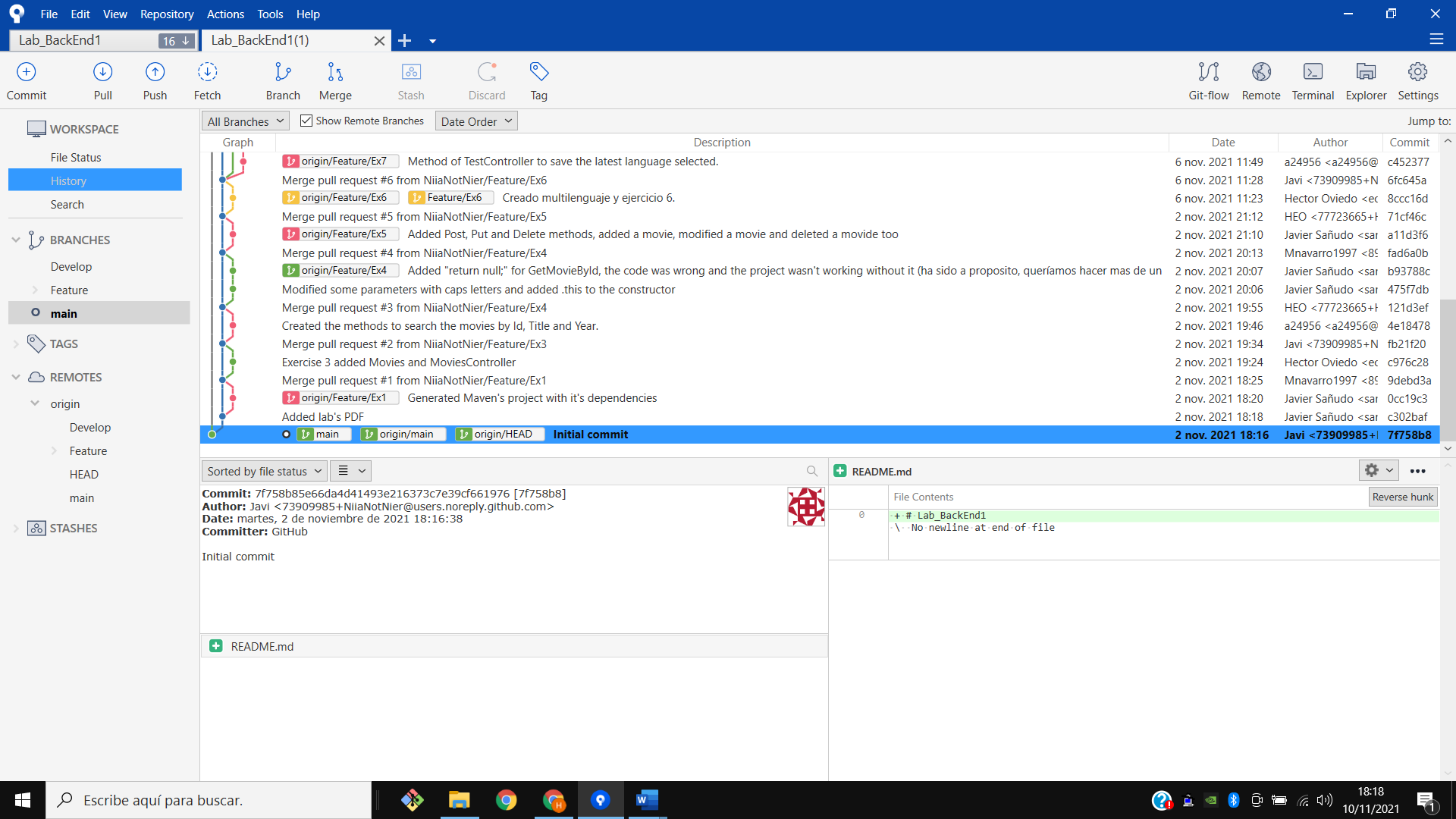
# PROYECTO MOVIES

1 - Crea un nuevo proyecto web usando el framework Spring boot. –

Añade las dependencias necesarias para el correcto funcionamiento del proyecto.



2 - Prepara el entorno según la metodología git-flow. –

Utiliza git a cada uno de los puntos siguientes de este trabajo. - Aplica las buenas prácticas de git.

## 3 - Crea un controlador REST que devuelva el listado de películas con al menos los campos Título y Año.

CLASES

Movies

public class Movies {

    public int id;

    public String title;

    public int year;

    public Movies(int id, String title, int year) {

        this.id = id;

        this.title = title;

        this.year = year;

    }

    public Movies() {

    }

    public String getTitle() {

        return title;

    }

    public void setTitle(String title) {

        this.title = title;

    }

    public int getYear() {

        return year;

    }

    public void setYear(int year) {

        this.year = year;

    }

    public int getId() {

        return id;

    }

    public void setId(int id) {

        this.id = id;

    }

}

MoviesController

@RestController

public class MoviesController {

    public static ArrayList<Movies> movies = new ArrayList(Arrays.asList(new Movies(1, "High School Musical", 2012),

            new Movies(2, "Burlesque", 1997), new Movies(3, "Superman", 1990), new Movies(4, "Solo en Casa", 1999),

            new Movies(5, "La chica de la curva", 2015), new Movies(6, "La huerfana", 2016),

            new Movies(7, "Los chicos del Maiz", 2009)));

    @GetMapping("Movies")

    public List<Movies> GetMovies() {

        return movies;

## 

## 4 - Permite filtrar ese listado de películas por año y/o título

CLASES

MoviesController

FindMovieByID – FindMovieByTile - FindMovieByYear

  private Movies FindMovieByID(int id) {

        for (Movies movie : movies) {

            if (movie.getId() == id) {

                return movie;

            }

        }

        return null;

    }

    private Movies FindMovieByTitle(String title) {

        for (Movies movie : movies) {

            if (movie.getTitle() == title) {

                return movie;

            }

        }

        return null;

    }

    private Movies FindMovieByYear(int year) {

        for (Movies movie : movies) {

            if (movie.getYear() == year) {

                return movie;

            }

        }

        return null;

    }

## 5 - Añade el resto de las operaciones CRUD al controlador REST.

CLASES

GET - POST – PUT - DELETE

  @PostMapping("/Movies")

    public Movies AddMovie(@RequestBody Movies newMovie) {

        movies.add(newMovie);

        return newMovie;

    }

    @PutMapping("/Movies/{id}")

    public Movies UpdateMovie(@PathVariable("id") int id, @RequestBody Movies updateMovie) {

        Movies m = FindMovieByID(id);

        m.setTitle(updateMovie.getTitle());

        return m;

    }

    @DeleteMapping("/Movies/{id}")

    public Movies DeleteMovie(@PathVariable("id") int id) {

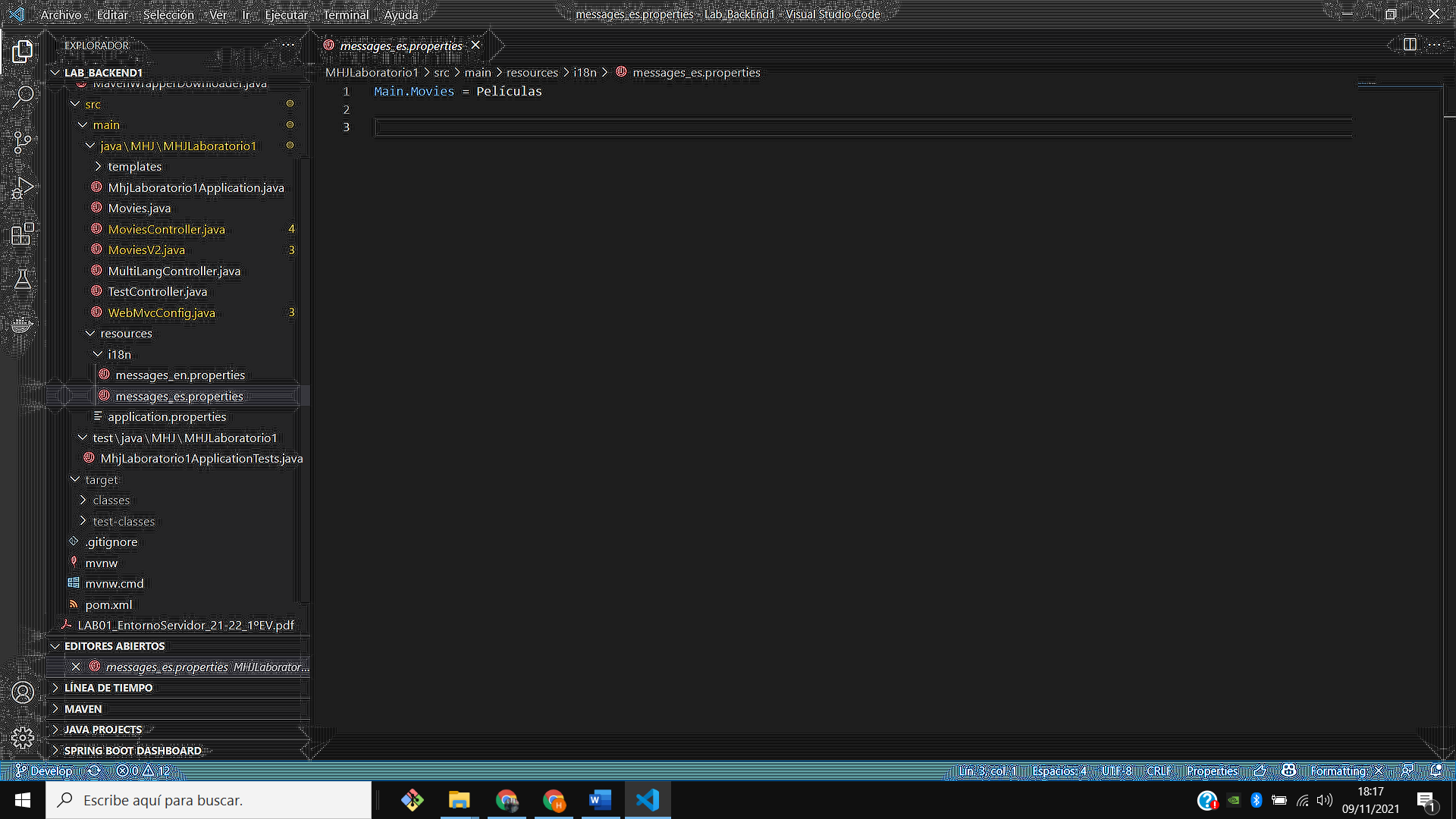
        Movies m = FindMovieByID(id);

        movies.remove(m);

        return null;

    }

## 6 - Crea el fichero de recursos de traducción (i18n) que permita a la API devolver la información en español o en inglés. -



## 

## 7 - Añade un método al controlador principal que devuelva en HTML al menos la palabra “Películas” o “Movies”, según el idioma elegido

@RestController

public class MultiLangController {

    @GetMapping("/test")

    public String Test(Locale locale) {

        var messages = ResourceBundle.getBundle("i18n\\messages", locale);

        return messages.getString("Main.Movies");

    }

}

@Configuration

@ComponentScan("com.example.API")

public class WebMvcConfig implements WebMvcConfigurer {

    @Override

    public void addInterceptors(InterceptorRegistry registry) {

        registry.addInterceptor(localeInterceptor());

    }

    @Bean

    public LocaleChangeInterceptor localeInterceptor() {

        LocaleChangeInterceptor localeInterceptor = new LocaleChangeInterceptor();

        localeInterceptor.setParamName("lang");

        return localeInterceptor;

    }

    @Bean

    public LocaleResolver localeResolver() {

        return new CookieLocaleResolver();

    }

    @Bean(name = "messageSource")

    public MessageSource getMessageSource() {

        ReloadableResourceBundleMessageSource mSource = new ReloadableResourceBundleMessageSource();

        mSource.setBasename("classpath:i18n/messages");

        mSource.setDefaultEncoding("UTF-8");

        return mSource;

    }

}

## 8 - Crea una página web con un desplegable para seleccionar el idioma de la web.

<html xmlns:th="http://www.thymeleaf.org">

<body>

    <h1 th:text="#{Main.Movies}"></h1>

    <span th:text = "#{lang.change}"></span>:

    <select id= "locales">

        <option value= ""></option>

        <option value="en" th:text= "#{lang.en}"></option>

        <option value="es" th:text= "#{lang.es}"></option>

    </select>

</body>

</html>

## 9 - La API debe guardar el último idioma elegido por el usuario para las siguientes veces que entre en la página web.

@Controller

public class TestController {

    @GetMapping("/testHtml")

    public ModelAndView Test() {

        return new ModelAndView("test");

    }

}

## 10 - Defensa individual del proyecto en un video de máximo 10 minutos donde se pueda ver funcionando el proyecto y cómo está hecho. –

## WEBGRAFIA

<https://docs.aws.amazon.com/es_es/apigateway/latest/developerguide/api-gateway-create-api-from-example.html>

<https://www.netmentor.es/entrada/api-rest-csharp>

<https://start.spring.io/>

<https://medium.com/@guillaume.viguierjust/building-a-multilingual-restful-api-2678add7febe>

<https://oracle-max.com/creando-un-api-rest-con-java-y-netbeans-que-devuelva-un-json-en-su-request/>

<https://www.apuntesdejava.com/2010/11/restful-la-forma-mas-ligera-de-hacer.html>

<http://expertojava.ua.es/experto/restringido/2014-15/rest/rest.html>

<https://ichi.pro/es/cree-una-api-de-peliculas-restful-usando-springboot-y-mongodb-8912081183002>

<https://devs4j.com/2018/01/04/crea-un-api-rest-de-la-forma-mas-simple-con-java-y-spring-data-rest/>