Nombre: Juan Ramón Ostos Rubio Grupo: G2-06

Informe individual de actividades del proyecto

Datos generales

URL del Repositorio de GitHub: https://github.com/gii-is-DP1/dp1-2020-g2-06
Nombre de usuario en GitHub: juanramonuvus/Juanra (no sé porque tengo dos…)
Rama del usuario en Github: juaostrub

Participación en el proyecto

Historias de usuario en las que he participado

-He implementado completas las historias de usuario HU-20, HU-22, HU-38
-He desarrollado junto a mi compañero Alejandro Barranco Ledesma las historias de usuario HU-3, HU-6, HU-34, HU-35, HU-36,
-He desarrollado junto a mi compañero Jesús Aparicio Ortiz la HU-6
-He desarrollado junto a mis compañeros David Brincau Cano y Víctor Granero Gil las historias de usuario HU-4, HU-5, HU-33
-Con todo el equipo las historias de usuario HU-1, HU-40
-He ayudado en un menor grado en las historias de usuario HU-17, HU-18, HU-37

Funcionalidad implementada

-He implementado el controlador TutorController, CompeticiónController, LogroController, he añadido varios métodos en el controlador de ArticuloController, también he participado a un menor nivel en el ProblemaController con el apartado de estadísticas y he implementado un método en el ApiRestController y realizado junto a Alejandro Barranco Ledesma y Jesús Aparicio Ortiz el NoticiaController
-He implementado los servicios de TutorService, CompeticiónService, LogroService y añadido varios métodos en el servicio de ArticuloService y realizado junto a Alejandro Barranco Ledesma el NoticiaService
-He creado la entidad Tutor, Competición, Administrador, Logro y añadida un atributo en la entidad Articulo y realizado junto a Alejandro Barranco Ledesma la entidad Noticia
-He implementado las vistas /tutor/createOrUpdateTutor, /tutor/tutorDetails, /tutoresList, competiciones/competicionDetails, competiciones/competicionesList, competiciones/createOrUpdateCompeticionForm. He implementado artículos/articulosDetails, /artículos/articulosList, /artículos/createOrUpdateArticuloForm junto David Brincau Cano y Victor Granero Gil he implementado, tambien las vista de /problemas/problemaDetails añadiendo las estadísticas a cada problema.
-He creado el validador “PassValidator”, la clase “PassConstraint” para tener una etiqueta personalizada
-He realizado toda la política de logs que hay implementada en el sistema
-He realizado las estadísticas de los problemas, con la inserción de gráficos, con la librería de libre uso llamada Morris.js
-También he retocado varias vistas como puede ser a la hora de ver el listado de problemas o la vista de la publicación, entre otras que no he podido que no puedo concretar.

-He realizado cambios en la vista del menú, concretamente el tag de menú, añadiendo nuestras direcciones
-He realizado junto a David Brincau Cano varias cargas de datos
-He modificado el método configure de la clase SecurityConfiguration, ya que daba inconsistencias
-He implementado paginación con Ayax en la vista de /tutores/tutoresList
-He implementado varias consultas sql en el EnvioRepository

[Además, si has implementado alguna clase de utilidad, validador específico en una clase a parte o un formatter para un tipo de dato concreto deberías indicarlo también.]

Esto hace un total de 13 clases implementadas por mí y 4 interfaces definidos.

Pruebas implementadas

Pruebas unitarias

He creado tests unitarios para 5 servicios (Noticia, Tutor, Administrador, Publicacion, Logro), 3 controladores (Tutor, Noticia, Pregunta Tutor), he creado 1 validador ("PassValidator"). Eso hace un total de 5 clases de test unitarios con un total de 33 métodos anotados con @Test.

Pruebas de Controlador

He creado 4 casos de prueba positivo y 2 negativos de controlador para la HU-1, para las historias de usuario HU-3, HU-6, HU-34, HU-35, HU-36, he creado 6 casos de prueba positivos y 5 casos de prueba negativos, para las historias de usuario HU-14 y HU-15 he creado 2 casos positivos y 3 casos negativos

Ejemplos de funcionalidades implementadas

Entidades (máximo de dos ejemplos)

Entidad Tutor

```
package org.springframework.samples.petclinic.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Table;
import javax.validation.constraints.NotEmpty;

import org.springframework.samples.constraint.EmailConstraint;
import org.springframework.samples.constraint.PassConstraint;
import com.fasterxml.jackson.annotation.JsonIgnore;

import lombok.Getter;
import lombok.Setter;

@Getter
@Setter
@Entity
@Table(name = "tutores")
public class Tutor extends BaseEntity{
```

```
	@Column(name = "nombre")
	@NotEmpty(message= "El campo 'Nombre' no puede estar vacío")
	private String nombre;

	@Column(name = "apellidos")
	@NotEmpty(message= "El campo 'Apellidos' no puede estar vacío")
	private String apellidos;

	private Boolean enabled;

	@EmailConstraint
	@Column(unique=true)
	@JsonIgnore
	private String email;

	@Column(name = "pass")
	@PassConstraint
	@NotEmpty
	@JsonIgnore
	private String pass;

	@Column(name = "imagen")
	private String imagen;
```

Aquí en la entidad de Tutor tal como se puede ver en el trozo de código de arriba, con los atributos: nombre, email, apellidos, pass, imagen, esta clase extiende de BaseEntity del cual hereda la clave primaria “id” un integer y un método que devuelve un booleano “isNew()”, Este modelo es uno de los tipos de usuario que hay en nuestro página y que desempeña acciones como: Crear/editar/borrar noticias/artículos, resolver dudas de alumnos y revisar envíos y problemas

Entidad Administrador

```
package org.springframework.samples.petclinic.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Table;
import javax.validation.constraints.Email;
import javax.validation.constraints.NotEmpty;

import org.springframework.samples.constraint.EmailConstraint;
import org.springframework.samples.constraint.PassConstraint;

import lombok.Getter;
import lombok.Setter;

@Getter
@Setter
@Entity
@Table(name="administradores")
public class Administrador extends BaseEntity{

	@EmailConstraint
```

```
	@Column(unique=true)
	private String email;

	@Column(name="pass")
	@PassConstraint
	private String pass;

	private Boolean enabled;
}
```

Aquí esta la entidad Administrador tal como se puede ver en el trozo de código de arriba, con los atributos: email, pass, esta hereda de BaseEntity del cual hereda la clave primaria "id" y método que devuelve un booleano "isNew()", Este modelo es uno de los tipos de usuario que hay en nuestro página y que desempeña acciones como: Crear/editar tutores y creadores, editar perfiles de alumno, y llevar la gestión de la página.

Servicio

Servicio Tutor

```
package org.springframework.samples.petclinic.service;

import java.util.Collection;
import java.util.Optional;

import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Slice;
import org.springframework.samples.petclinic.model.Noticia;
import org.springframework.samples.petclinic.model.Tutor;
import org.springframework.samples.petclinic.repository.TutorRepository;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.stereotype.Service;

@Service
public class TutorService {

	@Autowired
	TutorRepository tutoRepo;

	@Autowired
	NoticiaService noticiaService;

	@Autowired
	ArticuloService articuloService;


	public Collection<Tutor> findAll(){
		return tutoRepo.findAll();
	}

	public Optional<Tutor> findById(int id){
		return tutoRepo.findById(id);
	}

	public void save(@Valid Tutor tutor) {
```

```
		BCryptPasswordEncoder encoder = new BCryptPasswordEncoder();
		tutor.setPass(encoder.encode(tutor.getPass()));
		tutoRepo.save(tutor);
	}

	public Collection<Noticia> findTutorNoticias(int id){
		return noticiaService.findNoticiasByTutor(id);
	}

	public Optional<Tutor> findByEmail(String email) {
		return tutoRepo.findByEmail(email);
	}

	public Slice<Tutor> findTutorPage(Pageable pageable){
		return tutoRepo.findTutorPage(pageable);
	}


}
```

Este es el servicio del Tutor, que en el cabe de destacar los métodos de: findTutorPage, findTutorNoticias, las cuales devuelven un Slice de tutores para poder paginar los tutores en la página, y obtener los tutores participantes de una noticia, respectivamente.

Controlador

Controlador Tutor

```
package org.springframework.samples.petclinic.web;


import java.io.IOException;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.List;
import java.util.Optional;
import java.util.stream.Collectors;

import javax.servlet.http.HttpServletRequest;
import javax.validation.Valid;

import org.springframework.beans.BeanUtils;
import org.springframework.beans.BeansException;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Sort;
import org.springframework.samples.petclinic.model.Tutor;
import org.springframework.samples.petclinic.service.AdministradorService;
import org.springframework.samples.petclinic.service.AlumnoService;
import org.springframework.samples.petclinic.service.ArticuloService;
import org.springframework.samples.petclinic.service.AuthService;
import org.springframework.samples.petclinic.service.CreadorService;
import org.springframework.samples.petclinic.service.FileService;
import org.springframework.samples.petclinic.service.NoticiaService;
import org.springframework.samples.petclinic.service.PreguntaTutorService;
import org.springframework.samples.petclinic.service.TutorService;
import org.springframework.samples.petclinic.util.Utils;
```

```
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.multipart.MultipartFile;

import lombok.extern.slf4j.Slf4j;

@Slf4j
@Controller
@RequestMapping("/tutores")
public class TutorController {

	private final Path rootImage =
Paths.get("src/main/resources/static/resources/images/tutores");

	@Autowired
	AlumnoService alumnoService;

	@Autowired
	CreadorService creadorService;

	@Autowired
	TutorService tutorService;

	@Autowired
	AdministradorService administradorService;

	@Autowired
	ArticuloService articuloService;

	@Autowired
	NoticiaService noticiaService;

	@Autowired
	FileService fileService;

	@Autowired
	AuthService authService;

	@Autowired
	PreguntaTutorService preguntaTutorService;


	@GetMapping("")
	public String listTutores(ModelMap model) {
		model.addAttribute("tutores", tutorService.findAll());
		return "/tutores/tutoresList";
	}

	@GetMapping("/new")
	public String initCreationForm(ModelMap model) {
		Tutor tutor = new Tutor();
		model.put("tutor", tutor);
		return "tutores/createOrUpdateTutorForm";
```

```
	}

	@PostMapping("/new")
	public String processCreationForm(@Valid Tutor tutor,BindingResult
result,ModelMap model,@RequestParam("image") MultipartFile imagen) throws
IOException {
		boolean emailExistente =
Utils.CorreoExistente(tutor.getEmail(),alumnoService,tutorService,creadorServic
e,administradorService);
		if(emailExistente) {
			model.clear();
			model.addAttribute("tutor", tutor);
			model.addAttribute("message", "Ya existe una cuenta con ese
correo asociado");
			return "tutores/createOrUpdateTutorForm";
		}
		if(result.hasErrors()|| imagen.getBytes().length/(1024*1024)>10 ||
imagen.isEmpty()) {
			model.clear();
			model.addAttribute("tutor", tutor);
			List<String> errores = result.getAllErrors().stream().map(x-
>x.getDefaultMessage()).collect(Collectors.toList());
			if(imagen.isEmpty() ||
imagen.getBytes().length/(1024*1024)>10) {
				errores.add("La imagen debe tener un tamaño inferior
a 10MB");
			}
			model.addAttribute("message", errores);
			return "tutores/createOrUpdateTutorForm";
		}else {
			String extensionImagen[] =
imagen.getOriginalFilename().split("\\.");
			String name =
Utils.diferenciador(extensionImagen[extensionImagen.length-1]);
			tutor.setImagen("resources/images/tutores/"  + name);
			fileService.saveFile(imagen,rootImage,name);
			fileService.imageCrop("resources/images/tutores/"  + name,
fileService);
			tutor.setEnabled(true);
			tutorService.save(tutor);
			authService.saveAuthoritiesTutor(tutor.getEmail(), "tutor");
			model.addAttribute("message", "Tutor creado con éxito");
			return listTutores(model);
		}
	}


	@GetMapping("/{id}/edit")
	public String editTutor(@PathVariable("id") int id, ModelMap model,
HttpServletRequest request) {
		Optional<Tutor> tutor = tutorService.findById(id);
		if(tutor.isPresent()) {

	if(!tutorService.findById(id).get().getEmail().equals(SecurityContextHold
er.getContext().getAuthentication().getName()) &&
!Utils.authLoggedIn().equals("administrador")) {
				model.addAttribute("message","Solo puedes editar tu
propio perfil");
				log.warn("Un usuario esta intentando editar un
articulo sin tener los permisos necesarios, con sesion "+request.getSession());
```

```
                        return listTutores(model);
                }
                model.addAttribute("tutor", tutor.get());
                return"tutores/createOrUpdateTutorForm";
            }else {
                model.addAttribute("message","No se encuentra el tutor
seleccionado");
                return listTutores(model);
            }
      }

      @PostMapping("/{id}/edit")
      public String editTutor(@PathVariable("id") int id, @Valid Tutor
modifiedTutor, BindingResult binding, HttpServletRequest request,ModelMap
model,@RequestParam("image") MultipartFile imagen) throws BeansException,
IOException {
            Optional<Tutor> tutor = tutorService.findById(id);
            boolean emailExistente =
Utils.CorreoExistente(modifiedTutor.getEmail(),alumnoService,tutorService,cread
orService,administradorService);

      if(!tutorService.findById(id).get().getEmail().equals(SecurityContextHold
er.getContext().getAuthentication().getName()) &&
!Utils.authLoggedIn().equals("administrador")) {
                model.addAttribute("message","Solo puedes editar tu propio
perfil");
                log.warn("Un usuario esta intentando editar un articulo sin
tener los permisos necesarios, con sesion "+request.getSession());
                return listTutores(model);
            }
            if(emailExistente) {
                model.clear();
                model.addAttribute("tutor", modifiedTutor);
                model.addAttribute("message", "Ya existe una cuenta con ese
correo asociado");
                return "tutores/createOrUpdateTutorForm";
            }
            if(binding.hasErrors()|| imagen.getBytes().length/(1024*1024)>10)
{
                model.clear();
                model.addAttribute("tutor", tutor.get());
                List<String> errores =
binding.getAllErrors().stream().map(x-
>x.getDefaultMessage()).collect(Collectors.toList());
                if(imagen.isEmpty() ||
imagen.getBytes().length/(1024*1024)>10) {
                        errores.add("La imagen debe tener un tamaño inferior
a 10MB");
                }
                model.addAttribute("message", errores);
                return "tutores/createOrUpdateTutorForm";
            }else {
                if(!imagen.isEmpty()) {
                        String extensionImagen[] =
imagen.getOriginalFilename().split("\\.");
                        String aux = tutor.get().getImagen();
                        String name =
Utils.diferenciador(extensionImagen[extensionImagen.length-1]);
                        tutor.get().setImagen("resources/images/tutores/"  +
name);
```

```
	fileService.delete(Paths.get("src/main/resources/static/" + aux));
                        fileService.saveFile(imagen,rootImage,name);
                        fileService.imageCrop("resources/images/tutores/"  +
name, fileService);
                    }
                    BeanUtils.copyProperties(modifiedTutor, tutor.get(),
"id","imagen");
                    tutorService.save(tutor.get());
                    model.addAttribute("message","Tutor actualizado con exito");
                    return listTutores(model);
              }
	}

	@GetMapping("/{id}")
	public String tutorDetails(@PathVariable("id") int id, ModelMap model) {
		Optional<Tutor> tutor = tutorService.findById(id);

	if(tutor.get().getEmail().equals(SecurityContextHolder.getContext().getAu
thentication().getName()) || Utils.authLoggedIn().equals("administrador")) {
                    model.addAttribute("me",true);
              }else {
                    model.addAttribute("me",false);
              }

              if(tutor.isPresent()) {
                    model.addAttribute("tutor", tutor.get());

	model.addAttribute("preguntasTutor",preguntaTutorService.findByProblemaNo
tAnswered());
                    return "tutores/tutorDetails";

              }else {
                    model.addAttribute("message", "El tutor al que intenta
acceder no existe");
                    return listTutores(model);
              }
	}
}
```

El código que queda expuesto arriba hace referencia al controlador de la entidad Tutor, este controlador tiene varios métodos, los cuales se pueden agrupar en los que son “tutores/{id}/edit”, “tutores/{id}”, “tutores”, “tutores/new”, en el método que tiene asociado el RequestMapping “tutores” se encarga de enviar al modelo toda la lista de tutores, la cual está paginada, los métodos asociados con los RequestMapping “tutores/new” se encarga enviar al modelo los campos a rellenar para poder crear un tutor, también en el proceso de crear un tutor, se da alta como usuario de la página, porque hay que diferenciar entre crear un tutor y crear una cuenta de tutor, en este proceso de crear un tutor, también se da de alta en la página. El método que tiene asociado el RequestMapping “tutores/{id}”, se encarga de enviar al modelo los datos del tutor asociado a ese id, en ese método se le pasa al modelo un atributo un llamada “me” ese atributo comprueba la cuenta que está mirando dicho perfil, si es el mismo usuario se pone a true para que el propio tutor pueda modificar. Los métodos que tienen asociado el RequestMapping “tutores/{id}/edit”, en estos métodos se comprueba por un lado, que es el mismo usuario o un administrador, quien puede editar dicho perfil y una vez

comprobado, y haber rellenado los campos, si enviamos dichos datos, el método de controlador que se encarga de gestionar el post de esos datos comprueba que si se ha cambiado el email no sea el mismo que uno existente que la imagen cumpla los requisitos de tamaño y peso(esto se comprueba también a la hora de crear un tutor nuevo), el resto de datos se validan gracias a las diferentes etiquetas implementadas en el mismo modelo, si hay errores se enviara de nuevo al formulario con el mensaje de los errores, si no es así, se te devuelve a la vista del perfil de ese tutor (Esto también ocurre a la hora de crear uno nuevo)

Repositorio

Repositorio Tutor

```
package org.springframework.samples.petclinic.repository;

import java.util.Collection;
import java.util.Optional;

import org.springframework.dao.DataAccessException;
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Slice;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.Repository;
import org.springframework.data.repository.query.Param;
import org.springframework.samples.petclinic.model.Alumno;
import org.springframework.samples.petclinic.model.Tutor;

public interface TutorRepository extends Repository<Tutor, String> {

	Collection<Tutor> findAll() throws DataAccessException;

	Optional<Tutor> findById(int id) throws DataAccessException;

	void save(Tutor tutor) throws DataAccessException;

	@Query("SELECT DISTINCT t FROM Tutor t WHERE t.email LIKE :email")
	Optional<Tutor> findByEmail(@Param("email") String email);

	@Query("SELECT DISTINCT t FROM Tutor t")
	public Slice<Tutor> findTutorPage(Pageable pageable);

}
```

En este repositorio hacemos distintas llamadas que trae la propia extensión de “Repository” y dos consultas jpql en las que obtenemos un slice de todos los tutores para su posterior paginación y una consulta que devuelve el Tutor dado un determinado email

Conversor o Formatter (si aplica)

No aplica

Validador y anotación asociada (PassValidator)

Validador Pass

```
package org.springframework.samples.constraint.validators;
```

```
import javax.validation.ConstraintValidator;
import javax.validation.ConstraintValidatorContext;

import org.springframework.samples.constraint.PassConstraint;

public class PassValidator implements ConstraintValidator<PassConstraint,
String>{

	public void initialize(PassConstraint contactNumber) {
	}

	@Override
	public boolean isValid(String value, ConstraintValidatorContext context)
{
		char[] v = value.toCharArray();
		boolean numero = false;
		boolean especial = false;
		boolean mayuscula = false;
		for(int i = 0; i<v.length; i++) {
			char p = v[i];
			if(Character.isDigit(p)) {
				numero = true;
			}else if((!Character.isLetter(p))&(!Character.isDigit(p))) {
				especial = true;
			}else if(Character.isUpperCase(p)) {
				mayuscula = true;
			}

		}
		return value!=null && (v.length>=8) && numero && especial &&
mayuscula;
	}

}
```

En este validador se comprueba por un lado que la longitud de la contraseña es igual o mayor a 8 caracteres, que dicha contraseña tenga una mayúscula, un número y un carácter especial (@,#,$,%,/,&,|,etc…)

Etiquetas personalizadas (PassConstraint)

Etiquetas Pass

```
package org.springframework.samples.constraint;

import java.lang.annotation.Documented;
import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;

import javax.validation.Constraint;
import javax.validation.Payload;

import org.springframework.samples.constraint.validators.PassValidator;

@Documented
@Constraint(validatedBy = PassValidator.class)
```

```
@Target( { ElementType.FIELD})
@Retention(RetentionPolicy.RUNTIME)
public @interface PassConstraint {
	String message() default "La contraseña debe tener una longitud minima de
8, una mayuscula, un número y un caracter especial";
	Class<?>[] groups() default {};
	Class<? extends Payload>[] payload() default {};
}
```

En esta clase, gracias al validador que he creado (PassValidator), obtengo mi etiqueta personalizada para poder comprobar que la contraseña tienen el formato que nosotros queremos para que sean más seguras, junto al mensaje que saltara cuando dicha contraseña no la cumpla ("La contraseña debe tener una longitud mínima de 8, una mayúscula, un número y un carácter especial")

Ejemplos de pruebas implementadas

Pruebas unitarias (máximo de dos ejemplos)

Un test de la prueba TutorServiceTests

```
@Test
	void shoulInsertNoticia() {
		Tutor tutor = this.tutorService.findById(0).get();
		Integer numNoticiasAntiguos =
noticiaService.findNoticiasByTutor(tutor.getId()).size();
		Set<Tutor> autores = new HashSet<Tutor>();
		autores.add(tutor);

		Noticia noticiaNueva = new Noticia();
		noticiaNueva.setAutores(autores);
		noticiaNueva.setAutor(tutor);
		noticiaNueva.setFechaPublicacion(LocalDate.now());

	noticiaNueva.setImagen("https://eina.unizar.es/sites/eina.unizar.es/files
/concurso_adabyron.jpg");
		noticiaNueva.setName("Concurso de AdaByron");
		noticiaNueva.setTexto("It is a long established fact that a reader
will be distracted by the readable content of a page when looking at its layout.
The point of using Lorem Ipsum is that it has a more-or-less normal distribution
of letters, as opposed to using 'Content here, content here', making it look like
readable English. Many desktop publishing packages and web page editors now use
Lorem Ipsum as their default model text, and a search for 'lorem ipsum' will
uncover many web sites still in their infancy. Various versions have evolved over
the years, sometimes by accident, sometimes on purpose (injected humour and the
like).");
		noticiaService.save(noticiaNueva);

		tutor = this.tutorService.findById(0).get();
		Integer numNoticiasNuevo =
noticiaService.findNoticiasByTutor(tutor.getId()).size();
		assertThat(numNoticiasNuevo).isEqualTo(numNoticiasAntiguos+1);
		assertNotEquals(numNoticiasAntiguos, numNoticiasNuevo, "El número
de noticias asociado a este tutor no es correcto");
	}
```

Justificación:
- Arrange: crear una noticia con datos válidos.
- Act: guardar la noticia en la base de datos.
- Assert: que este tutor tenga asociado una noticia más de las que tenía antes,

-Una prueba en la que se crea una noticia, que ha creado un tutor concreto, y ver que al crearla el tutor tiene una noticia más de las que tenía anteriormente.

Un test de la prueba del PublicacionServiceTests

```
@Test
	public void shouldNotInsertPublicacion() {
		Publicacion publicacion = new Publicacion();
		publicacion.setAlumno(alumnoService.findById(0).get());
		publicacion.setFecha(LocalDateTime.now());
		publicacion.setTexto("");

		Assertions.assertThrows(ConstraintViolationException.class, ()-> {
			publicacionService.save(publicacion);
		});

	}
```

Justificación:
- Arrange: crear una publicación con datos no válidos.
- Act: guardar la publicación en la base de datos.
- Assert: que esta publicación haga saltar un ConstraintViolationException,

-Una prueba en el que se trata de crear una publicación para un problema concreto, pero al crearla, capturamos una "ConstraintViolationException" ya que anteriormente en el modelo hemos estipulado que no se puede enviar ninguna publicación con un texto vacío.

Pruebas unitarias parametrizadas (si aplica)

No aplica

Pruebas de controlador

Un test de la prueba del NoticiaControllerTests

```
@WithMockUser(username = "jesus@us.es", authorities = "tutor")
	@Test
	void testProcessCreationFormSuccess()throws Exception{
		byte[] somebytes = { 1, 5, 5, 0, 1, 0, 5 };
		mockMvc.perform(MockMvcRequestBuilders.multipart("/noticias/new")
                                                .file(new
MockMultipartFile("image","resources/images/noticias/20201223178419180000000.jpg
", "text/plain", somebytes))
                                                .with(csrf())
                                                .param("autores", "3")
                                                .param("_autores", "on")
                                                .param("name", "Concurso problemas
recursivos")
                                                .param("texto", "El proximo 10 de
febrero tendrá lugar un concurso en la Universidad de Sevilla sobre problemas
recursivos"))
```

```
                .andExpect(status().is3xxRedirection())
                .andExpect(view().name("redirect:/noticias/"));
        }
```

-En esta prueba de se comprueba que un tutor con correo jesus@us.es, puede crear una noticia nueva y asignársela, una vez rellenado los campos, se comprueba el éxito de la prueba, ya que podemos ver que re direcciona a la vista con todo el resto de noticias, de lo contrario se hubiera quedado en el mismo formulario.

Un test de la prueba del TutorControllerTests

```
@WithMockUser(value = "spring", authorities = "administrador")
        @Test
        void testProcessCreationFormFailure() throws Exception {
                byte[] somebytes = { 1, 5, 5, 0, 1, 0, 5 };
                mockMvc.perform(MockMvcRequestBuilders.multipart("/tutores/new")
                                        .file(new MockMultipartFile("image","file.jpg",
"image/jpeg", somebytes))
                                        .with(csrf())
                                        .param("nombre", "Juanra")
                                        .param("apellidos", "Ostos")
                                        .param("email", "hulalalala@gmail.es")
                                        .param("pass", "Estacontraseñanonoesvalida"))
                .andExpect(status().isOk())
                .andExpect(view().name("tutores/createOrUpdateTutorForm"));
        }
```

-En esta prueba se intenta crear un nuevo tutor, al rellenar los campos y hacer un post la prueba se queda en la misma página con el formulario, ya que como podemos ver, la contraseña no tiene todos los requisitos que se le piden, de lo contrario se iría al perfil de dicho tutor.

- Es importante saber que al realizar las pruebas de controlador se crean mocks con la etiqueta @MockBean de los servicios que se van a mockear de forma que se simulen para que pueda tener toda la información necesaria. También un @BeforeEach se crean objetos que se van a devolver cuando en el controlador se requieran datos de los servicios mockeados. Sólo se simulan con given los métodos que devuelven algún valor.

Principales problemas encontrados

-A la hora de organizar el proyecto no teníamos, en general, muy claro que estábamos aplicando Scrum y sabiendo de antemano, hubiéramos intentado aplicarlo más para que después a la hora de realizar documentos y la carga de trabajo, todo hubiera estado más organizado.

-Cuando hemos realizado las pruebas de controlador, tuvimos problemas a la hora de mockear tipo de imagen, pero lo solucionamos creando una lista de bits y luego utilizando el método MockMultipartFile, para mockear las fotos de los tutores

-Cuando realicé las pruebas del controlador de Noticia tuve problemas a la hora de mockear su atributo de autores. Noticia es una entidad que tiene una relación n:n con Tutor, donde noticia tiene un atributo que es “autores” un Set con las id´s de los tutores, y

al no saber cómo mockearlo tuve que utilizar la aplicación Wireshark, para ver que se enviaba a la hora de hacer una nueva noticia, y así pude conseguir los datos que se enviaban en el formulario para después ponerlos en las pruebas el Set de id´s.

-Respecto a porque en el proyecto no hay ningún tipo de competición ni nada, es porque sopesando lo de las competiciones nos dimos cuenta que era una parte que iba a ser muy difícil de programar para lo que aportaba al proyecto, por eso tomamos la decisión de quitarlo de nuestro proyecto, aunque yo hubiera implementado tanto el modelo, servicios, controladores y vistas del mismo, por eso he visto conveniente que aunque no esté en el proyecto, incluirlo en este documento para que se vea reflejado.

-También el hecho de que este informe individual solo se avisara con semanas de antelación, antes de su entrega, ha hecho que yo y otros compañeros no tengamos tan claro que hemos realizado y sobretodo con este proyecto al tener tal cantidad de código. Y eso ha hecho que no pueda reflejar todo mi trabajo como me gustaría, al contrario, si hubiera sabido de esto, y apuntar de una forma más detallada todo lo que iba haciendo.

-A la hora de hacer las estadísticas no sabía muy bien cómo hacerlo, ya que no sabía si hacerlo con una api, pero tal como se refleja en el documento 3, tome la decisión de hacer uso de una librería de javascript de libre uso

Otros comentarios

-Investigación y desarrollo del uso de la librería javascript Morris.js para la implementación de graficas personalizables

-Recalcar que, aunque en una menor medida, me he pasado por todas las clases del proyecto, ya sea para arreglar errores puntuales o añadir algún que otro método.

Detailed report

01/09/2020 - 14/02/2021

Total: 140:13:14 Billable: 00:00:00 Amount: 0.00 USD

Date	Description	Duration	User
09/02/2021	Arreglando pruebas del servicio Logro CodeUs G2-L6	02:19:00 11:00:00AM - 01:19:00PM	Juaostrub
08/02/2021	Trabajando en la documentación y en el reporte individual CodeUs G2-L6	05:30:00 04:00:00PM - 09:30:00PM	Juaostrub
07/02/2021	Realizando documentación, revisando página y nueva carga de base de datos CodeUs G2-L6	04:30:00 04:30:00PM - 09:00:00PM	Juaostrub
07/02/2021	Realizando la documentación CodeUs G2-L6	02:45:00 11:30:00AM - 02:15:00PM	Juaostrub
06/02/2021	Completando las historias de usuario: H1, H3, H4, H5, H6 CodeUs G2-L6	02:30:00 11:00:00AM - 01:30:00PM	Juaostrub
05/02/2021	Realizado la regla de negocio 2 y creando el documento del final delivery CodeUs G2-L6	04:00:00 09:30:00AM - 01:30:00PM	Juaostrub
04/02/2021	Completando la politica de logs, cambio en el menu y retoques en varias vistas CodeUs G2-L6	03:50:00 11:30:00PM - 03:20:00AM	Juaostrub
04/02/2021	Completando la politica de logs, cambio en el menu y retoques en varias vistas CodeUs G2-L6	01:00:00 11:00:00PM - 12:00:00AM	Juaostrub
04/02/2021	Terminado pruebas de servicios: Tutor, Noticia, Publicacion, Administrador, Logro; retoque en el fichero de seguridad y cambiada la funcion imagecrop CodeUs G2-L6	04:00:00 04:30:00PM - 08:30:00PM	Juaostrub
03/02/2021	Realizacndo pruebas de controlador de Noticia y preguntaTutor CodeUs G2-L6	04:30:00 04:30:00PM - 09:00:00PM	Juaostrub
03/02/2021	Prueba de controlador de Noticia y carga de datos CodeUs G2-L6	03:20:00 10:40:00AM - 02:00:00PM	Juaostrub

02/02/2021	Corregido errores en las estadísticas y haciendo la política de logs CodeUs G2-L6	02:26:00 11:30:00PM - 01:56:00AM	Juaostrub
02/02/2021	avances en la politica de logs, retoque de prueba de controlador Tutor, nueva carga de base de datos CodeUs G2-L6	03:43:00 11:00:00AM - 02:43:00PM	Juaostrub
01/02/2021	Realizada la prueba de controlador de Tutor y echando un vistazo a la política de logs CodeUs G2-L6	04:10:00 04:00:00PM - 08:10:00PM	Juaostrub
01/02/2021	Realizando pruebas de controladores CodeUs G2-L6	03:35:00 10:40:00AM - 02:15:00PM	Juaostrub
11/01/2021	Mejorando la entidad Logro CodeUs G2-L6	01:30:00 11:10:00AM - 12:40:00PM	Juaostrub
10/01/2021	Realizada la entidad Logro, sus pruebas y completada las vistas CodeUs G2-L6	03:30:00 06:00:00PM - 09:30:00PM	Juaostrub
10/01/2021	Realizada algo de la prueba de controlador de la entidad Tutor CodeUs G2-L6	04:30:00 04:30:00PM - 09:00:00PM	Juaostrub
10/01/2021	Reajuste del proyecto, creacion de la relacion n:n Tutor-Noticia, creacion de la entidad Logro CodeUs G2-L6	04:00:00 12:00:00AM - 04:00:00AM	Juaostrub
08/01/2021	Pruebas 3º CodeUs G2-L6	02:00:00 11:00:00PM - 01:00:00AM	Juaostrub
08/01/2021	Instalacion Docker y realizacion de pruebas en el controlador CodeUs G2-L6	03:00:00 06:00:00PM - 09:00:00PM	Juaostrub
08/01/2021	Login en la pagina por parte de un alumno CodeUs G2-L6	02:10:00 12:40:00PM - 02:50:00PM	Juaostrub
07/01/2021	Pruebas 2º CodeUs G2-L6	04:30:00 05:00:00PM - 09:30:00PM	Juaostrub
07/01/2021	Pruebas 1º CodeUs G2-L6	02:20:00 12:40:00PM - 03:00:00PM	Juaostrub
04/01/2021	Realizado Validadores de Contraseña y FechaFin CodeUs G2-L6	02:50:00 10:40:00AM - 01:30:00PM	Juaostrub
28/12/2020	Terminado las estadísticas del problema CodeUs G2-L6	09:00:00 11:00:00AM - 08:00:00PM	Juaostrub

23/12/2020	Realizando el apartado de estadisticas de problema CodeUs G2-L6	04:00:00 05:00:00PM - 09:00:00PM	Juaostrub
22/12/2020	Finalizada la paginación de la vista del perfil del tutor CodeUs G2-L6	02:10:00 04:30:00PM - 06:40:00PM	Juaostrub
22/12/2020	paginación de la vista del perfil del tutor 2º CodeUs G2-L6	01:30:00 04:00:00PM - 05:30:00PM	Juaostrub
22/12/2020	paginación de la vista del perfil del tutor 3º CodeUs G2-L6	02:00:00 12:00:00PM - 02:00:00PM	Juaostrub
21/12/2020	Creación de la entidad Administrador y paginación de la vista del perfil del tutor 1º CodeUs G2-L6	03:10:00 10:50:00AM - 02:00:00PM	Juaostrub
20/12/2020	Terminado la relacion n:n Articulos-Tutores y sus vistas 2º CodeUs G2-L6	01:50:00 11:00:00PM - 12:50:00AM	Juaostrub
19/12/2020	Terminado la relacion n:n Articulos-Tutores y sus vistas 1º CodeUs G2-L6	03:20:00 10:40:00AM - 02:00:00PM	Juaostrub
17/12/2020	Realizacion de la relacion n:n Articulos-Tutores 2º CodeUs G2-L6	03:00:00 06:00:00PM - 09:00:00PM	Juaostrub
16/12/2020	Realizacion de la relacion n:n Articulos-Tutores CodeUs G2-L6	02:00:00 12:00:00PM - 02:00:00PM	Juaostrub
05/12/2020	Cambio a la hora de mostrar los problemas CodeUs G2-L6	00:30:00 08:00:00PM - 08:30:00PM	Juaostrub
05/12/2020	Creacion de la entidad Competicion y algunas vistas de las mismas CodeUs G2-L6	02:22:00 05:00:00PM - 07:22:00PM	Juaostrub
05/12/2020	Tutores - Relaciones OneToMany y vista actualizadas CodeUs G2-L6	00:30:00 12:00:00AM - 12:30:00AM	Juaostrub
28/11/2020	Pruebas unitarias noticias CodeUs G2-L6	01:00:00 07:00:00PM - 08:00:00PM	Juaostrub
28/11/2020	Terminado la entidad Alumno y añadida la vista de Noticia CodeUs G2-L6	03:00:00 03:00:00PM - 06:00:00PM	Juaostrub
28/11/2020	Completar los campos de la entidad tutor CodeUs G2-L6	01:00:00 12:00:00AM - 01:00:00AM	Juaostrub

21/11/2020	Creacion entidad tutor#2 CodeUs G2-L6	04:00:00 11:00:00PM - 03:00:00AM	Juaostrub
20/11/2020	Creacion entidad tutor#1 CodeUs G2-L6	02:00:00 12:00:00PM - 02:00:00PM	Juaostrub
14/11/2020	Creación de mockups CodeUs G2-L6	03:10:00 12:00:00PM - 03:10:00PM	Juaostrub
13/11/2020	Construcción del modelo de negocio CodeUs G2-L6	00:53:14 01:37:18PM - 02:30:32PM	Juaostrub
04/11/2020	Puesta en común y descripción general del proyecto. CodeUs G2-L6	01:30:00 01:00:00PM - 02:30:00PM	Juaostrub
03/11/2020	Creación de Entidad / Controlador / Repositorio / Servicio / Vista de Noticia CodeUs G2-L6	02:00:00 11:00:00PM - 01:00:00AM	Juaostrub
02/11/2020	Creación de Entidad / Controlador / Repositorio / Servicio / Vista de Noticia CodeUs G2-L6	02:00:00 10:00:00AM - 12:00:00PM	Juaostrub
19/10/2020	Enumeración de entidades, roles e historias de usuario. CodeUs G2-L6	01:50:00 10:40:00AM - 12:30:00PM	Juaostrub