Nombre: David Brincau Cano Grupo: G2-06

Datos generales

URL del Repositorio de GitHub: https://github.com/gii-is-DP1/dp1-2020-g2-06.git
Nombre de usuario en GitHub: davbrican
Rama del usuario en Github: davbrican

Participación en el proyecto

Historias de usuario en las que he participado

He implementado completas las historias de usuario HU-1, HU-2, HU-26, HU-16, HU-19 y HU-25, además, he desarrollado junto a mi compañero vicgragil y juaostrub la historias de usuario HU-4, HU-5, HU-6 y con vicgragil HU-39, HU-40.
También he ayudado en HU-13 y HU-20
Finalmente, implementé 2 historias de usuario que finalmente decidimos eliminar. Una de ellas es similar a la 11, que consistía en que cada alumno pudiese indicar la dificultad del problema y este se indicaba con una media global; y otra consistía en la entidad competición, que agruparía ciertos problemas, tendría puntuación propia… y al 60% del desarrollo aproximadamente decidimos que era muy lioso y preferimos no acabarla.

Funcionalidad implementada

He implementado la entidad, repositorio, servicio y controlador de Creador y Artículo junto con vicgragil. He implementado también con vicgragil la funcionalidad de artículos y creador, de creación y edición de los mismos, así como las vistas de ambos. También he creado eso mismo de forma principalmente individual, de Aclaración, Comentario, PuntuacionProblema y Competiciones.
Puntuar un problema y las competiciones, poco después de hacerla, decidimos que no formase parte de la página y se borró lo que había hasta el momento, que eran todas las clases necesarias, y la vista y formularios de puntuaciones en la vista problemas.
He colaborado en solución de errores y modificación de vistas como perfil, problema y lista de alumnos.
Además, he paginado la lista de creadores y de publicaciones en el foro en sus correspondientes vistas, y la vista de verificación del email
También, he investigado e implementado la funcionalidad para el A+ de envio de correo electrónico por el protocolo SMTP, con la ayuda de alebarled en un problema que tuve, así como, gracias a esto, la verificación del correo electrónico.
Finalmente, he diseñado las imágenes de los logros, y ayudado en la creación de los mismos, así como ayudado en la solución de ciertos errores y mejoras en aspectos como el hecho de no poder editar el correo de un alumno desde el perfil para evitar errores.

Esto hace un total de 18 clases implementadas por mí y 6 interfaces definidos.

Pruebas implementadas

Pruebas unitarias

He creado tests unitarios para 2 servicios (CreadorService, ComentarioService), 4 controladores (CreadorController, ComentarioController, PublicacionController, NormaWebController). Eso hace un total de 6 clases de test unitarios con un total de 16 métodos anotados con @Test.

Pruebas de Controlador

He creado 8 casos de prueba positivo para HU-1, HU-13, HU-26, HU-27, HU-28, HU-29, HU-30, HU-31
15 negativos de controlador para la HU-1, HU-12, HU-13, HU-26, HU-27, HU-28, HU-29, HU-30, HU-31

Ejemplos de funcionalidades implementadas

Entidades (máximo de dos ejemplos)

src/main/java/org/springframework/samples/petclinic/model/Aclaracion.java
Aclaracion

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

import javax.persistence.Entity;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;
import javax.validation.constraints.NotEmpty;

import lombok.EqualsAndHashCode;
import lombok.Getter;
import lombok.NonNull;
import lombok.Setter;

@Getter
@Setter
@EqualsAndHashCode(callSuper=true)
@Entity
@Table(name="aclaraciones")
public class Aclaracion extends BaseEntity{

	/*Cada aclaración está asociada a 1 tutor, y cada uno tiene varias aclaraciones, por lo que le asociamos el parametro Tutor con una relación ManyToOne*/
	@ManyToOne
	@JoinColumn(name="id_tutor")
	private Tutor tutor;

	/*Cada problema tiene asociado muchas aclaraciones*/
	@ManyToOne
	@JoinColumn(name="id_problema")
	private Problema problema;

	/*La aclaración es un texto que escribe un tutor, por lo que debe tener un String texto*/
	@NotEmpty
	private String texto;


}
```

Servicio

src/main/java/org/springframework/samples/petclinic/service/AclaracionService.java
AclaracionService

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

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

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.samples.petclinic.model.Aclaracion;
import org.springframework.samples.petclinic.repository.AclaracionRepository;
import org.springframework.stereotype.Service;

@Service
public class AclaracionService {

	@Autowired
	private AclaracionRepository aclaracionRepository;

	public Collection<Aclaracion> findAll(){
		return aclaracionRepository.findAll();
	}

	public Optional<Aclaracion> findById(int id){
		return aclaracionRepository.findById(id);
	}

	public void save(Aclaracion aclaracion) {
		aclaracionRepository.save(aclaracion);
	}

	public Collection<Aclaracion> findAllByTutor(int id){
		return aclaracionRepository.findAllByTutor(id);
	}

	public Collection<Aclaracion> findAllByProblema(int id){
		return aclaracionRepository.findAllByProblema(id);
	}

}
```

Controlador

src/main/java/org/springframework/samples/petclinic/web/AclaracionController.java
AclaracionController

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

import java.io.IOException;

import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.samples.petclinic.model.Aclaracion;
import org.springframework.samples.petclinic.service.AclaracionService;
import org.springframework.samples.petclinic.service.ProblemaService;
```

```
import org.springframework.samples.petclinic.service.TutorService;
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.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;

@Controller
@RequestMapping("/aclaraciones")
public class AclaracionController {

	@Autowired
	ProblemaService problemaService;

	@Autowired
	TutorService tutorService;

	@Autowired
	private AclaracionService aclaracionService;

	/*Para crear una aclaración se accede desde el formulario de
problemaDetails. Por ello, invocamos a un método post /aclaraciones/new con el
cual
	 * comprobamos si el Binding realizado en el formulario corresponde con los
datos a insertar, o si tiene errores, y en caso contrario
	 * se crea una aclaración con los datos de entrada recurriendo a los
servicios necesarios*/
	@PostMapping("/new")
	public String processCreationForm(@Valid Aclaracion aclaracion,
BindingResult result, ModelMap model, @RequestParam("idProblema") Integer
idProblema) throws IOException {
		if(result.hasErrors()) {

	model.addAttribute("message",result.getFieldError().getField());
			return "/problemas/"+idProblema;
		} else {

	aclaracion.setProblema(problemaService.findById(idProblema).get());

	aclaracion.setTutor(tutorService.findByEmail(SecurityContextHolder.getConte
xt().getAuthentication().getName()).get());
			this.aclaracionService.save(aclaracion);
			return "redirect:/problemas/"+idProblema;
		}
	}

}
```

Repositorio

src/main/java/org/springframework/samples/petclinic/repository/AclaracionRepository.java

AclaracionRepository

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

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

import org.springframework.dao.DataAccessException;
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.Aclaracion;

public interface AclaracionRepository extends Repository<Aclaracion, String>{
	/*Los siguientes 3 métodos son implementados con la extensión Repository, y las 2 últimas, son consultas SQL creadas por mí para obtener lo que necesitaba*/
	Collection<Aclaracion> findAll() throws DataAccessException;

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

	void save(Aclaracion aclaracion) throws DataAccessException;

	@Query(value="SELECT * FROM aclaraciones WHERE aclaraciones.id_tutor=:id",
nativeQuery = true)
	Collection<Aclaracion> findAllByTutor(@Param("id") int id) throws
DataAccessException;

	@Query(value="SELECT * FROM aclaraciones WHERE
aclaraciones.id_problema=:id", nativeQuery = true)
	Collection<Aclaracion> findAllByProblema(@Param("id") int id) throws
DataAccessException;

}
```

Conversor o Formatter (si aplica)

No aplica

Validador y anotación asociada (si aplica, máximo de dos ejemplos)

No aplica

Etiquetas personalizadas (si aplica, máximo de dos ejemplos)

No aplica

Ejemplos de pruebas implementadas

Pruebas unitarias (máximo de dos ejemplos)

src/test/java/org/springframework/samples/petclinic/service/ComentarioServiceTests.java
ComentarioServiceTest

```
package org.springframework.samples.petclinic.service;
import static org.assertj.core.api.Assertions.assertThat;

import java.util.Collection;

import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.autoconfigure.orm.jpa.DataJpaTest;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.samples.petclinic.model.Comentario;
import org.springframework.stereotype.Service;
```

```java
@DataJpaTest(includeFilters = @ComponentScan.Filter(Service.class))
public class ComentarioServiceTests {

		@Autowired
		ComentarioService comentarioService;

		@Autowired
		AlumnoService alumnoService;

		@Autowired
		EnvioService envioService;

		/*Haciendo una llamada a la funcion findAll() de ComentarioService, y compruebo que haya comentarios, para lo cual hago un assertThat, que compruebe que el tamaño sea mayor que 0*/
		@Test
		public void shouldFindAll() {

	assertThat(comentarioService.findAll().size()).isGreaterThan(0);
		}

		/*Haciendo una llamada a la funcion findAll() de ComentarioService, tomo el tamaño de la lista, y tras hacer uso de la llamada save, compruebo que haya cambiado el tamaño*/
		@Test
		public void shouldInsertComentario() {
			Collection<Comentario> comentarios = this.comentarioService.findAll();
			int found = comentarios.size();

			Comentario comentario = new Comentario();

			comentario.setAlumno(alumnoService.findById(0).get());
			comentario.setTexto("Muy buen test");
			comentario.setEnvio(envioService.findById(1).get());


			this.comentarioService.save(comentario);
			comentarios = this.comentarioService.findAll();

			assertThat(comentarios.size()).isEqualTo(found +1);
		}

		/*Haciendo uso de la funcion getById del servicio, tomo un comentario, y compruebo que sus parámetros sean los correctos*/
		@Test
		public void shouldFindById() {
			Comentario c = comentarioService.findById(0).get();
			assertThat(c.getEnvio().getId()).isEqualTo(10);
			assertThat(c.getAlumno().getId()).isEqualTo(0);
			assertThat(c.getTexto()).isEqualTo("Muy buena resolución del ejercicio, muy simple y muy claro.");
		}

		/*Haciendo una llamada a la funcion findAllByEnvio() de ComentarioService, compruebo que haya la cantidad de comentarios adecuadas para ese envio*/
		@Test
```

```
	public void shouldFindAllByEnvio() {

	assertThat(comentarioService.findAllByEnvio(10).size()).isEqualTo(2);
	}

	/*Haciendo una llamada a la funcion findAllByEnvio() de
ComentarioService, compruebo que haya la cantidad de comentarios adecuadas para
ese alumno*/
	@Test
	public void shouldFindAllByAlumno() {

	assertThat(comentarioService.findAllByAlumno(2).size()).isEqualTo(2);
	}



}
```

Pruebas unitarias parametrizadas (si aplica)

No aplica

Pruebas de controlador

src/test/java/org/springframework/samples/petclinic/web/NormaWebControllerTests.java
NormaWebControllerTests

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

import static org.mockito.BDDMockito.given;
import static
org.springframework.security.test.web.servlet.request.SecurityMockMvcRequestPostPr
ocessors.csrf;
import static
org.springframework.test.web.servlet.request.MockMvcRequestBuilders.get;
import static
org.springframework.test.web.servlet.request.MockMvcRequestBuilders.post;
import static
org.springframework.test.web.servlet.result.MockMvcResultMatchers.status;
import static
org.springframework.test.web.servlet.result.MockMvcResultMatchers.view;

import java.util.ArrayList;
import java.util.Optional;

import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.extension.ExtendWith;
import org.mockito.Mockito;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.boot.test.mock.mockito.MockBean;
import org.springframework.samples.petclinic.model.NormaWeb;
import org.springframework.samples.petclinic.model.Tutor;
import org.springframework.samples.petclinic.service.AclaracionService;
import org.springframework.samples.petclinic.service.FileService;
import org.springframework.samples.petclinic.service.NormaWebService;
import org.springframework.samples.petclinic.service.ProblemaService;
import org.springframework.samples.petclinic.service.TutorService;
import org.springframework.security.test.context.support.WithMockUser;
```

```
import
org.springframework.security.test.web.servlet.setup.SecurityMockMvcConfigurers;
import org.springframework.test.annotation.DirtiesContext;
import org.springframework.test.context.junit.jupiter.SpringExtension;
import org.springframework.test.web.servlet.MockMvc;
import org.springframework.test.web.servlet.setup.MockMvcBuilders;
import org.springframework.web.context.WebApplicationContext;


@ExtendWith(SpringExtension.class)
@SpringBootTest(webEnvironment=SpringBootTest.WebEnvironment.MOCK)
@DirtiesContext
public class NormaWebControllerTests {


	private static final int TEST_NORMA_ID = 0;


	private MockMvc mockMvc;

	 @Autowired
	    private WebApplicationContext context;

	@MockBean
	private TutorService tutorService;

	@MockBean
	private NormaWebService normaWebService;

	@MockBean
	private FileService fileService;

	@MockBean
	private ProblemaService problemaService;

	@MockBean
	private AclaracionService aclaracionService;

	private NormaWeb normaWeb;

	/*Este metodo se ejecuta antes de cada test, para inicializar mockeando los
nuevos datos con los que comprobaremos todo.
	 * Aquí se crea un tutor, con una norma asociada y se emplea los given
	 * para simular datos.*/
	@BeforeEach
		void setup() {
		mockMvc = MockMvcBuilders
				.webAppContextSetup(context)
				.apply(SecurityMockMvcConfigurers.springSecurity())
				.build();

			Tutor tutor = new Tutor();
			tutor.setEmail("davbrican@us.es");
			tutor.setId(0);
			tutor.setApellidos("Brincau");
			tutor.setEnabled(true);
			tutor.setNombre("David");
			tutor.setPass("123SADgfh@");
```

```
	tutor.setImagen("/static/resources/images/noticias/2020122317827911000000.jpg");
			Optional<NormaWeb> n = Optional.empty();

			normaWeb = new NormaWeb();
			normaWeb.setId(TEST_NORMA_ID);
			normaWeb.setAutor(tutor);
			normaWeb.setName("RESPETAR");
			normaWeb.setDescripcion("Respeta a todo el mundo por igual");
			n = Optional.of(normaWeb);

			given(this.normaWebService.findAll()).willReturn(new ArrayList<NormaWeb>());

	given(this.normaWebService.findById(TEST_NORMA_ID)).willReturn(n);
			given(this.tutorService.findAll()).willReturn(new ArrayList<Tutor>());

	given(this.tutorService.findByEmail(Mockito.anyString())).willReturn(Optional.of(tutor));

	}

	//Historia de usuario 29 y 30 caso positivo
	/*Con mockMvc.perform _____  .andExpect(status().isOk());  comprobamos que esas vistas son accesibles siendo un tutor, es decir, que un tutor puede solicitar
	 * accesoa  los formularios de creacion, edicion y borrado de normas.*/
	@WithMockUser(username = "davbrican@us.es", authorities = "tutor")
	@Test
	void testcomprobarUrls() throws Exception {
		mockMvc.perform(get("/normasWeb")).andExpect(status().isOk());
		mockMvc.perform(get("/normasWeb/new")).andExpect(status().isOk());

	mockMvc.perform(get("/normasWeb/0/edit")).andExpect(status().isOk());

	mockMvc.perform(get("/normasWeb/0/delete")).andExpect(status().isOk());
	}

	//Historia de usuario 29 caso positivo
	/*Aquí comprobamos que el formulario de creacion y edicion es accesible para un tutor*/
	@WithMockUser(value = "spring", authorities = "tutor")
	@Test
	void testInitCreateFormSuccess() throws Exception{
		mockMvc.perform(get("/normasWeb/new")).andExpect(status().isOk())
		.andExpect(view().name("normasWeb/createOrUpdateNormaWebForm"));
	}

	//Historia de usuario 29 caso negativo
	/*Aquí comprobamos que el formulario de creacion y edicion no es accesible para un alumno, creador ni administrador*/
	@WithMockUser(value = "spring", authorities = {"alumno", "creador", "administrador"})
	@Test
	void testInitCreateFormFailure() throws Exception{

	mockMvc.perform(get("/normasWeb/new")).andExpect(status().is4xxClientError());
```

```
	}

	//Historia de usuario 29 caso positivo
	/*Se llama al metodo post /normasWeb/new y simulando los parametros de entrada, comprobamos que se hace una redirecciona  la nueva página, señal de que se ha insertado isn errores.*/
	@WithMockUser(value = "spring", authorities = "tutor")
	@Test
	void testProcessCreationFormSuccess()throws Exception{
		mockMvc.perform(post("/normasWeb/new")
					.with(csrf())
					.param("name", "ATENCION")
					.param("descripcion", "Estar atentos durante las explicaciones"))
		.andExpect(status().is3xxRedirection())
		.andExpect(view().name("redirect:/normasWeb/"));
	}

	//Historia de usuario 30 caso negativo
	/*Se llama al metodo post /normasWeb/new y simulando los parametros de entrada, observamos que da un codigo 200 pero se envia de nuevoa  la vista del formulario
	 * debido a que el parametro descripcion no puede estar vacio.*/
	@WithMockUser(value = "spring", authorities = "tutor")
	@Test
	void testProcessCreationFormFailure()throws Exception{
		mockMvc.perform(post("/normasWeb/new")
					.with(csrf())
					.param("name", "ATENCION")
					.param("descripcion", ""))
		.andExpect(status().isOk())
		.andExpect(view().name("normasWeb/createOrUpdateNormaWebForm"));
	}

	//Historia de usuario 30 caso positivo
	/*Tratamos de editar una norma de forma correcta con un tutor, y este nos devuelve a la lista de normas tras comprobar que no hay fallos en la insercion*/
	@WithMockUser(value = "spring", authorities = "tutor")
	@Test
	void testEditNormaWebSucces()throws Exception{
		mockMvc.perform(post("/normasWeb/"+TEST_NORMA_ID+"/edit")
					.with(csrf())
					.param("name", "ATENCION")
					.param("descripcion", "Hola buenas tardes."))
		.andExpect(status().isOk())
		.andExpect(view().name("normasWeb/normasWebList"));
	}

	//Historia de usuario 30 caso negativo
	/*Tratamos de editar una norma de forma correcta pero registrados como uno de los usuarios sin acceso a este privilegio, y nos devuelve un error 403 isFrobidden*/
	@WithMockUser(value = "spring", authorities = {"alumno", "creador", "administrador"})
	@Test
	void testEditNormaWebFailure()throws Exception{
		mockMvc.perform(post("/normasWeb/"+TEST_NORMA_ID+"/edit")
					.with(csrf())
```

```
                                        .param("name", "ATENCION")
                                        .param("descripcion", "Hola"))
                .andExpect(status().isForbidden());
        }

        //Historia de usuario 31 caso negativo
        /*Accedemos al borrado de una norma, introduciendo bien tanto la id como
todos los parámetros necesarios, y siendo un tutor, y por ello nos redirige a la
pagina tomada como la pagina de
         * exito para este caso */
        @WithMockUser(username = "davbrican@us.es", authorities = "tutor")
        @Test
        void testDeleteNormaWebSucces()throws Exception{

        mockMvc.perform(get("/normasWeb/"+TEST_NORMA_ID+"/delete")).andExpect(statu
s().isOk())
                .andExpect(view().name("normasWeb/normasWebList"));;
        }

        //Historia de usuario 31 caso negativo
        /*Intentamos borrar correctamente una nroma pero con  la sesion iniciada
como alumno, creador, y administrador, y nos
         * da error de verificacion, un error del grupo 4xx ya que tenemos
prohibido ela cceso del lado del cliente.*/
        @WithMockUser(authorities = {"alumno","administrador","creador"})
        @Test
        void testDeleteNormaWebFailure()throws Exception{

        mockMvc.perform(get("/normasWeb/"+TEST_NORMA_ID+"/delete")).andExpect(statu
s().is4xxClientError());
        }
}
```

Principales problemas encontrados

Principalmente, ha habido problemas de organización de las tareas y de control del trabajo realizado, causando esto que a la hora de indicar lo que he realizado, no recuerde algunas cosas, y que además, en el reparto de tareas no haya sido equitativo al 100%.

Otros comentarios

He realizado principalmente junto con vicgragil, los diagramas y la documentación más pesada. Realmente hemos trabajado todos la documentación, pero Victor y yo hemos hecho una menor labor de desarrollo de código y más de documentación del proyecto.
Además, he realizado la verificación del correo electrónico mediante un email de verificación utilizando el protocolo SMTP. Esta es una de las tareas para el A+ hecha íntegramente por mí excepto en un problema que tuve de puertos para aplicar el protocolo donde me ayudó alebarled.

Detailed report

09/01/2020 - 02/10/2021

Total: 99:27:00 Billable: 00:00:00 Amount: 0.00 USD

Date	Description	Duration	User
02/09/2021	Documentacion grupal e individual CodeUs G2-L6	00:40:00 02:30:00AM - 03:10:00AM	David Brincau
02/08/2021	Documentacion grupal e individual CodeUs G2-L6	04:15:00 03:45:00PM - 08:00:00PM	David Brincau
02/08/2021	Documentacion grupal e individual CodeUs G2-L6	05:00:00 10:00:00AM - 03:00:00PM	David Brincau
02/07/2021	Carga de datos CodeUs G2-L6	02:30:00 07:00:00PM - 09:30:00PM	David Brincau
02/07/2021	Documentacion CodeUs G2-L6	03:00:00 04:00:00PM - 07:00:00PM	David Brincau
02/07/2021	Documentacion CodeUs G2-L6	03:00:00 11:00:00AM - 02:00:00PM	David Brincau
02/06/2021	Arreglo de fallos, normaweb test controller CodeUs G2-L6	03:50:00 05:00:00PM - 08:50:00PM	David Brincau
02/06/2021	Verificacion por token en correo CodeUs G2-L6	01:00:00 12:30:00PM - 01:30:00PM	David Brincau
02/06/2021	Verificacion por token en correo CodeUs G2-L6	01:00:00 10:00:00AM - 11:00:00AM	David Brincau
02/05/2021	Verificacion por token en correo CodeUs G2-L6	02:00:00 05:00:00PM - 07:00:00PM	David Brincau
02/05/2021	Verificacion por token en correo CodeUs G2-L6	03:00:00 10:00:00AM - 01:00:00PM	David Brincau
02/05/2021	Envio de correo de verificacion CodeUs G2-L6	02:00:00 01:00:00AM - 03:00:00AM	David Brincau
02/04/2021	Envio de correo de verificacion CodeUs G2-L6	02:00:00 05:00:00PM - 07:00:00PM	David Brincau

02/04/2021	Envio de correo de verificacion CodeUs G2-L6	02:30:00 09:30:00AM - 12:00:00PM	David Brincau
02/03/2021	Paginaciones, revision de errores y test controllers CodeUs G2-L6	02:40:00 04:50:00PM - 07:30:00PM	David Brincau
02/03/2021	Paginaciones, revision de errores y test controllers CodeUs G2-L6	01:40:00 10:20:00AM - 12:00:00PM	David Brincau
02/02/2021	Paginacion Publicaciones en foro CodeUs G2-L6	02:35:00 05:15:00PM - 07:50:00PM	David Brincau
02/02/2021	Paginacion Creadores CodeUs G2-L6	00:45:00 04:30:00PM - 05:15:00PM	David Brincau
02/02/2021	Paginacion Creadores CodeUs G2-L6	03:00:00 11:30:00AM - 02:30:00PM	David Brincau
02/01/2021	Pruebas de controlador Creador, Comentario, Publicación y Norma Web CodeUs G2-L6	02:20:00 03:40:00PM - 06:00:00PM	David Brincau
02/01/2021	Pruebas de controlador Creador, Comentario, Publicación y Norma Web CodeUs G2-L6	03:00:00 10:00:00AM - 01:00:00PM	David Brincau
01/17/2021	TestController Creador CodeUs G2-L6	03:30:00 10:30:00AM - 02:00:00PM	David Brincau
01/12/2021	Arreglar modelado de capas CodeUs G2-L6	03:00:00 04:30:00PM - 07:30:00PM	David Brincau
01/11/2021	lista de logros CodeUs G2-L6	00:30:00 11:30:00AM - 12:00:00PM	David Brincau
01/11/2021	Diagrama de capas CodeUs G2-L6	01:00:00 10:00:00AM - 11:00:00AM	David Brincau
01/10/2021	Diagrama de capas CodeUs G2-L6	01:00:00 08:00:00PM - 09:00:00PM	David Brincau
01/10/2021	Revisión y creación de entidades, tests y formularios CodeUs G2-L6	04:30:00 03:30:00PM - 08:00:00PM	David Brincau
01/10/2021	Revisión y creación de entidades, tests y formularios CodeUs G2-L6	02:00:00 12:00:00PM - 02:00:00PM	David Brincau
01/09/2021	Revisión y creación de entidades, tests y formularios CodeUs G2-L6	04:15:00 07:45:00PM - 12:00:00AM	David Brincau

01/08/2021	Revisión y creación de entidades, tests y formularios CodeUs G2-L6	04:15:00 03:30:00PM - 07:45:00PM	David Brincau
12/29/2020	Vistas y funciones en controller de aclaración y comentario CodeUs G2-L6	04:30:00 11:00:00AM - 03:30:00PM	David Brincau
12/20/2020	Entidad Aclaración CodeUs G2-L6	02:00:00 12:30:00PM - 02:30:00PM	David Brincau
12/18/2020	Complejidad de problemas CodeUs G2-L6	02:30:00 02:10:00AM - 04:40:00AM	David Brincau
12/17/2020	Entidad Comentario CodeUs G2-L6	02:30:00 11:40:00PM - 02:10:00AM	David Brincau
11/28/2020	Entidad Creador CodeUs G2-L6	02:00:00 04:00:00PM - 06:00:00PM	David Brincau
11/21/2020	Entidad Articulo CodeUs G2-L6	01:12:00 05:30:00PM - 06:42:00PM	David Brincau
11/14/2020	Creación de mockups CodeUs G2-L6	03:10:00 12:00:00PM - 03:10:00PM	David Brincau
11/04/2020	Puesta en común y descripción general del proyecto. CodeUs G2-L6	01:30:00 01:00:00PM - 02:30:00PM	David Brincau
11/04/2020	Reglas de negocio CodeUs G2-L6	01:30:00 11:00:00AM - 12:30:00PM	David Brincau
11/03/2020	Reglas de negocio CodeUs G2-L6	01:00:00 01:30:00PM - 02:30:00PM	David Brincau
10/19/2020	Enumeración de entidades, roles e historias de usuario. CodeUs G2-L6	01:50:00 10:40:00PM - 12:30:00AM	David Brincau