Nombre: Victor Javier Granero Gil Grupo: G2-6

Informe individual de actividades del proyecto

Datos generales

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

Participación en el proyecto

Historias de usuario en las que he participado

He desarrollado junto a mi compañero David lo correspondiente a la entidad “Creador” de la historia de usuario HU-1, además los dos juntos, junto a Juan Ramón desarrollamos las historias de usuario HU-4 y HU-5 y de la HU-6 lo correspondiente al apartado ”Artículos”. También junto a Jesús he desarrollado las historias de usuario H14 y H15.

He realizado la paginación con Ajax para las historias de usuario H6 (el listado de artículos) y H23 (listado de alumnos)

Funcionalidad implementada

He implementado el controlador PreguntasTutorController junto a Jesus, ArticuloController y Creador junto a David y he añadido varios métodos a los servicios CreadorService, PreguntaTutorService y ArticuloService. También he creado la entidad PreguntaTutor, y Creador y Articulo junto a David. Además, he participado en las vistas de artículo, creador y he realizado la paginación para la vista alumnosList.

Esto hace un total de clases 9 implementadas y 7 interfaces definidos.

Pruebas implementadas

Pruebas unitarias

He creado tests unitarios para 4 servicios (PreguntaTutorService, NormaWebService, NoticiaService, AclaracionService), 2 controladores (AlumnoControllerTest, ArticuloControllerTest). Eso hace un total de 6 clases de tests unitarios con un total de 29 métodos anotados con @Test.

Pruebas de Controlador

He creado 2 casos de prueba positivo y 2 negativos de controlador para la HU-4.
He creado 1 casos de prueba positivo y 1 negativos de controlador para la HU-5.
He creado 2 casos de prueba positivo de controlador para la HU-6.
He creado 1 casos de prueba positivo y 2 negativos de controlador para la HU-33.
He creado 2 casos de prueba positivo de controlador para la HU-23.
He creado 1 casos de prueba positivo y 1 negativos de controlador para la HU-24.
He creado 2 casos de prueba positivo de controlador para la HU-25.
He creado 2 casos de prueba positivo y 2 negativos de controlador para la HU-1.

Ejemplos de funcionalidades implementadas

Entidades (máximo de dos ejemplos)

PreguntaTutor
/dp1-2020-g2-06
/src/main/java/org/springframework/samples/petclinic/model/PreguntaTutor.java

Esta clase servirá para que los alumnos registrados en la página puedan realizar preguntas en los problemas de la página para que alguno de los tutores la responda. Esta clase deberá albergar un Integer con el id del alumno, un Integer con el id del problema y un String con la pregunta. Cuando algún tutor responda la pregunta el id de este y su respuesta se guardarán en un Integer y un String respectivamente.

```
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.Setter;

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


	@ManyToOne
	@JoinColumn(name="id_tutor")
	private Tutor tutor;


	@ManyToOne
	@JoinColumn(name="id_alumno")
	private Alumno alumno;


	@ManyToOne
	@JoinColumn(name="id_problema")
	private Problema problema;

	@NotEmpty(message= "El campo 'Pregunta' no puede estar vacío")
	private String pregunta;

	private String respuesta;

}
```

Servicio

Junto a David
ArticuloService
/dp1-2020-g2-06
/src/main/java/org/springframework/samples/petclinic/service/ArticuloService.java

Este servicio corresponde a la entidad Articulo, con el tendremos acceso al repositorio de artículos de la página. En este servicio creamos métodos para obtener el artículo o los artículos necesarios en las distintas vistas de la página.

Podemos obtener todos los artículos del repositorio con el método findAll en forma de Collection, pero usaremos el método findAllArticulosPage el cual nos permitirá acceder a todos los artículos de forma paginada.
En este servicio también creamos métodos para poder guardar un artículo dado (save(Articulo)), para poder encontrar un artículo específico del repositorio (findArticuloById(Integer)) y para poder borrarlo (delete(Articulo)).
Además, creamos dos métodos para poder obtener todos los archivos en los que un tutor de, id dado, ha participado (findArticulosByTutor(Integer) y findAllArticulosByTutorPage(Integer, Pageable)).

```java
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.Articulo;
import org.springframework.samples.petclinic.repository.ArticuloRepository;
import org.springframework.stereotype.Service;

@Service
public class ArticuloService {

	@Autowired
	ArticuloRepository articuloRepo;


	public Collection<Articulo> findAll(){
		return articuloRepo.findAll();
	}

	public Optional<Articulo> findById(int id){
		return articuloRepo.findById(id);
	}

	public void delete(Articulo articulo) {
		articuloRepo.deleteById(articulo.getId());
	}
```

```
	public void save(@Valid Articulo articulo) {
		articuloRepo.save(articulo);
	}

	public Collection<Articulo> findArticulosByTutor(int id) {
		return articuloRepo.findArticulosByTutor(id);
	}

	public Slice<Articulo> findArticulosByTutorPage(int id, Pageable pageable){
		return articuloRepo.findArticulosByTutorPageable(id, pageable);
	}

	public Slice<Articulo> findAllArticulosPage(Pageable pageable){
		return articuloRepo.findAllArticulosPageable(pageable);
	}
}
```

Controlador

Junto a Jesús
PreguntaTutorController
/dp1-2020-g2-06
/src/main/java/org/springframework/samples/petclinic/web/PreguntaTutorController.java

Este controlador se encargará de realizar dos Post, uno a preguntaTutor/new, cuando un alumno cree una pregunta en algún problema y otro a preguntaTutor/answer, cuando algún tutor conteste alguna pregunta.
El primero se encarga de comprobar que no hay errores y de comprobar si es un alumno el que está logueado, después de esto creará la pregunta y la guardará en el repositorio.
El segundo añadirá la respuesta en la preguntaTutor sólo si la respuesta dada por un tutor no es vacía.

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

import java.io.IOException;
import java.util.List;
import java.util.stream.Collectors;

import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.samples.petclinic.model.PreguntaTutor;
import org.springframework.samples.petclinic.service.AlumnoService;
import org.springframework.samples.petclinic.service.PreguntaTutorService;
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("/preguntatutor")
public class PreguntaTutorController {
	@Autowired
	AlumnoService alumnoService;

	@Autowired
	ProblemaService problemaService;

	@Autowired
	TutorService tutorService;

	@Autowired
	private PreguntaTutorService preguntaTutorService;

	@Autowired
	ProblemaController problemaController;

	@Autowired
	TutorController tutorController;


	@PostMapping(value = "/new")
	public String processCreationForm(@Valid PreguntaTutor preguntaTutor,
BindingResult result,ModelMap model,@RequestParam("idProblema") Integer
idProblema) throws IOException {
		if (result.hasErrors()) {
			List<String> errores = result.getAllErrors().stream().map(x-
>x.getDefaultMessage()).collect(Collectors.toList());
			model.addAttribute("message", errores);
			return problemaController.problemaDetails(idProblema,model);
		}
		else {
			String email =
SecurityContextHolder.getContext().getAuthentication().getName();

	preguntaTutor.setAlumno(alumnoService.findByEmail(email).get());

	preguntaTutor.setProblema(problemaService.findById(idProblema).get());
			preguntaTutorService.save(preguntaTutor);
			model.addAttribute("message","Pregunta enviada con éxito");
			return problemaController.problemaDetails(idProblema,model);
		}
	}

	@PostMapping(value = "/answer")
	public String answer(ModelMap model,@RequestParam("idTutor") Integer
idTutor,@RequestParam("preguntaTutor") Integer
idpreguntaTutor,@RequestParam("respuesta") String respuesta) throws IOException {
		if (respuesta.equals(" ")) {
			model.addAttribute("message","La respuesta no puede estar
vacía");
			return tutorController.tutorDetails(idTutor, model);
		}
		else {
			PreguntaTutor preguntaTutor =
preguntaTutorService.findById(idpreguntaTutor).get();
			preguntaTutor.setTutor(tutorService.findById(idTutor).get());
			preguntaTutor.setRespuesta(respuesta);
```

```
                    preguntaTutorService.save(preguntaTutor);

                    model.addAttribute("message","Respuesta realizada con éxito");

                    return tutorController.tutorDetails(idTutor, model);
                }
        }
}
```

Repositorio

Junto a David.
ArticuloRepository
/dp1-2020-g2-06
/src/main/java/org/springframework/samples/petclinic/repository/ArticuloRepository.java

Este repositorio será el responsable de dar al servicio los artículos demandados, para ello creamos las funciones findAll que devolverá una collection con todos los artículos, findById que devolverá el artículo, correspondiente a un id dado, si existiera. También deleteById para borrar un determinado artículo y save el cual será encargado de guardar artículos en el repositorio.
Por último, creamos tres funciones con querys que nos servirán para obtener todos los artículos de un tutor dado (la primera en forma de collection y la segunda en forma de un Slice para poder hacer la paginación) y la ultima la usaremos para obtener todos los artículo del repositorio de forma paginada.

```
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.Articulo;

public interface ArticuloRepository extends Repository<Articulo, Integer>{

        Collection<Articulo> findAll() throws DataAccessException;

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

        void deleteById(int id) throws DataAccessException;

        void save(Articulo articulo) throws DataAccessException;

        @Query(value="SELECT id,name,fecha_publicacion,imagen_articulo,texto FROM
Articulos as a LEFT JOIN Articulos_Autores at WHERE a.id = at.articulo_id and
at.autores_id = :id", nativeQuery = true)
        public Collection<Articulo> findArticulosByTutor(@Param("id") int id);
```

```
	@Query(value="SELECT id,name,fecha_publicacion,imagen_articulo,texto FROM
Articulos as a LEFT JOIN Articulos_Autores at WHERE a.id = at.articulo_id and
at.autores_id = :id", nativeQuery = true)
	public Slice<Articulo> findArticulosByTutorPageable(@Param("id") int id,
Pageable pageable);

	@Query(value="SELECT * FROM Articulos articulo", nativeQuery=true)
	public Slice<Articulo> findAllArticulosPageable(Pageable pageable);

}
```

Ejemplos de pruebas implementadas

Pruebas unitarias (máximo de dos ejemplos)

AclaracionServiceTest
/dp1-2020-g2-06
/src/test/java/org/springframework/samples/petclinic/service/AclaracionServiceTests.java

En este test se comprobará que el funcionamiento del método save del servicio de articulo es el correcto y que en efecto se inserta una aclaración, dada como parámetro, al repositorio.
Para ello, en el arrange, obtenemos todas las aclaraciones del repositorio, guardamos el número total y creamos una nueva aclaración.
En el act guardamos la aclaración recién creada y volvemos a guardar el número de aclaraciones.
Por último, hacemos assert de que el nuevo valor sea igual al anterior más 1.

```
		@Test
		public void shouldInsertAclaracion() {
			Collection<Aclaracion> aclaraciones =
this.aclaracionService.findAll();
			int found = aclaraciones.size();

			Aclaracion aclaracion = new Aclaracion();


			aclaracion.setProblema(problemaService.findById(0).get());
			aclaracion.setTutor(tutorService.findById(0).get());
			aclaracion.setTexto("Prueba");



			this.aclaracionService.save(aclaracion);
			aclaraciones = this.aclaracionService.findAll();

			assertThat(aclaraciones.size()).isEqualTo(found +1);

		}
```

En la siguiente prueba se comprueba que el método shouldFindAclarcionesByTutorInitial() devuelve sólo las aclaraciones respondidas por el tutor de la id dada.
Para ello el arrange está incluido en el act ya que en una línea podemos llamar al método con la id de un tutor (act) y guardar la respuesta en una collection de aclaraciones (arrange). Para terminar nos aseguramos en el assert que el tamaño de la collection creada es 1 ya que en la carga de datos inicial el tutor con id 2 está asociado a una sola aclaración.

```
	@Test
	public void shouldFindAclaracionesByTutorInitial() {
		Collection<Aclaracion> aclaraciones =
this.aclaracionService.findAllByTutor(2);
		assertThat( aclaraciones.size()).isEqualTo(1);
	}
```

Pruebas de controlador

ArticuloControllerTests
/dp1-2020-g2-06
/src/test/java/org/springframework/samples/petclinic/web/ArticuloControllerTests.java

El siguiente código corresponde al arrange de los tests de ArticuloControllerTests, en él mockeamos un artículo con un tutor como autor del mismo para las pruebas.

```
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.result.MockMvcResultMatchers.model;
import static
org.springframework.test.web.servlet.result.MockMvcResultMatchers.status;
import static
org.springframework.test.web.servlet.result.MockMvcResultMatchers.view;

import java.time.LocalDate;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.Optional;
import java.util.Set;

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.mock.web.MockMultipartFile;
import org.springframework.samples.petclinic.model.Articulo;
import org.springframework.samples.petclinic.model.Tutor;
import org.springframework.samples.petclinic.service.AclaracionService;
import org.springframework.samples.petclinic.service.ArticuloService;
```

```
import org.springframework.samples.petclinic.service.FileService;
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.request.MockMvcRequestBuilders;
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 ArticuloControllerTests {


	private static final int TEST_ARTICULO_ID = 0;


	private MockMvc mockMvc;

	 @Autowired
	    private WebApplicationContext context;

	@MockBean
	private TutorService tutorService;

	@MockBean
	private ArticuloService articuloService;

	@MockBean
	private FileService fileService;

	@MockBean
	private ProblemaService problemaService;

	@MockBean
	private AclaracionService aclaracionService;

	private Articulo articulo;


	@BeforeEach
		void setup() {
		mockMvc = MockMvcBuilders
				.webAppContextSetup(context)
				.apply(SecurityMockMvcConfigurers.springSecurity())
				.build();
			Set<Tutor> autores = new HashSet<Tutor>();
			Tutor tutor = new Tutor();
			tutor.setEmail("vicgragil@us.es");
			Optional <Articulo> n = Optional.empty();
			autores.add(tutor);
			articulo = new Articulo();
			articulo.setId(TEST_ARTICULO_ID);
```

```
                articulo.setAutores(autores);
                articulo.setFechaPublicacion(LocalDate.now());
                articulo.setImagen("/resources/images/pets.png");
                articulo.setName("Esto es una prueba");
                articulo.setTexto("Esto es un texto de prueba");
                n = Optional.of(articulo);

        given(this.articuloService.findAll()).willReturn(new
        ArrayList<Articulo>());
        given(this.articuloService.findById(TEST_ARTICULO_ID)).willReturn(n);
        given(this.articuloService.findArticulosByTutor(TEST_ARTICULO_ID)).willRetu
        rn(new ArrayList<Articulo>());
        given(this.tutorService.findAll()).willReturn(new ArrayList<Tutor>());
        given(this.tutorService.findByEmail(Mockito.anyString())).willReturn(Option
        al.of(tutor));

        }
```

En el siguiente test probamos que el controlador de artículo permite a un tutor eliminar un artículo dado por su id, para ello en el act mockeamos que el tutor hace una llamada a la url "/articulos/"+TEST_ARTICULO_ID+"/delete", siendo Test_Articulo_Id el id del artículo que se quiere borrar. Y seguidamente, en el assert comprobamos que el controlador devuelve un código 200 Ok y la vista de artículosList donde deberíamos estar después de borrar un artículo.

```
        @WithMockUser(username = "vicgragil@us.es", authorities = "tutor")
        @Test
        void testDeleteArticuloSucces()throws Exception{
               mockMvc.perform(get("/articulos/"+TEST_ARTICULO_ID+"/delete"))
               .andExpect(status().isOk())
               .andExpect(view().name("/articulos/articulosList"));;
```

En el siguiente test comprobamos, al igual que en el anterior, que un tutor tiene acceso a borrar un artículo, pero en este creamos un caso negativo de la historia de usuario al intentar borrar el artículo asociado a una id inexistente en la base de datos.
El arrange corresponde a la primera línea dentro del test, donde accedemos a la url "/articulos/"+500+"/delete" y el assert a las líneas siguientes, donde comprobamos que la respuesta del controlados nos devuelve a la lista de los artículos pero esta vez con un mensaje de error.

```
        @WithMockUser(authorities = "tutor")
        @Test
        void testDeleteArticuloFailure2()throws Exception{
               mockMvc.perform(get("/articulos/"+500+"/delete"))
               .andExpect(status().isOk())
               .andExpect(model().attribute("message", "No podemos encontrar el
               articulo que intenta borrar"))
               .andExpect(view().name("/articulos/articulosList"));
        }
```

Nombre: Victor Javier Granero Gil Grupo: G2-6

Principales problemas encontrados

Mientras desarrollábamos el proyecto decidimos en varias ocasiones modificar las historias de usuario y tuvimos que arreglar entidades añadiendo parámetros y relaciones por lo que he tenido que dedicar bastante tiempo a cambiar el diagrama de diseño.

También me he encontrado con problemas al desarrollar pruebas, más que nada de desarrollo como tal de las pruebas, la metodología la entiendo. Contando con los compañeros y con la teoría de las presentaciones he ido solucionándolos.

A la hora de pensar el proyecto queríamos añadir competiciones a la página y estuvimos unos días pensando como diseñar las relaciones entre las entidades afectadas ya que queríamos guardar los rankings de temporadas anteriores. Al final decidimos quitar las competiciones por no sobrecargarnos con exceso de trabajo ya que teníamos lo necesario para alcanzar la nota a la que aspiramos.

Otros comentarios

En este proyecto he trabajado bastante con David, intentado realizar programación en parejas y además he dedicado bastante tiempo a diagramas y documentación de historias de usuario.
En cuanto al A+ mi compañero Alejandro se encargó de investigar la paginación con Ajax y entre todos hemos implantado la paginación en las diferentes vistas de la página, yo en concreto he paginado los artículos en el apartado de artículos de la página y los alumnos de la vista de alumnos de la página.

Detailed report

09/10/2020 - 02/11/2021

Total: 92:07:00 Billable: 00:00:00 Amount: 0.00 USD

Date	Description	Duration	User
02/09/2021	Documentacion grupal e individual CodeUs G2-L6	03:50:00 11:30:00AM - 03:20:00PM	Victor
02/08/2021	Informe Individual CodeUs G2-L6	07:30:00 04:30:00PM - 12:00:00AM	Victor
02/07/2021	Diagrama de Dominio/Diseño y arreglo de pruebas CodeUs G2-L6	05:00:00 06:00:00PM - 11:00:00PM	Victor
02/06/2021	Test controller version Final CodeUs G2-L6	05:00:00 06:00:00PM - 11:00:00PM	Victor
02/05/2021	Test controller Alumno CodeUs G2-L6	05:30:00 05:30:00PM - 11:00:00PM	Victor
02/03/2021	Test controller articulos CodeUs G2-L6	05:00:00 05:00:00PM - 10:00:00PM	Victor
02/02/2021	Paginacion Alumnos y Articulos CodeUs G2-L6	05:30:00 05:30:00PM - 11:00:00PM	Victor
02/01/2021	Actualización de controladores y servicios para gestión de sesiones. CodeUs G2-L6	02:15:00 07:00:00PM - 09:15:00PM	Victor
01/31/2021	Tests Controller CodeUs G2-L6	05:00:00 05:00:00PM - 10:00:00PM	Victor
01/11/2021	Validaciones de sesión en controladores CodeUs G2-L6	01:20:00 02:00:00PM - 03:20:00PM	Victor
01/10/2021	Escenarios de las Historias de Usuario y Reglas de Negosio. Creacion PreguntaTutor CodeUs G2-L6	05:30:00 03:00:00PM - 08:30:00PM	Victor
01/10/2021	Diagrama de Dominio/Diseño CodeUs G2-L6	02:00:00 01:00:00PM - 03:00:00PM	Victor
01/09/2021	Diagrama de Dominio/Diseño CodeUs G2-L6	04:00:00 06:00:00PM - 10:00:00PM	Victor

01/08/2021	Diagrama de Dominio/Diseño CodeUs G2-L6	02:30:00 09:00:00PM - 11:30:00PM	Victor
01/08/2021	Instalacion de Docker y edicion de planificación CodeUs G2-L6	03:00:00 06:00:00PM - 09:00:00PM	Victor
01/07/2021	Pruebas de servicios CodeUs G2-L6	05:30:00 05:30:00PM - 11:00:00PM	Victor
01/07/2021	Pruebas de servicios CodeUs G2-L6	01:00:00 01:00:00PM - 02:00:00PM	Victor
12/28/2020	Arreglado github y las historias de usuario CodeUs G2-L6	01:00:00 04:00:00PM - 05:00:00PM	Victor
12/20/2020	Historias de Usuario CodeUs G2-L6	02:00:00 10:00:00AM - 12:00:00PM	Victor
12/19/2020	Historias de Usuario CodeUs G2-L6	01:30:00 05:30:00PM - 07:00:00PM	Victor
12/18/2020	Historias de Usuario CodeUs G2-L6	03:00:00 06:00:00PM - 09:00:00PM	Victor
12/05/2020	Ranking de alumnos (Historia de usuario 4) CodeUs G2-L6	01:00:00 08:00:00PM - 09:00:00PM	Victor
11/28/2020	Vista Articulo CodeUs G2-L6	02:00:00 06:30:00PM - 08:30:00PM	Victor
11/28/2020	Entidad Creador CodeUs G2-L6	02:00:00 04:00:00PM - 06:00:00PM	Victor
11/21/2020	Entidad Artículo CodeUs G2-L6	01:12:00 05:30:00PM - 06:42:00PM	Victor
11/14/2020	Creación de mockups CodeUs G2-L6	03:10:00 12:00:00PM - 03:10:00PM	Victor
11/13/2020	Construcción del modelo de negocio CodeUs G2-L6	01:00:00 01:30:00PM - 02:30:00PM	Victor
11/04/2020	Puesta en común y descripción general del proyecto. CodeUs G2-L6	01:30:00 01:00:00PM - 02:30:00PM	Victor
11/03/2020	Relaciones de Entidades CodeUs G2-L6	01:30:00 06:00:00PM - 07:30:00PM	Victor

10/19/2020	Enumeración de entidades, roles e historias de usuario. CodeUs G2-L6	01:50:00 10:40:00AM - 12:30:00PM	Victor