Evaluación y mejora para el desarrollo de software

Informa de Pruebas

Unidad 3 y 4 – Darío Ramos

Tabla de Contenido

Informe de Pruebas

1. Proyecto y Responsables

En este proyecto revisaremos un pequeño proyecto donde existen usuarios que podrán comentar en publicaciones hechas por otros usuarios, que podrán asignar tópicos a ellas, calificarlas, darles likes o dislikes, unirse a grupos, cambiar sus comentarios o desactivar sus cuentas. Está claro, sin el mayor miedo de sonar simplista, que es un sistema relativamente sencillo que busca explotar las capacidades del aseguramiento de pruebas y testeos, y, en sí, representa lo que se podría hacer en un sistema real de este tipo. Se hizo este documento con esfuerzo, y espero, se pueda valorar como la experiencia que fue utilizar este lenguaje y framework que, de cierta manera ha tenido una cierta carga por su documentación, pero que, tiene mucho contenido tan interesante como curioso. Revisemos.

Darío Guillermo Ramos Ramón – 4A TIDSM

2. Casos de Prueba

Revisemos todos los casos de prueba de este sistema de comentarios, publicaciones, usuarios, tópicos, grupos, entre otros.

Nombre	Creación de un usuario
ID	Itr1-U8
Descripción de la prueba	Crear un usuario y verificar su creación.
Caso de Uso	Creación de usuario
Descripción del caso de uso	El usuario se puede registrar utilizando datos obligatorios, como correo, contraseña, nombre; junto a algunos adicionales, para ser añadido al sistema y estar activado para el uso de su cuenta.
Datos de entrada	'name': 'Darío', 'middle_name': 'Ramos', 'last_name': 'Ramón', 'status': 'Reading books! 📖', 'genre': null,
Resultado esperado	'status': 201, 'data': isArray,
Resultado obtenido	'status': 422,
Resultado final	FAILED

```
test('Create an user.', async ({client, assert}) => {
  const response = await client.post('/users').form({
    'name': 'Darío',
    'middle_name': 'Ramos',
    'last_name': 'Ramón',
    'status': 'Reading books! 📖',
    'genre': null,
  })

  response.assertStatus(201)
  assert.isArray(response.body()['data'])
```

```
x Create an user.

  Assertion Error: expected 404 to equal 201

- Expected  - 1
+ Received  + 1

- 201
+ 404
```

```
public async store ({ request, response }) {
  await request.validate(UserValidator)
  const user = await User.create({
    name: request.input('name'),
    middle_name: request.input('middle_name', null),
    last_name: request.input('last_name'),
    status: request.input('status', null),
    genre: request.input('genre', null),
    email: request.input('email'),
    active: true,
  })
  user.save()
  response.created(
    {
      msg: (user.genre === 'male' ?
        'El usuario ha sido creado.' :
        user.genre === 'female' ? 'La usuaria ha sido creada.' : 'Se creó un usuario.'),
      data: user,
    }
  )
}
```

Nombre	Buscar un método no existente
ID	Itr2-U9
Descripción de la prueba	Busca proporcionar soluciones al usar un verbo o método que no está soportado por un controlador.
Caso de Uso	Petición a cualquier ruta
Descripción del caso de uso	El controlador maneja cualquier petición dada para cualquier verbo, las que no estén definidas regresan un mensaje de no encontrado.
Datos de entrada	…
Resultado esperado	'status': 404, ‘msg’: 'No se encontró ningún método para manejar su petición.',
Resultado obtenido	'status': 404, 'msg': 'No se encontró ningún método para manejar su petición.',
Resultado final	PASSED

```
test('Cannot find the method given.', async ({client}) => {
  const response = await client.patch('/comments')

  response.assertStatus(404)
  response.assertBodyContains({
    'msg': 'No se encontró ningún metodo para manejar su petición.',
  })
}),
```

```
public async decide ({ request, response }) {
  switch (request.intended()) {
    case 'GET':
      return this.index({request, response})
    case 'POST':
      return this.store({request, response})
    case 'PUT':
      return this.update({request, response})
    case 'DELETE':
      return this.destroy({request, response})
    default:
      return response.notFound({
        msg: 'No se encontró ningún metodo para manejar su petición.',
      })
  }
}
```

Nombre	Publicar una publicación por ruta genérica.
ID	Itr3-U10
Descripción de la prueba	Se busca publicar una publicación por una ruta incorrecta, para verificar la seguridad de la ruta.
Caso de Uso	Publicar una publicación
Descripción del caso de uso	Se puede publicar un post o una publicación por los usuarios y para ellos.
Datos de entrada	'title': 'How to prepare spaghetti 🍝!', 'content': '1.- Check the package directions of the spaghetti you plan to cook. 2.- Cook the spaghetti in salted wáter 3.- Check the package directions of the spaghetti you plan to cook.',
Resultado esperado	'status': 201, 'msg': 'La publicación ha sido creada.',
Resultado obtenido	'status': 406, 'msg': 'No utilice este método para crear posts.',
Resultado final	FAILED

```
test('Make a post from the route: (posts).', async ({client}) => {
  const response = await client.post('/posts').form({
    'title': 'How to prepare spaghetti 🍝!',
    'content': '1.- Check the package directions of the spaghetti you plan to cook.'
             + '2.- Cook the spaghetti in salted water.\n'
             + '3.- Check the package directions of the spaghetti you plan to cook.'
  })

  response.assertStatus(201)
  response.assertBodyContains({
    'msg': 'La publicación ha sido creada.',
  })
}),
```

```
× Make a post from the route: (posts).

  Assertion Error: expected 406 to equal 201

- Expected  - 1
+ Received  + 1

- 201
+ 406
```

```
public async decide ({ request, response }) {
  switch (request.intended()) {
    case 'GET':
      return this.index({request, response})
    case 'PUT':
      return this.update({request, response})
    case 'DELETE':
      return this.destroy({request, response})
    case 'POST':
      response.notAcceptable({
        msg: 'No utilize este método para crear posts.',
      })
      break
    default:
      response.notFound({
        msg: 'No se encontró ningún metodo para manejar su petición.',
      })
      break
  }
}
```

Nombre	Publicar un comentario por ruta especializada.
ID	Itr4-U11
Descripción de la prueba	Se busca publicar un comentario fuera de las rutas comunes que podrían confundir al usuario.
Caso de Uso	Publicar un comentario
Descripción del caso de uso	Se puede publicar tanto como comentarios para las publicaciones que nosotros deseemos hablar de ellas
Datos de entrada	'content': 'I love this post! 🥊', 'dislikes': 6, 'likes': 1,
Resultado esperado	'status': 201, 'data': notEmpty
Resultado obtenido	'status': 201, 'data': notEmpty
Resultado final	PASSED

```
const response = await client.post('/comments/post/1').form({
  'content': 'I love this post! 🥊',
  'dislikes': 6,
  'likes': 1,
})

response.assertStatus(201)
assert.isNotEmpty(response.body()['data'])
```

```
public async store ({ request, response }) {
  let id = request.param('id', null)
  if (id !== null) {
    let post = await Post.query().where('id', request.param('id'))
    if (post !== null) {
      await request.validate(CommentValidator)
      const comment = await Comment.create({
        content: request.input('content'),
        likes: request.input('likes', null),
        dislikes: request.input('dislikes', null),
        active: true,
      })
      comment.save()
      response.created(
        {
          msg: 'El comentario ha sido creado.',
          data: comment,
        }
      )
    } else {
      response.created(
        {
          msg: 'No se encontró la publicación a comentar.',
        }
      )
    }
  } else {
    response.notFound(
      {
        msg: 'No se ha indicado ninguna publicación a la cual contestar.',
      }
    )
```

Nombre	Añadir una calificación negativa
ID	Itr5-U12
Descripción de la prueba	Se busca valorar una publicación con una calificación negativa para probar su capacidad de manejar este error.
Caso de Uso	Calificar una publicación.
Descripción del caso de uso	Los usuarios pueden calificar las publicaciones de los demás usuarios.
Datos de entrada	'rate': -1,
Resultado esperado	'status': 406, 'rating': isNumber && isBelow(0), 'msg': 'No se pueden insertar calificaciones negativas.'
Resultado obtenido	'status': 406, 'rating': isNumber && isBelow(0), 'msg': 'No se pueden insertar calificaciones negativas.'
Resultado final	PASSED

```
const response = await client.post('/posts/1/rating').form({
  'rate': -1,
})

response.assertStatus(406)
assert.isNumber(response.body()['rating'])
assert.isBelow(response.body()['rating'], 0)
response.assertBodyContains({
  msg: 'No se pueden insertar calificaciones negativas.',
})
```

```
public async rate ({ request, response }) {        You, 7 hours ago • Nuevos
  const rate : number = parseInt(request.input('rate', null))
  if (rate !== null) {
    const post = await Post.find(request.param('id'))
    if (post !== null) {
      if (rate < 0) {
        response.status(406)
        response.send({
          msg: 'No se pueden insertar calificaciones negativas.',
          rating: rate,
        })
      } else {
        post.rating = rate
        post.save()
        response.accepted({
          msg: 'Se ha cambiado la calificación',
          rating: rate,
        })
      }
    } else {
      response.notFound(
        {
          msg: 'No existe ninguna publicación con la clave especificada.',
        }
      )
    }
  } else {
    response.forbidden(
      {
        msg: 'Debes de incluir una calificación adecuada.',
      }
    )
```

Nombre	Crear un tópico si no existe y agregarlo
ID	Sub1-U14
Descripción de la prueba	Cuando asignes un tópico a una publicación se debe verificar si ya existe para valorar si crearlo y añadirlo a la base de datos o simplemente agregarlo a la publicación.
Caso de Uso	Clasificar una publicación.
Descripción del caso de uso	Los usuarios pueden clasificar sus publicaciones con múltiples tópicos que ellos pueden crear o adjuntar.
Datos de entrada	‘topic’: 'Drama',
Resultado esperado	'status': 201, ‘msg’: 'Se creó un nuevo tópico y se agregó al post.',
Resultado obtenido	'status': 201, ‘msg’: 'Se creó un nuevo tópico y se agregó al post.',
Resultado final	PASSED

```
test('Try to add a topic that is not in the database and add it to DB.', async ({client}) => {
  const response = await client.put('/posts/1/topic').form({
    topic: 'Drama',
  })

  response.assertStatus(201)
  response.assertBody({
    msg: 'Se creó un nuevo tópico y se agregó al post.',
  })
}),
```

```
public async topic ({ request, response }) {
  let topic = request.input('topic', null)
  let found
  if (topic !== null) {
    const post = await Post.query().where('id', request.param('id'))
    if (post !== null) {
      found = await Topic.query().where('title', topic)
      if (found.length === 0) {
        //
        const newTopic = await Topic.create({
          title: request.input('topic'),
        })
        newTopic.save()
        //
        const postsTopics = await PostsTopics.create({
          post_id: parseInt(request.param('id')),
          topic_id: newTopic.id,
        })
        postsTopics.save()
        response.created(
          {
            msg: 'Se creó un nuevo tópico y se agregó al post.',
          }
        )
      } else {
        //
        const postsTopics = await PostsTopics.create({
          post_id: request.param('id'),
          topic_id: found[0].id,
        })
```

Nombre	Editar un comentario con autentificación
ID	Sub2-U15
Descripción de la prueba	Cuando se edita un comentario se requiere revisar si el propietario de este comentario es la persona que lo publicó.
Caso de Uso	Editar un comentario.
Descripción del caso de uso	Los usuarios pueden editar los comentarios que ellos mismos publicaron en alguno de las publicaciones de los usuarios.
Datos de entrada	‘email’: 'yeojin@gmail.com', ‘password’: '1234567890', ‘content': 'Let\'s sing a song! 🫖',
Resultado esperado	'status': 201, ‘msg’: 'El comentario se ha actualizado.',
Resultado obtenido	'status': 403, ‘msg’: 'Tu cuenta no existe en el sistema.',
Resultado final	FAILED

```
test('Edit a comment without being yours.', async ({client}) =>
  const response = await client.put('/comments/1').form({
    email: 'yeojin@gmail.com',
    password: '1234567890',
    content: 'Let\'s sing a song! 🫖',
  })

  response.assertStatus(200)
  response.assertBody({
    msg: 'El comentario se ha actualizado.',
  })
}),
```

```
× Edit a comment without being yours.

  Assertion Error: expected 403 to equal 200

- Expected  - 1
+ Received  + 1

- 200
+ 403
```

```
public async update ({ request, response }) {
  const validation = await request.validate(CommentValidator)
  if (request.param('id', null) !== null) {
    const comment = await Comment.query().where('id', request.param('id'))
    if (comment !== null) {
      comment[0].merge(validation).save()
      let email = request.input('email', null)
      const userComment = await User.find(comment[0].id)
      if (email !== null && userComment !== null) {
        const user = await User.query().where('email', email)
        if (user === null || user.length === 0) {
          response.forbidden({
            msg: 'Tu cuenta no existe en el sistema.',
          })
        } else if (user[0].id === userComment.id && user[0].password === userComment.password) {
          response.accepted({
            msg: 'El comentario se ha actualizado.',
          })
        } else {
          response.unauthorized({
            msg: 'No tienes el permiso para editar.',
          })
        }
      } else {
        response.notFound({
          msg: 'Debes iniciar sesión para editar.',
        })
      }
```

Nombre	Crear un grupo de usuarios y revisión de sus datos
ID	Sub3-U16
Descripción de la prueba	Se creará un grupo de usuarios, dando sus especificaciones y revisando su capacidad inicial para mantener usuarios.
Caso de Uso	Crear grupos de usuario
Descripción del caso de uso	Los usuarios podrán crear grupos para juntar a los usuarios que ellos quieran en múltiples formaciones para identificarse.
Datos de entrada	'title': 'Group for Discord 🎮', 'description': 'To play games!',
Resultado esperado	'status': 201, 'data': {} && isNotEmpty, 'empty': isBoolean && isTrue, 'msg': 'El grupo fue creado.'
Resultado obtenido	'status': 201, 'data': {} && isNotEmpty, 'empty': isBoolean && isTrue, 'msg': 'El grupo fue creado.'
Resultado final	PASSED

```
test('Create a group for users.', async ({client, assert}) => {
  const response = await client.post('/groups').form({
    title: 'Group for Discord 🎮',
    description: 'To play games!',
  })

  response.assertStatus(201)
  assert.isNotEmpty(response.body()['data'])
  assert.isBoolean(response.body()['empty'])
  assert.isTrue(response.body()['empty'])
  response.assertBodyContains({
    msg: 'El grupo fue creado.',
    data: {},
  })
}),
```

```
public async store ({ request, response }) {
  await request.validate(GroupValidator)
  const group = await Group.create({
    title: request.input('title', null),
    description: request.input('description', null),
  })
  group.save()
  response.created({
    msg: 'El grupo fue creado.',
    data: group,
    empty: true,
  })
}
```

Nombre	Encontrar una publicación con igual apreciación
ID	Sub4-U17
Descripción de la prueba	Se buscará una publicación que tenga la misma cantidad de likes como de dislikes.
Caso de Uso	Otorgar likes y dislikes
Descripción del caso de uso	A favor de la retroalimentación del usuario, no solo una publicación se puede calificar, sino, cada persona puede dar un me gusta o no me gusta como una medida rápida de apreciación.
Datos de entrada	
Resultado esperado	'status': 202, 'likes': equal('dislikes'), 'dislikes': equal('likes'),
Resultado obtenido	'status': 202, 'likes': equal('dislikes'), 'dislikes': equal('likes'),
Resultado final	PASSED

```
test('Find a post with the same likes and dislikes.', async ({client, assert}) => {
  const response = await client.get('/posts')

  response.assertStatus(202)
  response.body()['data'].forEach(element => {
    if (element.likes === element.dislikes) {
      assert.equal(element.likes, element.dislikes)
    }
  })
}),
```

```
public async index ({ request, response }) {
  let post
  if (request.param('id', null) !== null) {
    post = await Post.query().where('id', request.param('id'))
      .where('active', true).preload('comments').preload('user')
    if (post.length > 0) {
      response.accepted(
        {
          msg: 'Este es la publicación que buscas.',
          data: post,
        })
    } else {
      response.notFound(
        {
          msg: 'No se encontró la publicaión.',
        }
      )
    }
  } else {
    post = await Post.query().where('active', true).preload('comments')
    if (post.length > 0) {
      response.accepted(
        {
          msg: 'Estos son todos las publicaciones.',
          data: post,
        }
      )
    } else {
      response.notFound(
        {
          msg: 'Aún no hay ninguna publicación.',
```

Nombre	Eliminar todos los usuarios del sistema con excepciones
ID	Sub5-U18
Descripción de la prueba	Se buscará eliminar o desactivar todos los usuarios contando con un arreglo de excepciones.
Caso de Uso	Eliminar usuarios
Descripción del caso de uso	Se tiene la posibilidad de desactivar la cuentas de los usuarios que hayan introducido, dejándolas, de cierto modo, como eliminadas.
Datos de entrada	exceptions: ['yeojin@grrrverse.com'],
Resultado esperado	'status': 202, 'exception': isNotEmpty 'msg': 'Todos los usuarios han sido desactivados.',
Resultado obtenido	'status': 202, 'exception': isNotEmpty 'msg': 'Todos los usuarios han sido desactivados.',
Resultado final	PASSED

```
test('Find a post with the same likes and dislikes.', async ({client, assert}) => {
  const response = await client.get('/posts')

  response.assertStatus(202)
  response.body()['data'].forEach(element => {
    if (element.likes === element.dislikes) {
      assert.equal(element.likes, element.dislikes)
    }
  })
}),
test('Delete all users in system.', async ({client, assert}) => {
  const response = await client.delete('/users').form({
    exceptions: ['yeojin@grrrverse.com'],
  })

  response.assertStatus(202)
  assert.isNotEmpty(response.body()['exception'])
  response.assertBodyContains({
    msg: 'Todos los usuarios han sido desactivados.',
  })
})
```

```
  } else {
    let exception = request.input('exceptions', null)
    let exceptions : User[] = []
    let users = await User.query().where('active', true)
    if (users.length > 0) {
      await users.forEach((user) => {
        if (exception.includes(user.email)) {
          exceptions.push(user)
        } else {
          user.active = !user.active
        }
      })
      response.accepted(
        {
          msg: 'Todos los usuarios han sido desactivados.',
          exception: exceptions,
        }
      )
    } else {
      response.notFound(
        {
          msg: 'No hay ningún usuario activo.',
        }
      )
    }
  }
```

3. Seguimiento de Errores

En esta sección revisaremos todos los errores que se cometieron y fueron reportados al programa de seguimiento de errores bajo los parámetros establecidos.

ID	Descripción	Registro	Actualización	Responsable	Módulo	Estado	Gravedad
0001	Acceso denegado a la BD	2023-11-29	2023-11-29	Darío Ramos	Base de Datos	Resuelta	Alta
0002	Conexión negada por el suite	2023-11-29	2023-11-29	Darío Ramos	Conexión	Asignada	Mayor
0003	Verbo incorrecto	2023-11-29	10:48 PM	Darío Ramos	Prueba	Resuelta	Media
0004	Falta de parámetros requeridos	2023-11-29	11:28 PM	Darío Ramos	Prueba	Resuelta	Media
0005	Devolución de estado incorrecta	2023-11-29	11:33 PM	Darío Ramos	Prueba	Resuelta	Menor
0006	No se manda el parámetro de calificación	2023-11-29	11:34 PM	Darío Ramos	Prueba	Resuelta	Media
0007	Falta de propiedad activa en Posts	2023-11-29	11:46 PM	Darío Ramos	Controlador	Resuelta	Media
0008	Devolución de estado incorrecta	2023-11-29	11:44 PM	Darío Ramos	Prueba	Resuelta	Media
0009	Error en nombramiento de parámetros	2023-11-29	11:46 PM	Darío Ramos	Prueba	Resuelta	Menor
0010	Falta de propiedad calificación en Post	2023-11-29	11:58 PM	Darío Ramos	Modelo	Resuelta	Media
0011	Validación incorrecta de número a cadena.	2023-11-30	12:15 AM	Darío Ramos	Controlador	Resuelta	Media
0012	Falta de guardado de relación tópicos a publicaciones	2023-11-30	1:06 AM	Darío Ramos	Controlador	Resuelta	Mayor
0013	Validación incorrecta de número a cadena.	2023-11-30	1:43 AM	Darío Ramos	Controlador	Resuelta	Media
0014	Falta de parámetros de tiempo en tabla	2023-11-30	1:45 AM	Darío Ramos	Base de Datos	Resuelta	Menor
0015	Obtención de valor indefinido buscando nulo	2023-11-30	1:47 AM	Darío Ramos	Controlador	Resuelta	Media

0016	Falta de aseguramiento de parámetro en el cuerpo	2023-11-30	1:48 AM	Darío Ramos	Prueba	Resuelta	Media
0017	Sin identificación para encontrar modelo.	2023-11-30	1:56 AM	Darío Ramos	Controlador	Resuelta	Mayor
0018	Falta de parámetro en prueba.	2023-11-30	1:58 AM	Darío Ramos	Prueba	Resuelta	Media
0019	Falta de adición de nulabilidad y parámetro opcional	2023-11-30	2:00 AM	Darío Ramos	Validación	Resuelta	Media
0020	Error en nombramiento de ruta	2023-11-30	2:10 AM	Darío Ramos	Ruteador	Resuelta	Mayor
0021	Devolución incorrecta de código de estado	2023-11-30	2:11 AM	Darío Ramos	Prueba	Resuelta	Menor
0022	Falta de aseguramiento de respuesta	2023-11-30	2:13 AM	Darío Ramos	Prueba	Resuelta	Media
0023	Falla en la precarga de relaciones en modelo	2023-11-30	2:29 AM	Darío Ramos	Controlador	Resuelta	Media
0024	Falla en repetición de condición de aseguramiento	2023-11-30	2:32 AM	Darío Ramos	Prueba	Resuelta	Media
0025	Falta de declaración de matriz o arreglo de modelo	2023-11-30	3:08 AM	Darío Ramos	Controlador	Resuelta	Media
0026	Falla al agregar valores a arreglo.	2023-11-30	3:10 AM	Darío Ramos	Controlador	Resuelta	Media

4. Versionamiento

Veamos las versiones de nuestro sistema, recordemos que este es un sistema de prueba y no representa el final de ningún producto.

Versión	Fecha	Importancia	Descripción	Porcentaje
0.0.1	Oct 17 de 2023	Micro	Se crea el proyecto.	5%
0.0.2	Oct 17 de 2023	Menor	Se añaden relaciones entre modelos.	8%
0.1.0	Oct 18 de 2023	Micro	Se añaden seeders y validaciones.	10%
0.2.0	Oct 19 de 2023	Menor	Se diseñan arreglos para las relaciones.	12%
0.2.0	Oct 20 de 2023	Micro	Se hacen optimizaciones.	20%
0.2.1	Nov 9 de 2023	Mayor	Se fabrican los primeros testeos.	40%
0.2.2	Nov 15 de 2023	Menor	Se hacen más pruebas con suites.	50%
0.3.0	Nov 27 de 2023	Mayor	Se crean cambios en el CRUD de la aplicación.	60%
0.4.0	Nov 29 de 2023	Menor	Se genera nueva suite de trabajo	70%
0.4.1	Nov 30 de 2023	Menor	Se termina la primera agrupación final de pruebas.	80%
1.0.0	Nov 30 de 2023	Mayor	Se termina el proyecto examen de evaluación.	100%