

Práctica 2

Análisis y Diseño de Sistemas 1

Integrantes

Carné	Nombre
201602625	Oscar Alfredo Llamas Lemus
201504051	Asunción Mariana Sic Sor
201601469	Oscar Eduardo Mazariegos López
201612383	Javier Antonio Álvarez González
201408549	Elba María Álvarez Domínguez

- [Enlace a repositorio en GitHub](#)

Tabla de Contenido

- [Práctica 2](#)
 - [Tabla de Contenido](#)
 - [Conceptos Básicos](#)
 - [Pruebas Unitarias](#)
 - [Estructura de las Pruebas Unitarias](#)
 - [Herramientas Utilizadas](#)
 - [Mocha](#)
 - [Chai](#)
 - [Chai-Http](#)
 - [Testing-Library / React](#)
 - [Pruebas Unitarias Realizadas](#)
 - [Backend](#)
 - [Frontend](#)
 - [Pruebas realizadas en Jest con react, estas pruebas fueron creadas con el objetivo de observar el correcto funcionamiento de la pagina.](#)
 - [Anexos](#)
 - [Capturas de pantalla de la aplicación.](#)
 - [Pantalla login](#)
 - [Pantalla perfil de usuario y datos.](#)
 - [Modal para crear una publicación](#)
 - [Visualizar publicaciones](#)
 - [Pantalla registro](#)
 - [Referencia Bibliográfica](#)

Conceptos Básicos

Pruebas Unitarias

Las pruebas unitarias (también test unitarios, o unit testing) son un método de pruebas de software que se realizan escribiendo fragmentos de código que testeará unidades de código fuente. El objetivo es asegurar que cada unidad funciona como debería de forma independiente.

No existe una definición cerrada respecto a lo que es una unidad de código, podría ser una clase, o podría ser simplemente un método.

Las pruebas unitarias son las primeras que se realizan ya con código escrito, y las crea el propio desarrollador.

Es importante también tener claro, que las pruebas unitarias, como el resto de pruebas, no demuestran la ausencia de errores en el código, ya que éstos pueden ser errores de integración, rendimiento, etc... que los test unitarios no son capaces de detectar.

Estructura de las Pruebas Unitarias

Por norma general, los unit test deberían seguir una estructura AAA, sobretodo para facilitar su lectura y entendimiento. AAA son las siglas de:

- **Arrange (Organizar):** En esta parte de la prueba, debes establecer las condiciones iniciales para poder realizarla, así como el resultado que esperas obtener. Y esto significa por ejemplo, declarar las variables y crear las instancias de los objetos.
- **Act (Accionar):** Es la parte de ejecución, tanto del fragmento de código de la prueba, como del fragmento de código a testear.
- **Assert (Comprobar):** Por último, se realiza la comprobación para verificar que el resultado obtenido, coincide con el esperado.

Herramientas Utilizadas

Mocha

Mocha es un framework de pruebas de JavaScript con numerosas funciones que se ejecuta en Node.js y en el navegador. Las pruebas de Mocha se ejecutan en serie, lo que permite informes flexibles y precisos. Como hemos dicho, Mocha es un framework de pruebas.

Chai

Chai es una librería de aserciones, que se puede emparejar con cualquier marco de pruebas de JavaScript. Chai tiene varias interfaces: `assert`, `expect` y `should`. Que permiten al programador elegir un estilo que le resulte más legible a la hora de desarrollar los test.

Chai-Http

Chai HTTP es una extensión de la librería CHAI, que permite realizar pruebas de integración con llamadas HTTP utilizando las aserciones de CHAI y todos los métodos de HTTP: GET, POST, PUT, DELETE, PATCH.

Testing-Library / React

Testing Library es un set de utilidades de testing simple y completo que promueve las buenas prácticas del testing. Es una ayuda para testear User Interfaces de forma centrada en el usuario.

Pruebas Unitarias Realizadas

Backend

Método / Función	Prueba Unitaria
<pre>router.post('/login', async (req, res) => { try {</pre>	<pre>describe('test login endpoint: ,()=>{ it('should login', (done) => { chai.request(url)</pre>

```

    const data = req.body;
    await Usuario.findOne({ username:
data.username, password: data.password },
function (err, docs) {
    if (err){
        console.log(err)
        res.status(404);
        res.send({ message : error
});
    } else if (docs == null) {
        res.status(404);
        res.send({ message :
"credenciales incorrectas o usuario no
existe" });
    } else{
        res.status(202);
        console.log("credenciales
correctas :3")
        res.json(docs);
    }
});

} catch (error) {
    console.log(error)
    res.status(404);
    res.send({ message : error });
    console.log("credenciales
incorrectas o usuario no existe :c");
}

});

```

```

        .post('/login')
        .send({username:
"201602625", password:
"123456789"})
        .end( function(err,res){
            console.log(res.body)

expect(res).to.have.status(202);
done();

        });
    });
});

```

```

router.post("/new", async (req, res) => {

    try {

        const data = req.body;
        await Usuario.findOne({ username:
data.username}, async function (err, docs) {
            if (err){
                console.log(err)
                res.status(404);
                res.send({ message : err });
            } else if (docs == null) {

                var result = "";

                if (data.image.toString() !=
"" ){

                    var nombrei = "fotos/" +

```

```

describe('test register
endpoint: ',()=>{
    it('should register new user',
(done) => {
        chai.request(url)
        .post('/new')
        .send({
            nombre: "Test",
            apellido: "User",
            username:
"000000003cl",
            password: "123456789",
            image: ""
        })
        .end( function(err,res){
            console.log(res.body)

expect(res).to.have.status(202)
done();

```

```

    uuid() + ".jpg";

    //se convierte la base64 a
    bytes

    let buff = new
    Buffer.from(data.image, 'base64');

    const params = {
      Bucket: "practica2-ayd1",
      Key: nombrei,
      Body: buff,
      ContentType: "image",
      ACL: 'public-read'
    };
    s3.putObject(params).promise();

    result = `https://practica2-
    ayd1.s3.us-east-2.amazonaws.com/` + nombrei;

  }

  await Usuario.create({
    nombre: data.nombre,
    apellido: data.apellido,
    username: data.username,
    password: data.password,
    image: result.toString()
  });
  res.status(202);
  res.json({ message : 'Usuario
registrado :')});
} else{
  res.status(404);
  res.json({ message : 'Usuario
ya existente :('});
}
});

} catch (error) {
  console.log(error)
  res.status(404);
  res.send({ message : error });
}
});

```

```

    });
  });
});

```

```

router.post("/nuevaPublicacion", async (req,
res) => {

  try {
    const data = req.body;

```

```

describe('test nueva publicacion
endpoint: ',()=>{
  it('should publicar', (done)
=> {
    chai.request(url)

```

```

var result = "";

if (data.image.toString() !== ""){

    var nombrei = "fotos/" + uuid() +
".jpg";

    //se convierte la base64 a bytes
    let buff = new
Buffer.from(data.image, 'base64');

    const params = {
        Bucket: "practica2-ayd1",
        Key: nombrei,
        Body: buff,
        ContentType: "image",
        ACL: 'public-read'
    };

    s3.putObject(params).promise();

    result = `https://practica2-
ayd1.s3.us-east-2.amazonaws.com/` + nombrei;

}

let date = new
Date().toISOString().split('T')[0];

await Publicacion.create({
    nombre: data.nombre,
    apellido: data.apellido,
    username: data.username,
    contenido: data.contenido,
    fecha: date.toString(),
    image: result.toString(),
});
res.status(202);
res.json({ message : 'Publicacion
registrada :')});

} catch (error) {
    console.log(error)
    res.status(404);
    res.send({ message : error });
}

});

```

```

router.get('/getPublicaciones', async (req,
res) => {

```

```

.post('/nuevaPublicacion')
.send({
    nombre: "Oscar",
    apellido: "Llamas",
    username: "201602625",
    contenido: "Hola
Mundo! Prueba unitaria",
    image: ""
})
.end( function(err,res){

console.log(res.body)

expect(res).to.have.status(202)
done();
});

});

describe('test segunda
publicacion endpoint: ',()=>{
    it('debería realizar una nueva
publicación', (done) => {
        chai.request(url)
        .post('/nuevaPublicacion')
        .send({
            nombre: "Oscar",
            apellido: "Llamas",
            username: "201602625",
            contenido: "Otra
prueba unitaria",
            image: ""
        })
        .end( function(err,res){

console.log(res.body)

expect(res).to.have.status(202)
done();
});
});
});

```

```

describe('get all post: ',()=>{
    it('should get all post',

```

```

try {

    await Publicacion.find({}, function
(err, publicaciones) {

        if (err){
            console.log(err)
            res.status(404);
            res.send({ message : err });
            console.log("Error al obtener
publicaciones :c");
        } else{
            res.status(202);
            console.log("Publicaciones
obtenidas correctamente :D")
            res.json(publicaciones);
        }

    });

} catch (error) {
    console.log(error)
    res.status(404);
    res.send({ message : error });
    console.log("Error al obtener
publicaciones :c");
}

});

```

```

(done) => {
    chai.request(url)
        .get('/getPublicaciones')
        .end( function(err,res){
            console.log(res.body)

            expect(res).to.have.status(202);
            done();
        });
});

```

Frontend

Pruebas realizadas en Jest con react, estas pruebas fueron creadas con el objetivo de observar el correcto funcionamiento de la pagina.

```

describe('FormularioLogin', ()=>{
    it('Debe de existir el texto para el campo contraseña', () =>{
        render(<FormLogin/>)
        expect(screen.queryByText(/contraseña/i)).toBeInTheDocument()
    })

    it('Debe tener boton de iniciar sesion', () =>{
        render(<FormLogin/>)
        expect(
            screen.getByRole(
                'button',
                {
                    name: /iniciar sesión/i
                }
            )
        )
    })

```

```

        ).toBeInTheDocument();
    })

    it('Debe mostart imagen de perfil default', () =>{
        let a = render(<FormLogin/>)
        // /expresion regular/ i -> ignore case
        expect(screen.queryByText(/userDefault/i)).toBeInTheDocument()
    })
});

```

```

describe('Que esten todos los campos necesarios', ()=>{
    it('Debe tener datos personales', () =>{
        expect(true);
        render(<FormularioRegistro/>);

        expect(screen.queryByText(/usuario/i)).toBeInTheDocument();
        expect(screen.queryByText(/nombre/i)).toBeInTheDocument();
        expect(screen.queryByText(/apellido/i)).toBeInTheDocument();
    })

    it('Debe tener contrasena y confirmacion contrasena', () =>{
        render(<FormularioRegistro/>)

        expect(screen.queryByText(/confirmar contraseña/i)).toBeInTheDocument()
    })

    it('Debe tener boton de Registrarse', () =>{
        render(<FormularioRegistro/>)

        expect(screen.getByRole('button', {name:
/Registrarse/i})).toBeInTheDocument();
    })
});

```

```

describe('Validacion de campos', ()=>{
    it('Que rechace campos no validos', () =>{

        let dummyFormulario = new FormularioRegistro();
        dummyFormulario.state = {
            userName: '',
            name: '',
            lastName: '',
            contra: '',
            contra2: '',
            foto: "",
            usuarios: []
        };

        let result = dummyFormulario.ComprobacionYMensaje();
    });

```

```

    expect(!result).toBeTruthy();
  });

  it('Que acepte campos validos', () =>{
    let dummyFormulario = new FormularioRegistro();
    dummyFormulario.state = {
      userName: 'valido',
      name: 'valido',
      lastName: 'valido',
      contra: 'contraValida',
      contra2: 'contraValida',
      // chapuz:
      foto: "any",
      usuarios: []
    }

    let result = dummyFormulario.ComprobacionYMensaje();
    expect(result).toBeTruthy();
  });
});

```

```

describe('Debe aparecer el titulo principal en la pagina', ()=>{
  it('Debe tener el titulo analisis', () =>{
    expect(true);
    render(<Menu/>);

    expect(screen.queryByText(/analisis/i)).toBeInTheDocument();
  })
  it('Debe tener el titulo diseño', () =>{
    expect(true);
    render(<Menu/>);

    expect(screen.queryByText(/diseño/i)).toBeInTheDocument();
  })
});

```

```

describe('Validacion de campos', ()=>{
  it('Las cookies tengan datos.', () =>{

    let perfil = new Profile();
    perfil.state = {
      username: '',
      nombre: '',
      apellido: '',
      publi: '',
    };
  });

  it('Existir boton para crear publicacion', () =>{
    expect(true);
  });
});

```

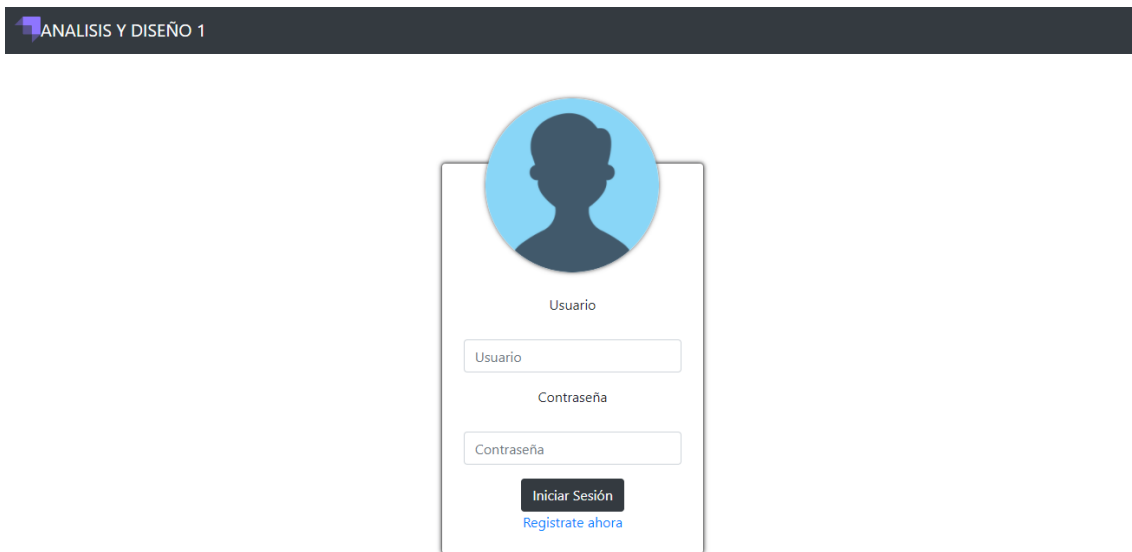


```
render(<Profile/>);
expect(screen.queryByText(/Crear Publicación/i)).toBeInTheDocument();
});
```

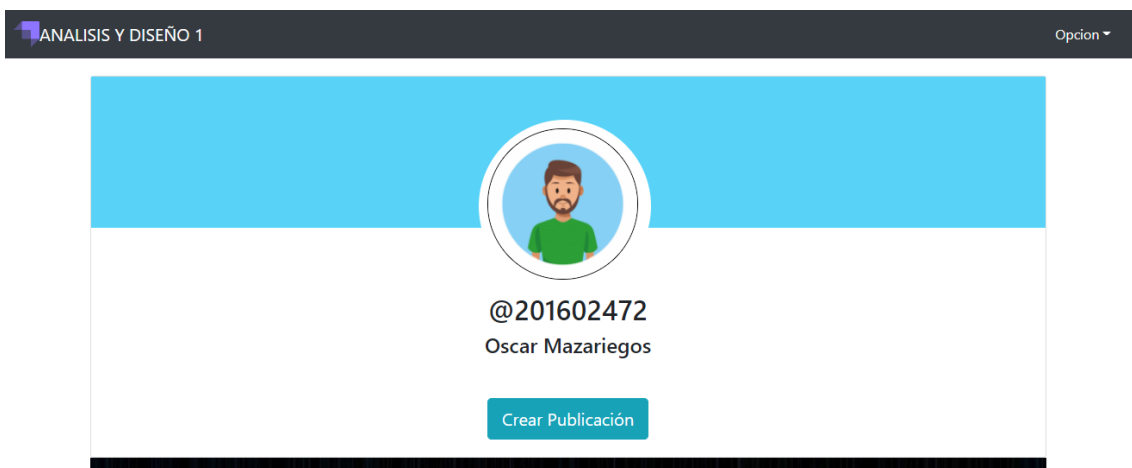
Anexos

Capturas de pantalla de la aplicación.

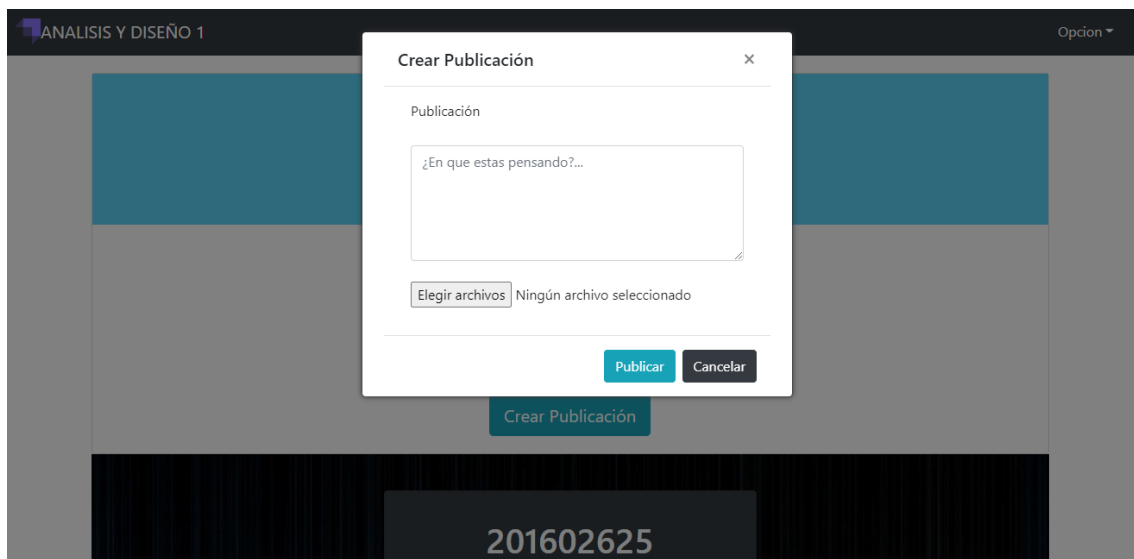
Pantalla login



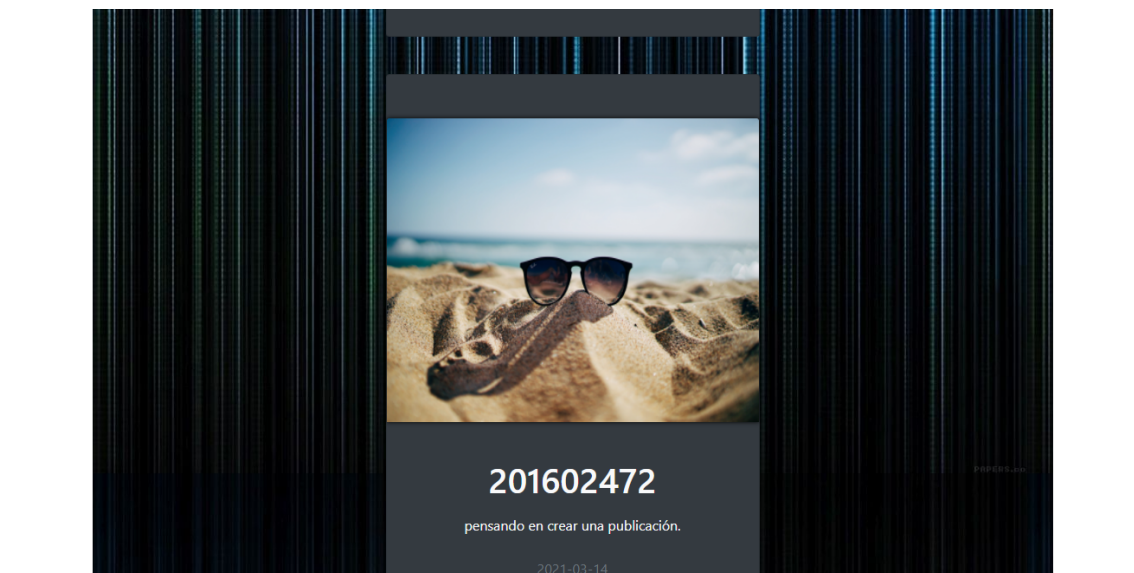
Pantalla perfil de usuario y datos.



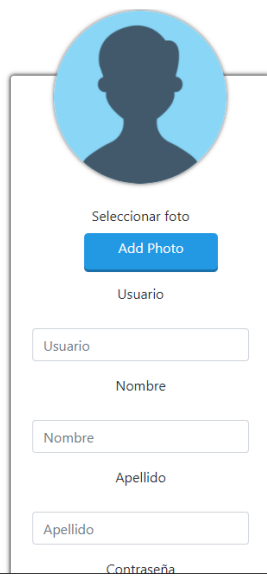
Modal para crear una publicación



Visualizar publicaciones



Pantalla registro



A user profile form with a circular placeholder for a profile picture at the top. Below the photo is a label 'Seleccionar foto' and a blue button labeled 'Add Photo'. Underneath is a label 'Usuario' followed by a text input field labeled 'Usuario'. Below that is a label 'Nombre' followed by a text input field labeled 'Nombre'. Then another label 'Apellido' followed by a text input field labeled 'Apellido'. At the bottom, there is a label 'Contraseña' followed by a text input field.

Referencia Bibliográfica

- Moreno, O. (2019). Pruebas unitarias: imprescindibles para programar. Marzo 13, 2021, de Oscar Moreno Sitio web: <http://oscarmoreno.com/pruebas-unitarias/>
- Desconocido. (2018). Testeando en Node.js usando Mocha y Chai. Marzo 13, 2021, de Syntonize Sitio web: <https://www.syntonize.com/node-js-usando-mocha-y-chai/>
- Cordero, N. (2018). Testeo de API REST con Mocha y Chai-HTTP. Marzo 13, 2021, de Paradigma Sitio web: <https://www.paradigmadigital.com/dev/testeo-api-rest-mocha-chai-http/>
- Bernalte, L. (2021). Por qué usar Testing Library en lugar de Enzyme . Marzo 13, 2021, de Dev Sitio web: <https://dev.to/lucasbernalte/por-que-usar-testing-library-en-lugar-de-enzyme-2c2>