# DESARROLLO DE APLICACIONES WEB AVANZADO

## LABORATORIO N° 09

## Spinners y Subpáginas

| | | | |
|---|---|---|---|
| ***Alumno(s):*** | | ***Nota*** | |
| ***Grupo:*** | | ***Ciclo:V*** | |

| *Criterio de Evaluación* | *Excelente (4pts)* | *Bueno (3pts)* | *Requiere mejora (2pts)* | *No acept. (0pts)* | *Puntaje Logrado* |
|---|---|---|---|---|---|
| Realiza un deploy con Now.sh | | | | | |
| Utiliza Bootstrap para interfaz de chat | | | | | |
| Desarrolla aplicaciones web con socket | | | | | |
| Realiza con éxito lo propuesto en la tarea. | | | | | |
| Es puntual y redacta el informe adecuadamente | | | | | |

# Laboratorio 09:
# Spinners y Subpáginas

## Objetivos:

Al finalizar el laboratorio el estudiante será capaz de:

- Entender el funcionamiento de los Spinners CSS
- Desarrollar aplicaciones web con el framework Bootstrap
- Implementación correcta de React Router

## Seguridad:

- Ubicar maletines y/o mochilas en el gabinete del aula de Laboratorio.
- No ingresar con líquidos, ni comida al aula de Laboratorio.
- Al culminar la sesión de laboratorio apagar correctamente la computadora y la pantalla, y ordenar las sillas utilizadas.

## Equipos y Materiales:

- Una computadora con:
  - Windows 7 o superior
  - VMware Workstation 10+ o VMware Player 7+
  - Conexión a la red del laboratorio

- Máquinas virtuales:
  - Windows 7 Pro 64bits Español - Plantilla

- Instalador de node.js

## Procedimiento:

### Lab Setup

1. **Configuración inicial del proyecto**
   1.1. Copie el contenido de la carpeta **lab08** (el laboratorio anterior) a excepción de la carpeta **node_modules** dentro de una carpeta llamada **lab09**
   1.2. Instalaremos una nueva librería en nuestra carpeta dawa_api:

```
>npm install --save morgan
```

   1.3. Luego modificaremos el archivo server.js para incluir esa librería.

```
const bodyParser = require('body-parser');
const logger = require('morgan');

app.use(logger(':method :remote-addr :url :response-time'));
app.use(bodyParser.urlencoded({ extended: true }));
app.use(bodyParser.json());
```

   Morgan es un middleware de registros. Permite ver que IP, qué método y tiempo de respuesta empleo en una consulta.

2. **Creación de página de perfil**
   2.1. Agregaremos en nuestro App.js (de lab09) las siguientes líneas, que harán uso de la url /profile para todo lo desarrollado en este laboratorio.

```
import Welcome from './views/Welcome';
import Chat from './views/Chat/Chat';
import Profile from './views/Profile/Profile';
import NotFound from './views/NotFound';

class App extends Component {
    render() {
        return (
            <BrowserRouter>
                <Layout>
                    <Switch>
                        <Route path="/" exact component={Home} />
                        <Route path="/details" component={Details} />
                        <Route path="/login" component={Login} />
                        <Route path="/welcome" component={Welcome} />
                        <Route path="/chat" component={Chat} />
                        <Route path="/profile" component={Profile} />
                        <Route
                            path="/logout"
                            render={() => {
```

   2.2. Ahora crearemos el archivo Profile.js dentro de la carpeta Profile que será creada dentro de views.

dawa ▸ lab09 ▸ src ▸ views ▸ Profile ▸ Profile.js ▸ ...

```
import React from 'react';
import { Switch, Route } from 'react-router-dom';
import { Helmet } from 'react-helmet';

import ProfileDetails from './ProfileDetails/ProfileDetails';
import ProfileEdit from './ProfileEdit/ProfileEdit';

const Profile = () => {
    return (
        <div className="container">
            <Helmet>
                <title>Perfil del usuario</title>
            </Helmet>
            <Switch>
                <Route path="/profile" exact component={ProfileDetails} />
                <Route path="/profile/edit" component={ProfileEdit} />
            </Switch>
        </div>
    );
};

export default Profile;
```

Este archivo tendrá acceso a dos rutas, la que muestra el detalle del perfil, y la que permite editar el perfil del usuario.

2.3. Para poder acceder a esta URL, modificaremos el archivo Header.js

```
import { Link, NavLink } from 'react-router-dom';
import { Nav, Navbar, NavDropdown } from 'react-bootstrap';

import imgLogo from '../../assets/img/logo-tecsup.png';

const Header = props => {
    return (
        <Navbar bg="dark" expand="lg" variant="dark" id="header">
            <div className="container">
                <Link className="navbar-brand" to="/">
                    <img src={imgLogo} alt="Tecsup Logo" />
                    Tecsup App
                </Link>
                <Navbar.Toggle aria-controls="basic-navbar-nav" />
                <Navbar.Collapse id="basic-navbar-nav" className="justify-content-end">
                    <Nav>
                        <NavLink to="/" exact className="nav-link">
                            Inicio
                        </NavLink>
                        <NavLink to="/login" className="nav-link">
                            Iniciar sesión
                        </NavLink>
                        <NavDropdown title="Usuario" id="basic-nav-dropdown">
                            <NavLink to="/profile" className="dropdown-item">
                                Mi Perfil
                            </NavLink>
                            <NavLink to="/chat" className="dropdown-item">
                                Chat
                            </NavLink>
                            <NavDropdown.Divider />
                            <NavLink to="/logout" className="dropdown-item">
                                Cerrar sesión
                            </NavLink>
                        </NavDropdown>
                    </Nav>
```

2.4. La vista que deseamos obtener tendrá la siguiente forma:


2.5. Crearemos el archivo ProfileDetails.js dentro de la carpeta ProfileDetails dentro de la carpeta Profile.

dawa ▸ lab09 ▸ src ▸ views ▸ Profile ▸ ProfileDetails ▸ ProfileDetails.js ▸ ...

```
import React, { Component } from 'react';
import { Tab, Nav } from 'react-bootstrap';
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';
import {
    faUser,
    faEdit,
    faKey,
    faCameraRetro
} from '@fortawesome/free-solid-svg-icons';

import getAvatar from '../../../utils/avatar';

import './ProfileDetails.css';

class ProfileDetails extends Component {
    state = {
        userName: '',
        email: ''
    };
    componentDidMount() {
        this.setState({
            userName: localStorage.getItem('userName'),
            email: localStorage.getItem('email')
        });
    }
    passwordHandler = e => {
        alert('Funcion por implementar!');
    };
    pictureHandler = e => {
        alert('Funcion por implementar!');
    };
```

2.6. Primero hemos obtenido la información del usuario a través del localStorage. Luego hemos definido funciones a implementar a futuro, que son el manejador de la contraseña y el de la foto de perfil.

```
render() {
    const urlLorem = 'https://www.lipsum.com/feed/html';
    return (
        <Tab.Container id="profile-tabs" defaultActiveKey="inicio">
            <div className="row pt-4">
                <div className="col-md-3">
                    <div className="profile-img">
                        <img
                            src={getAvatar(this.state.email)}
                            className="rounded-circle user_img"
                            alt=""
                        />
                    </div>
                </div>
                <div className="col-md-7">
                    <div className="profile-head">
                        <h5>
                            <FontAwesomeIcon icon={faUser} /> {this.state.userName}
                        </h5>
                        <h6>Web Developer y Diseñador</h6>
                        <p className="profile-rating">
                            RANKINGS : <span>8/10</span>
                        </p>
                        <Nav>
                            <Nav.Item>
                                <Nav.Link className="profile-tab" eventKey="inicio">
                                    Tab 1
                                </Nav.Link>
                            </Nav.Item>
                            <Nav.Item>
                                <Nav.Link className="profile-tab" eventKey="mensajes">
                                    Tab 2
                                </Nav.Link>
                            </Nav.Item>
                        </Nav>
                    </div>
                </div>
```

```
                <div className="col-md-2">
                    <button
                        className="btn btn-info btn-block"
                        onClick={() => this.props.history.push('/profile/edit')}
                    >
                        <FontAwesomeIcon icon={faEdit} /> Editar Perfil
                    </button>
                    <button
                        className="btn btn-primary btn-block"
                        onClick={this.pictureHandler}
                    >
                        <FontAwesomeIcon icon={faCameraRetro} /> Imagen
                    </button>
                    <button
                        className="btn btn-primary btn-block"
                        onClick={this.passwordHandler}
                    >
                        <FontAwesomeIcon icon={faKey} /> Contraseña
                    </button>
                </div>
            </div>
```

```
                <div className="row pb-4">
                    <div className="col-md-3">
                        <ul className="list-group">
                            <li className="list-group-item text-muted">
                                Actividad <i className="fa fa-dashboard fa-1x" />
                            </li>
                            <li className="list-group-item text-right">
                                <span className="pull-left">
                                    <strong>Shares</strong>
                                </span>{' '}
                                125
                            </li>
                            <li className="list-group-item text-right">
                                <span className="pull-left">
                                    <strong>Likes</strong>
                                </span>{' '}
                                13
                            </li>
                            <li className="list-group-item text-right">
                                <span className="pull-left">
                                    <strong>Posts</strong>
                                </span>{' '}
                                37
                            </li>
                        </ul>
                    </div>
```

En el siguiente bloque de código, vera un texto con Lorem Ipsum (texto de relleno). Así mismo, en la parte inicial del render de este ejercicio hay una URL que lleva directo a un generador online de este texto. Utilícelo para no tener que escribir todo.

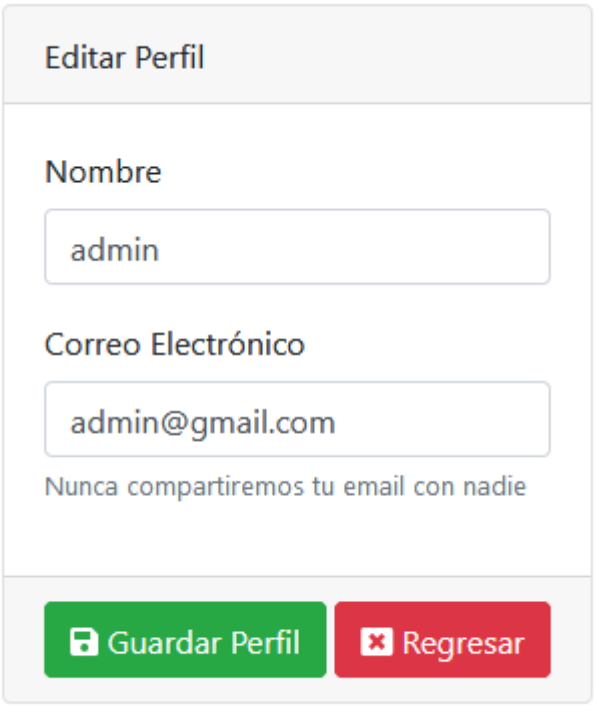
```
                    <div className="col-md-7">
                        <Tab.Content>
                            <Tab.Pane className="profile-pane" eventKey="inicio">
                                Para generar un texto Lorem Ipsum utilizaremos la URL:
                                <a href={urlLorem} target="_blank" rel="noopener noreferrer">
                                    {urlLorem}
                                </a>
                            </Tab.Pane>
                            <Tab.Pane className="profile-pane" eventKey="mensajes">
                                Lorem ipsum dolor sit amet, consectetur adipiscing elit.
                                Curabitur ac sapien lacus. In elementum urna sed sapien cursus
                                viverra. Vestibulum sed velit quis eros cursus lobortis. Aenean
                                pulvinar nibh lacinia auctor feugiat. Phasellus finibus mauris
                                sit amet elit pharetra feugiat vel eu eros. Sed rhoncus lacinia
                                lacus sed viverra. Pellentesque faucibus lectus et leo tempor,
                                id suscipit tellus lobortis. Sed id dolor ut diam venenatis
                                bibendum ut quis magna. Ut suscipit at magna quis tincidunt.
                                Vivamus sed ante eget eros finibus tristique eget a ligula.
                                Proin fermentum, erat id tristique suscipit, magna eros blandit
                                erat, ac faucibus sem lacus ac nisi. Quisque lacus urna,
                                sagittis eu augue et, mattis malesuada arcu. Fusce venenatis
                                porttitor justo, et vehicula velit facilisis ac. Vestibulum
                                dapibus suscipit velit. Vivamus sit amet neque placerat, ornare
                                sem sit amet, mattis tellus.
                            </Tab.Pane>
                        </Tab.Content>
                    </div>
                </div>
            </Tab.Container>
        );
    }
}

export default ProfileDetails;
```

2.7. Esto aún no debe funcionar del todo, ya que el email nos da error (o bota vacío en el mejor de los casos). Esto es debido a que no guardamos el email durante nuestro inicio de sesión. Modificaremos el archivo Login.js

```
14    submitHandler = e => {
15        e.preventDefault();
16        axios({
17            method: 'post',
18            url: 'user/signin',
19            data: {
20                username: this.state.usuario,
21                password: this.state.password
22            }
23        })
24            .then(response => {
25                localStorage.setItem('userId', response.data.data._id);
26                localStorage.setItem('userName', response.data.data.username);
27                localStorage.setItem('email', response.data.data.email);
28                localStorage.setItem('userToken', response.data.token);
29                this.props.history.push('/welcome');
30            })
31            .catch(error => {
32                console.log('hubo un error', error);
33            });
34    };
```

**3. Vista de edición de perfil**

3.1. La siguiente vista a implementar será la edición del perfil, que lucirá como este formulario:




3.2. Antes que nada, nuestro inicio de sesión y este formulario harán que nuestro pie de página se pegue demasiado. Le daremos una altura definida mínima en index.css

dawa ▸ lab09 ▸ src ▸ index.css ▸ ...

```css
#header img {
    width: 170px;
    margin-right: 15px;
}

#main {
    min-height: 80vh;
}

#main .carousel-inner img {
    max-height: 70vh;
    object-fit: cover;
    filter: grayscale(70%);
}

#footer {
    background: linear-gradient(90deg, #1c3643, #273b47 25%, #1e5372);
}
```

3.3. Crearemos el archivo ProfileEdit.js dentro de la carpeta ProfileEdit dentro de Profile

dawa ▸ lab09 ▸ src ▸ views ▸ Profile ▸ ProfileEdit ▸ ProfileEdit.js ▸ ...

```jsx
import React, { Component } from 'react';
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';
import { faSave, faWindowClose } from '@fortawesome/free-solid-svg-icons';

class ProfileEdit extends Component {
    state = {
        username: '',
        email: '',
        loading: false
    };
    render() {
        return (
            <div className="container d-flex justify-content-center mt-4 mb-4">
                <div className="card">
                    <div className="card-header">Editar Perfil</div>
                    <form onSubmit={this.submitHandler}>
                        <div className="card-body">
                            <div className="form-group">
                                <label>Nombre</label>
                                <input
                                    type="text"
                                    className="form-control"
                                    placeholder="Ingrese su nombre"
                                    value={this.state.username}
                                />
                            </div>
                            <div className="form-group">
                                <label>Correo Electrónico</label>
                                <input
                                    type="email"
                                    className="form-control"
                                    placeholder="Enter email"
                                    value={this.state.email}
                                />
                                <small className="form-text text-muted">
                                    Nunca compartiremos tu email con nadie
                                </small>
                            </div>
                        </div>
```

```
                        <div className="card-footer">
                            <button className="btn btn-success" type="submit">
                                <FontAwesomeIcon icon={faSave} /> Guardar Perfil
                            </button>{' '}
                            <button
                                className="btn btn-danger"
                                type="button"
                                onClick={this.goBackHandler}
                            >
                                <FontAwesomeIcon icon={faWindowClose} /> Regresar
                            </button>
                        </div>
                    </form>
                </div>
            </div>
        );
    }
}

export default ProfileEdit;
```

3.4. Por ahora solamente hemos creado el render, tenemos que manejar el estado de los inputs. Agregaremos las siguientes líneas en el render.

```
<div className="form-group">
    <label>Nombre</label>
    <input
        type="text"
        className="form-control"
        placeholder="Ingrese su nombre"
        value={this.state.username}
        onChange={e => this.inputHandler(e, 'username')}
    />
</div>
<div className="form-group">
    <label>Correo Electrónico</label>
    <input
        type="email"
        className="form-control"
        placeholder="Enter email"
        value={this.state.email}
        onChange={e => this.inputHandler(e, 'email')}
    />
    <small className="form-text text-muted">
        Nunca compartiremos tu email con nadie
    </small>
</div>
```

3.5. Ahora crearemos las funciones que modifican esos estados

```
class ProfileEdit extends Component {
    state = {
        username: '',
        email: '',
        loading: false
    };
    componentDidMount() {
        this.setState({
            username: localStorage.getItem('userName'),
            email: localStorage.getItem('email')
        });
    }
    inputHandler = (event, field) => {
        this.setState({ [field]: event.target.value });
    };
```

3.6. Finalmente, tenemos que enviar esta data al servidor. Al igual que con nuestro inicio de sesión, invocaremos a axios para poder hacer la llamada a nuestra API.

```
import React, { Component } from 'react';
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';
import { faSave, faWindowClose } from '@fortawesome/free-solid-svg-icons';

import axios from '../../../utils/axios';
import { PROFILE_EDIT } from '../../../utils/urls';
```

3.7. Cuando utilizamos una API, es una buena práctica agrupar todas las URLS que utilizaremos dentro de un solo archivo e invocarlas a través de una variable. Nosotros haremos esto por si en algún momento la URL varía, no necesitaremos modificar nuestro código de componente, solamente el archivo de URLs.

dawa ▸ lab09 ▸ src ▸ utils ▸ urls.js ▸ ...

```
export const PROFILE_EDIT = {
    method: 'put',
    url: 'user/' + localStorage.getItem('userId')
};
```

3.8. Crearemos entonces el submitHandler que enviará la data al servidor.

```
submitHandler = e => {
    e.preventDefault();
    const data = {
        username: this.state.username,
        email: this.state.email
    };
    this.setState({ loading: true });
    axios({
        ...PROFILE_EDIT,
        headers: {
            Authorization: localStorage.getItem('userToken')
        },
        data: data
    })
        .then(response => {
            console.log(response);
            this.setState({ loading: false });
            this.props.history.push('/profile');
            alert('Usuario editado con éxito!');
        })
        .catch(error => {
            console.error(error.response);
            this.setState({ loading: false });
            alert(error.response.data.message);
        });
};
```

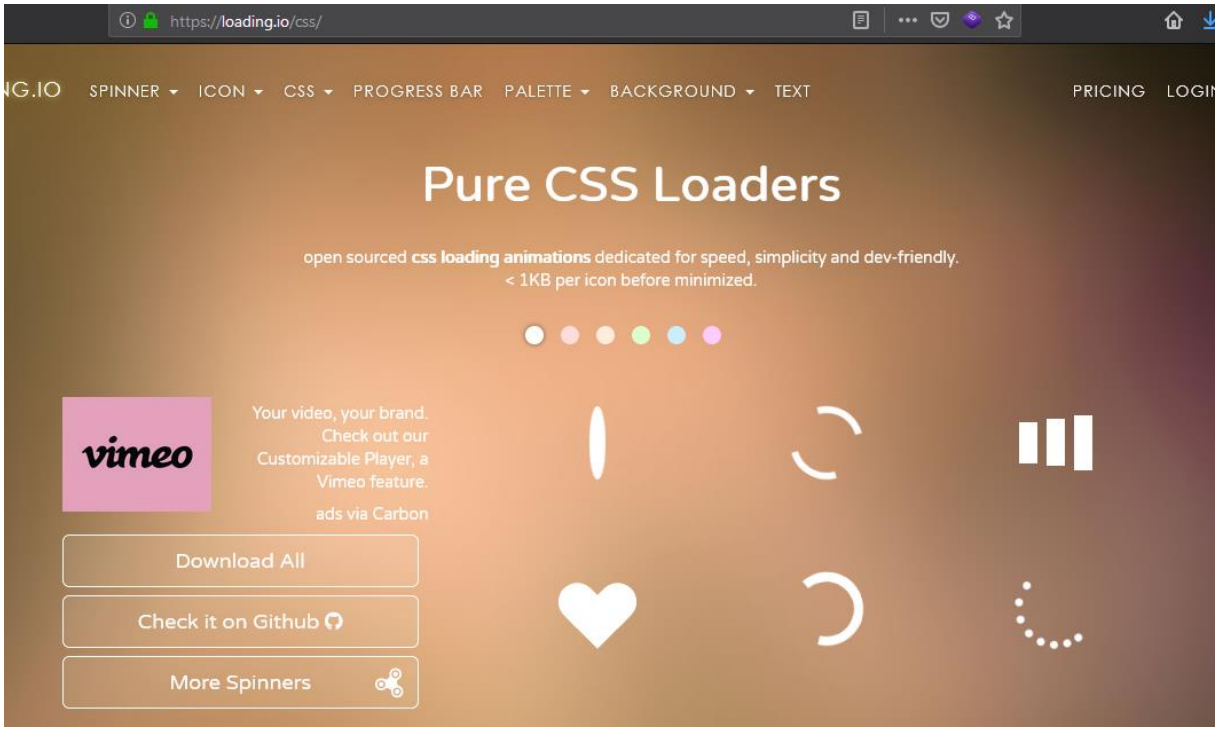
3.9. Agregaremos un botón de retroceso de esta vista.

```
goBackHandler = e => {
    this.props.history.goBack();
};
render() {
```

## 4. Componente de carga

4.1. Nuestra interfaz envía al servidor pero no le comunica esto al usuario. Sería bueno agregar un componente de carga para mostrarle esto. Entraremos a la siguiente URL:

https://loading.io/css/




4.2. Copiaremos el html y css de alguno de estos componentes, y lo pegaremos en un archivo que llamaremos Spinner.js y Spinner.css que ubicaremos dentro de una carpeta Spinner dentro de UI en Components. La idea es que el html esté dentro del render de Spinner.js (adjunto un ejemplo) y el css dentro de Spinner.css

dawa ▸ lab09 ▸ src ▸ components ▸ Spinner ▸ Spinner.js ▸ ...

```
import React, { Fragment } from 'react';

import './Spinner.css';

const Spinner = () => {
    return (
        <Fragment>
            <div className="lds-ellipsis">
                <div />
                <div />
                <div />
                <div />
            </div>
        </Fragment>
    );
};

export default Spinner;
```

4.3. Lo importaremos en nuestro ProfileEdit.js

```
import axios from '../../../utils/axios';
import { PROFILE_EDIT } from '../../../utils/urls';

import Spinner from '../../../components/Spinner/Spinner';

class ProfileEdit extends Component {
```

4.4. Haremos una modificación al render. Crearemos la variable content que contendrá al Spinner. Luego haremos un if que reemplazará el contenido de content por todo el contenido dentro de nuestro formulario (en la captura adjuntada, solamente estoy poniendo la primera parte, pero es todo el contenido de content y cerramos el if)

```
let content = <Spinner />;
if (!this.state.loading) {
    content = (
        <form onSubmit={this.submitHandler}>
            <div className="card-body">
                <div className="form-group">
```

4.5. En nuestro return, al haber retirado el content, lo reemplazaremos por {content}

```
return (
    <div className="container d-flex justify-content-center mt-4 mb-4">
        <div className="card">
            <div className="card-header">Editar Perfil</div>
            {content}
        </div>
    </div>
);
```

6. **Finalizar la sesión**
   6.1. Apagar el equipo virtual
   6.2. Apagar el equipo

## Conclusiones:

Indicar las conclusiones que llegó después de los temas tratados de manera práctica en este laboratorio.