

Universidad de Nariño.
Ingeniería de Sistemas.
Diplomado de actualización en nuevas tecnologías para el desarrollo
de Software.
Taller Unidad 3 Frontend

Marlon Tenganan

Desarrollar una aplicación Frontend en React que haga uso del Backend desarrollado en el taller anterior.

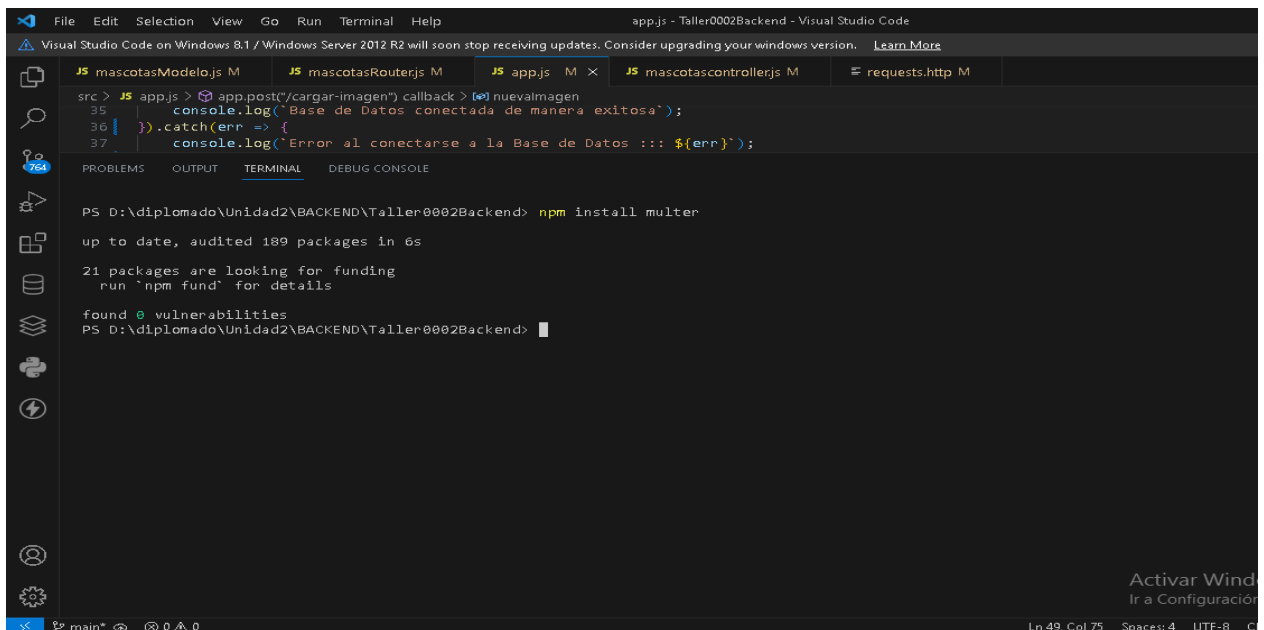
las imágenes, logotipos quedan a criterio del estudiante.

Para la calificación se tendrán en cuenta los siguientes criterios.

1. Creación de componentes y métodos asociados a los mismos.
(3 Ptos).

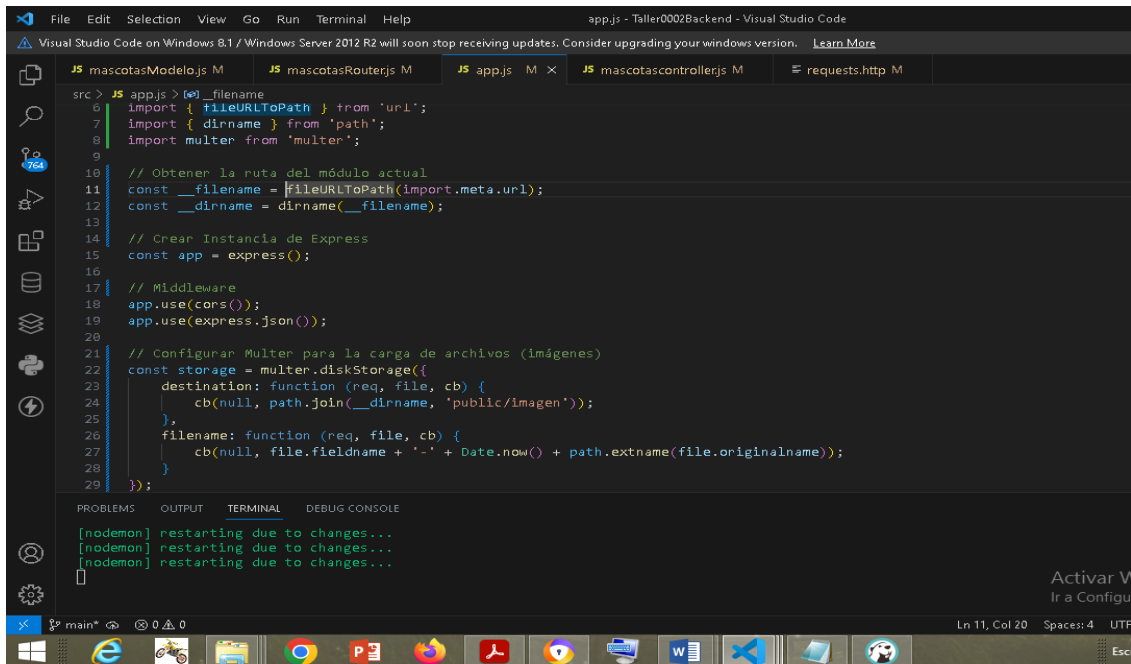
Para este Taller lo que hice fue modificar el backend

Descargue la extensión multer que es utilizada en node js y expeess para que se pueda utilizar para almacenar archivos



```
src > JS app.js > app.post('/cargar-imagen') callback > (nuevalimagen
35 | console.log('Base de Datos conectada de manera exitosa');
36 | }).catch(err => {
37 | console.log('Error al conectarse a la Base de Datos ::: ${err}');
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
PS D:\diplomado\Unidad2\BACKEND\Taller0002Backend> npm install multer
up to date, audited 189 packages in 6s
21 packages are looking for funding
run 'npm fund' for details
found 0 vulnerabilities
PS D:\diplomado\Unidad2\BACKEND\Taller0002Backend>
```

Ahora en mi app creo los método y funciones para que con multer se puede cargar la ruta de la imagen de cada perro, con un método storage = multer.diskStorage, el cual maneja la ruta en donde se almacena la imagen



```
src> JS app.js > |__| _filename
6   import { fileURLToPath } from 'url';
7   import { dirname } from 'path';
8   import multer from 'multer';
9
10  // Obtener la ruta del módulo actual
11  const __filename = fileURLToPath(import.meta.url);
12  const __dirname = dirname(__filename);
13
14  // Crear Instancia de Express
15  const app = express();
16
17  // Middleware
18  app.use(cors());
19  app.use(express.json());
20
21  // Configurar Multer para la carga de archivos (Imágenes)
22  const storage = multer.diskStorage({
23    destination: function (req, file, cb) {
24      cb(null, path.join(__dirname, 'public/imagen'));
25    },
26    filename: function (req, file, cb) {
27      cb(null, file.fieldname + '-' + Date.now() + path.extname(file.originalname));
28    }
29  });
```

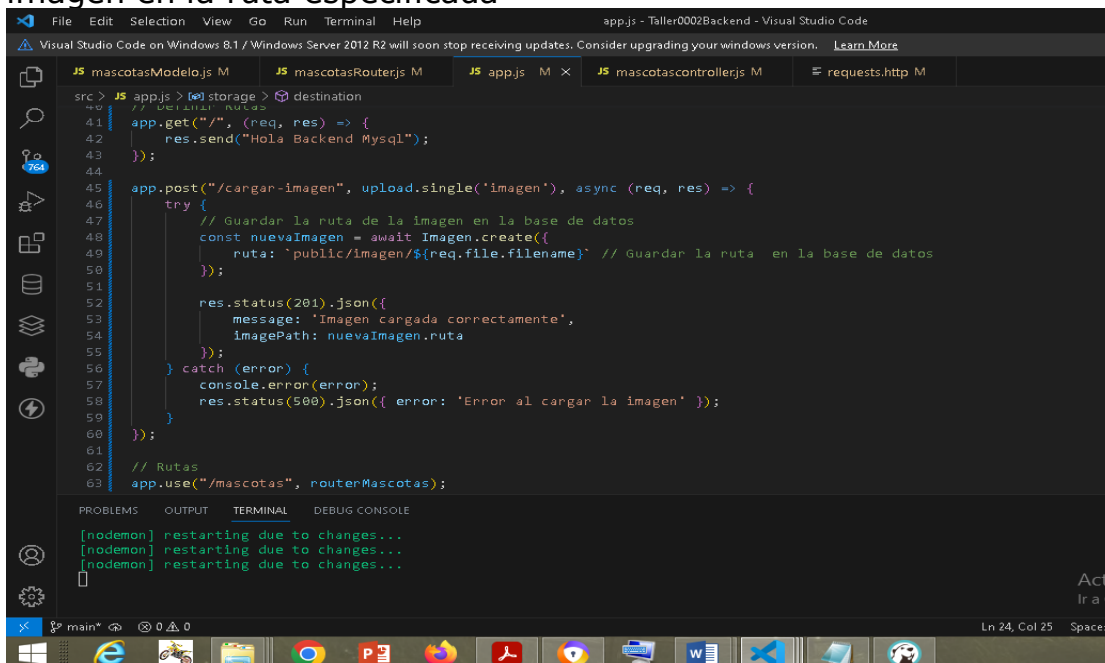
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
[nodemon] restarting due to changes...
[nodemon] restarting due to changes...
[nodemon] restarting due to changes...
```

Activar W...
Ir a Configu...

Ln 11, Col 20 Spaces: 4 UTF

Dentro del controlador también creamos un método post para cargar la imagen en la ruta especificada



```
src> JS app.js > |__| storage > |__| destination
41  app.get("/", (req, res) => {
42    res.send("Hola Backend Mysql");
43  });
44
45  app.post("/cargar-imagen", upload.single('imagen'), async (req, res) => {
46    try {
47      // Guardar la ruta de la imagen en la base de datos
48      const nuevaImagen = await Imagen.create({
49        ruta: `public/imagen/${req.file.filename}` // Guardar la ruta en la base de datos
50      });
51
52      res.status(201).json({
53        message: 'Imagen cargada correctamente',
54        imagePath: nuevaImagen.ruta
55      });
56    } catch (error) {
57      console.error(error);
58      res.status(500).json({ error: 'Error al cargar la imagen' });
59    }
60  });
61
62  // Rutas
63  app.use("/mascotas", routerMascotas);
```

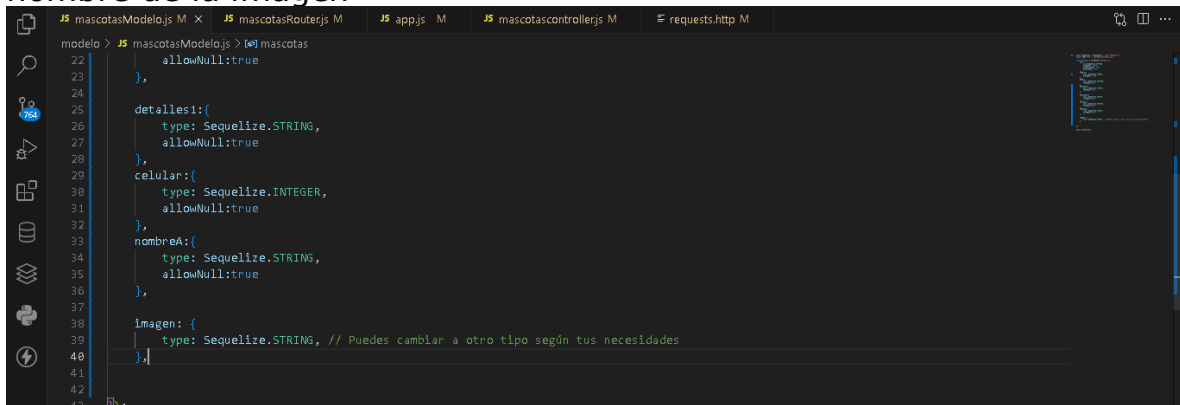
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
[nodemon] restarting due to changes...
[nodemon] restarting due to changes...
[nodemon] restarting due to changes...
```

Activar W...
Ir a C...

Ln 24, Col 25 Spaces:

Dentro de mi modelo declaramos el campo imagen que almacenara el nombre de la imagen



```
22     allowNull:true
23   },
24
25   detalles:{
26     type: Sequelize.STRING,
27     allowNull:true
28   },
29   celular:{
30     type: Sequelize.INTEGER,
31     allowNull:true
32   },
33   nombreA:{
34     type: Sequelize.STRING,
35     allowNull:true
36   },
37
38   imagen: {
39     type: Sequelize.STRING, // Puedes cambiar a otro tipo según tus necesidades
40   },
41 }
42
43
```

También con lo anterior en mi controlador declaramos a imagen con file

```
controladores > JS mascotascontrollerjs > crear1
17
18
19
20 const dataset = {
21   nombre: req.body.nombre,
22   edad: req.body.edad,
23   detalles: req.body.detalles,
24   detalles1: req.body.detalles1,
25   celular: req.body.celular,
26   nombreA: req.body.nombreA,
27   // En lugar de almacenar la ruta en la base de datos, almacenamos el nombre del
28   imagen: req.file.filename,
29 };
30
31 // Usar Sequelize para crear el recurso (mascota) en la base de datos
32 const resultado = await Mascota.create(dataset);
33 res.status(201).json({
34   mensaje: "Mascota creada correctamente",
35   resultado
36 });
37 } catch (error) {
38   console.error(`Error al crear la mascota: ${error}`);
39   res.status(500).json({
40     mensaje: `Error al crear la mascota: ${error}`
41   });
42 }
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

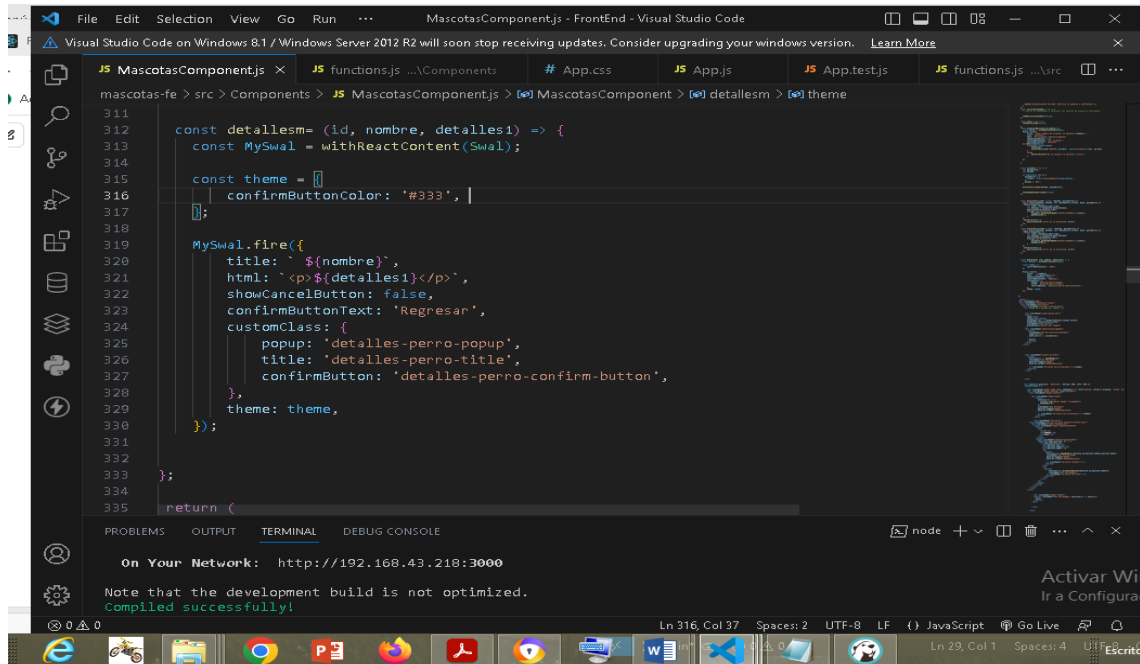
[nodemon] restarting due to changes...
[nodemon] restarting due to changes...
[nodemon] restarting due to changes...

Al igual en la base de datos creamos un campo imagen que guardara el nombre de tipo string

	detalles1	celular	nombreA	imagen
1	El rottweiler es un perro de tam	2,147,483,647	Marlon Tenganan	perro1
2	betuoso Muchos colombianos dicen ten	309,790,116	Jovany Tenganan	perro2
3	los perros shiwawa son de raza	12,345	Marlon Tenganan	perro3

2. Uso de HTML 5 y JavaScript (Se debe desarrollar una estructura ordenada, con código legible y documentado).
(1 Pto.).

en cuanto al front end aumentamos método
utilizamos java para mostrar los detalles del perro. sweetalert2 para
mostrar un cuadro de diálogo (modal) con detalles sobre una mascota
cuando la función

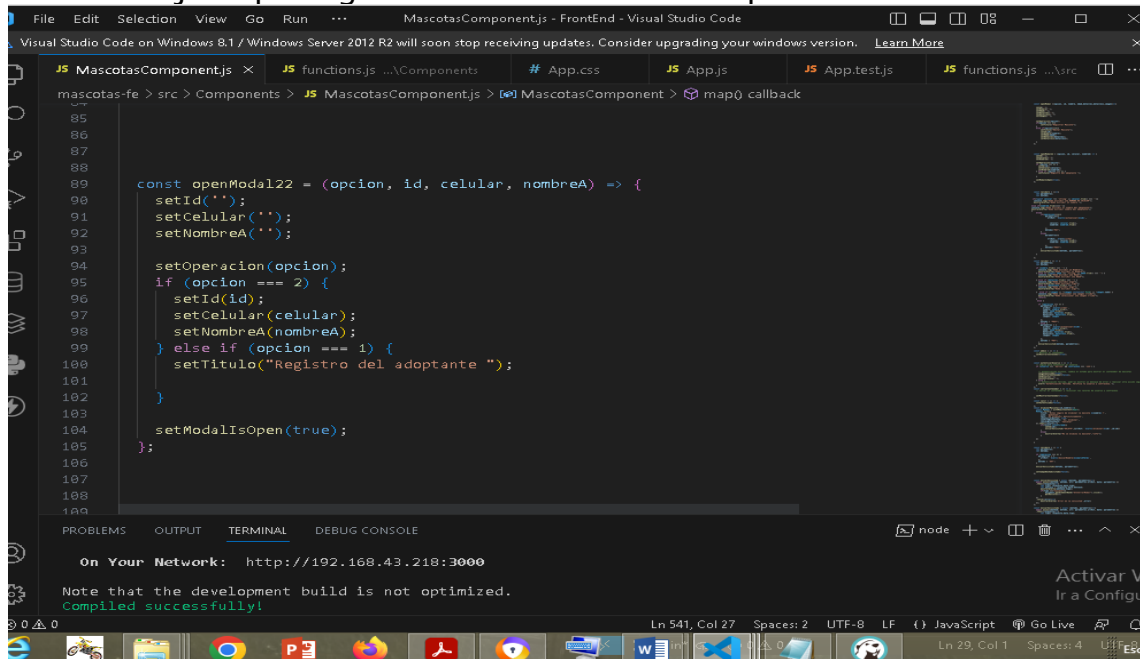


The screenshot shows a Visual Studio Code editor with a file explorer on the left and a code editor in the center. The file explorer shows a project structure with folders like 'src' and 'components'. The code editor displays a JavaScript file named 'MascotasComponent.js' with the following code:

```
311
312 const detalles= (id, nombre, detalles1) => {
313   const MySwal = withReactContent(Swal);
314
315   const theme = {
316     confirmButtonColor: '#333',
317   };
318
319   MySwal.fire({
320     title: `${nombre}`,
321     html: `<p>${detalles1}</p>`,
322     showCancelButton: false,
323     confirmButtonText: 'Regresar',
324     customClass: {
325       popup: 'detalles-perro-popup',
326       title: 'detalles-perro-title',
327       confirmButton: 'detalles-perro-confirm-button',
328     },
329     theme: theme,
330   });
331
332 };
333
334
335 return (
```

The bottom of the screenshot shows the terminal output with the message "On Your Network: http://192.168.43.218:3000" and "Note that the development build is not optimized. Compiled successfully!".

Función e java para guardar los datos del adoptante



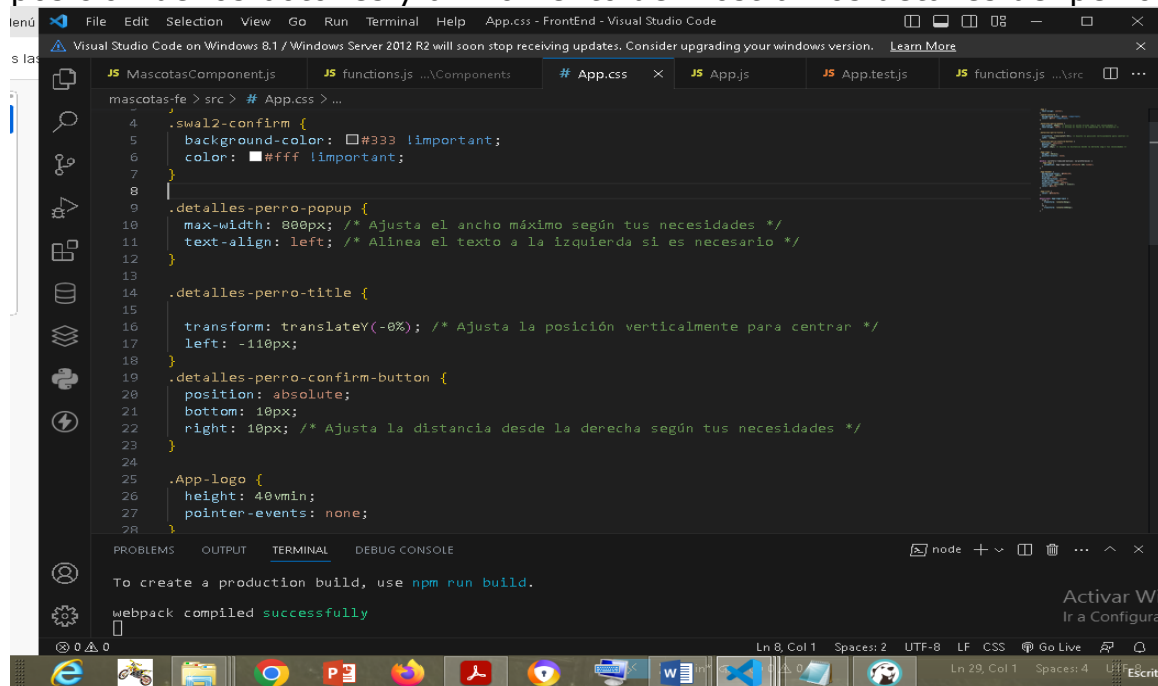
The screenshot shows a Visual Studio Code editor with a file explorer on the left and a code editor in the center. The file explorer shows a project structure with folders like 'src' and 'components'. The code editor displays a JavaScript file named 'MascotasComponent.js' with the following code:

```
85
86
87
88
89 const openModal22 = (opcion, id, celular, nombreA) => {
90   setId('');
91   setCelular('');
92   setNombreA('');
93
94   setOperacion(opcion);
95   if (opcion === 2) {
96     setId(id);
97     setCelular(celular);
98     setNombreA(nombreA);
99   } else if (opcion === 1) {
100     setTitulo("Registro del adoptante ");
101   }
102
103   setModalIsOpen(true);
104
105 };
106
107
108
109
```

The bottom of the screenshot shows the terminal output with the message "On Your Network: http://192.168.43.218:3000" and "Note that the development build is not optimized. Compiled successfully!".

3. Estilos CSS (Uso de Bootstrap), se debe generar una interface ordenada estructurada y agradable para el usuario final. (1 Pto.)

en cuanto al front end aumentamos métodos al css para manejar la posición de los botones y al momento de mostrar los detalles del perro

A screenshot of the Visual Studio Code editor interface. The main editor window displays a CSS file named 'App.css' with the following content:

```
4 .swal2-confirm {  
5   background-color: #333 !important;  
6   color: #fff !important;  
7 }  
8  
9  
10 .detalles-perro-popup {  
11   max-width: 800px; /* Ajusta el ancho máximo según tus necesidades */  
12   text-align: left; /* Alinea el texto a la izquierda si es necesario */  
13 }  
14  
15  
16 .detalles-perro-title {  
17   transform: translateY(-50%); /* Ajusta la posición verticalmente para centrar */  
18   left: -110px;  
19 }  
20  
21 .detalles-perro-confirm-button {  
22   position: absolute;  
23   bottom: 10px;  
24   right: 10px; /* Ajusta la distancia desde la derecha según tus necesidades */  
25 }  
26  
27  
28 .App-logo {  
29   height: 40vmin;  
30   pointer-events: none;  
31 }  
32
```

The interface includes a sidebar on the left with icons for Explorer, Search, Source Control, and Run and Debug. The bottom panel shows the 'TERMINAL' tab with the message 'webpack compiled successfully'. The status bar at the bottom indicates 'Ln 8, Col 1' and 'Spaces: 2'.

También el uso de bootstrap para que la impresión de los datos de los perros se haga en una tarjeta y no en una tabla como estaba antes

```
520
521
522 <div className="row mt-3">
523   <div className="col-12 col-lg-8 offset-0 offset-lg-2">
524     <div className="row">
525       {(searchResults?.length > 0 ? searchResults : mascotas)?.map((mascota, i) => (
526         <div className="col-md-4 mb-3" key={mascota.id}>
527           <div className="card">
528             <div
529               className="card-image"
530               style={{ backgroundImage: `url(${mascota.imagen})`, height: '200px', backgroundSize: 'cover' }}
531             ></div>
532             <div className="card-body">
533               <h5 className="card-title text-start">{mascota.nombre}</h5>
534               <p className="card-text text-start">Edad: {mascota.edad}</p>
535               <p className="card-text text-start">{mascota.detalles}</p>
536             <button
537               onClick={() => detallesm(mascota.id, mascota.nombre, mascota.detalles1)}
538               className="btn btn-primary mr-2"
539             >
540               Detalles
541             </button>
542           </div>
543         </div>
544       )}
545     </div>
546   </div>
547 </div>
```

On Your Network: http://192.168.43.218:3000

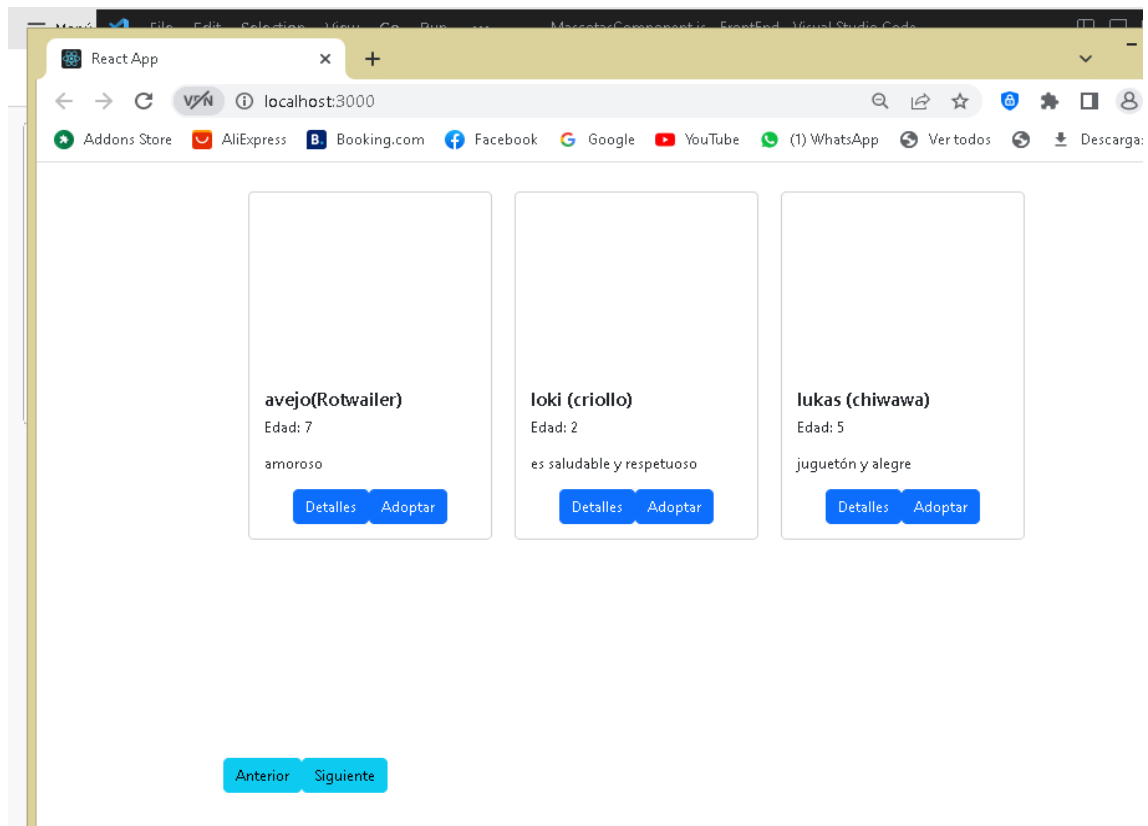
Note that the development build is not optimized.
Compiled successfully!

Botones adoptar y detalles

```
527
528
529 <div className="card">
530   <div
531     className="card-image"
532     style={{ backgroundImage: `url(${mascota.imagen})`, height: '200px', backgroundSize: 'cover' }}
533   ></div>
534   <div className="card-body">
535     <h5 className="card-title text-start">{mascota.nombre}</h5>
536     <p className="card-text text-start">Edad: {mascota.edad}</p>
537     <p className="card-text text-start">{mascota.detalles}</p>
538   </div>
539   <button
540     onClick={() => detallesm(mascota.id, mascota.nombre, mascota.detalles1)}
541     className="btn btn-primary mr-2"
542   >
543     Detalles
544   </button>
545   <button
546     onClick={() => openModal22(2)}
547     className="btn btn-primary mr-2"
548     data-bs-toggle="modal"
549     data-bs-target="#modalMascotas"
550   >
551     Adoptar
552   </button>
553 </div>
```

On Your Network: http://192.168.43.218:3000

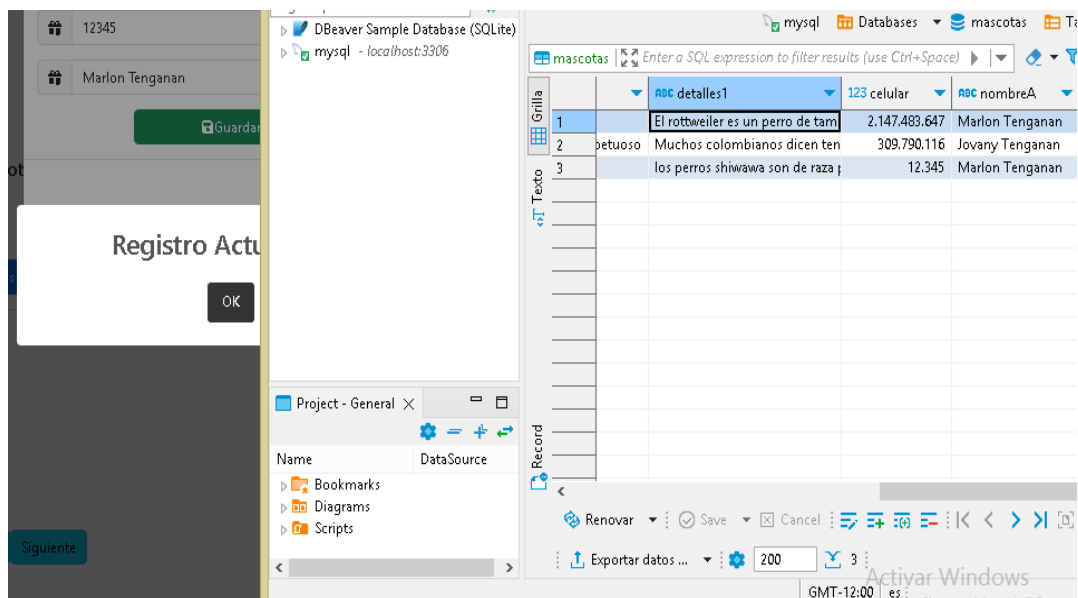
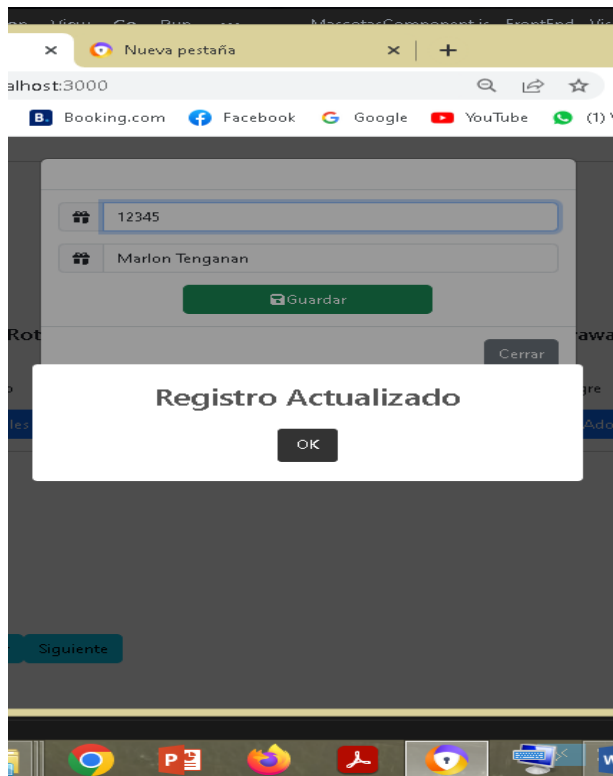
Note that the development build is not optimized.
Compiled successfully!



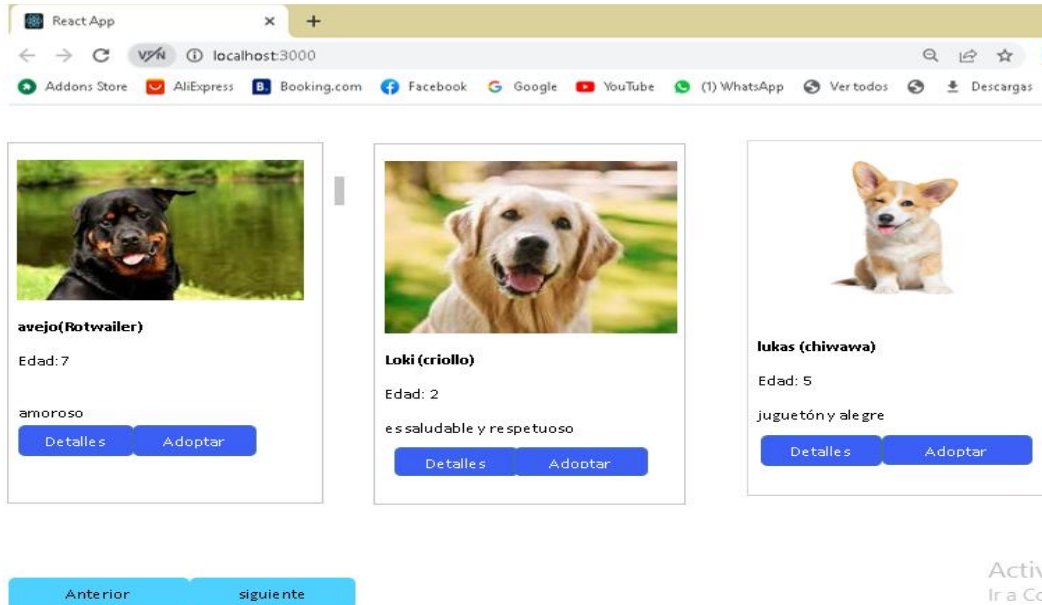
Botón detalles



Botón adoptar



Llamado de imágenes



Elaborar un informe que detalle el proceso de construcción y la implementación del aplicativo.

Subir el código y el informe a un repositorio remoto (GitHub), y enviar el link del mismo al correo vauxr@udenar.edu.co, dentro del cuerpo del correo se debe incluir los datos del estudiante.