PROFESIONAL

APLICACIÓN BOLETÍN INFORMATIVO - 3

INSTRUCCIONES:

Objetivo del proyecto:

En la clase 59, aprendiste sobre varios ciclos de vida y estados de componentes y desarrollaste una aplicación de contador.

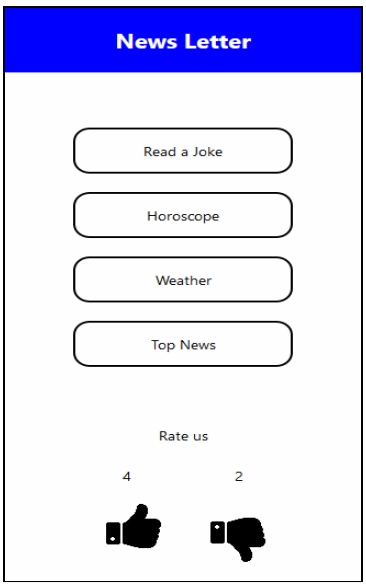
En el proyecto del día de hoy, completarán la aplicación del boletín al realizar un seguimiento de las reseñas de me gusta y no me gusta.

* Esta es una continuación de los Proyectos 57 y 58. Así que asegúrate de completarlos antes de intentar este. *

Historia:

En una encuesta que realizaste, ¡el noventa por ciento de tus amigos dijeron que realmente se beneficiarían de una aplicación tipo Boletín informativo.

Ya has comenzado a crear esta increíble aplicación para tus amigos. Has creado diferentes botones para que el usuario navegue rápidamente a diferentes pantallas. También conectaste esta aplicación a la base de datos de Firebase. Ahora tienes que escribir el código, para realizar un seguimiento de las reseñas de me gusta y no me gusta.



PROFESIONAL

APLICACIÓN BOLETÍN INFORMATIVO - 3

*** Esto es solo para tu referencia. Esperamos que apliques tu propia creatividad en el proyecto.**

Primeros Pasos:

1. Inicia sesión en Snack.
2. Abre el snack del Proyecto 58 y haz una copia de ese proyecto.
3. Cambia el nombre del snack como **Proyecto 59.**
4. Empieza a editar el snack

Tareas específicas para completar el proyecto:

1. Cada componente puede contener un estado y se llama usando **this.state.** El estado de un componente se llama solo en un constructor.
   - No olvides usar **super()** para heredar las propiedades de un componente.
   - A medida que hacemos un seguimiento de las reseñas, crea tanto el me gusta como el no me gusta en el estado.

```
constructor(){
  super();
  this.state={
    like:0,
    dislike:0
  }
}
```

2. En la **etiqueta de texto**, muestra las propiedades usando **this.state.<property_name>**

```
<Text>{this.state.like}</Text>
<Text>{this.state.dislike}</Text>
```

3. Crea una función que cambie el estado de los me gusta y no me gusta, incrementando el estado actual de me gusta o no me gusta en 1.
   - Utiliza **setState** – *establecer estado*, para esto.
4. Llama a la función usando **onPress** para la **etiqueta JSX <TouchableOpacity>** de las imágenes Me gusta y No me gusta.
5. Prueba si el incremento está ocurriendo.

- Si es así, ¡felicidades por tu primera aplicación react native usando estados y ciclos de vida!

6. Prueba la aplicación en tu dispositivo móvil antes de enviarla.

* Consulta las imágenes dadas arriba como referencia.

Envío del proyecto:

1. Guarda tu expo snack **(Ctrl / Comando + S).**
2. Copia la URL y envíalo en el panel de Proyectos del alumno, con el número de clase correcto.

Actividad desafiante adicional:

Diseña un pronóstico del tiempo usando json, ciclos de vida y estados.

En la clase **WeatherScreen** – *pantalla del clima*:

1. Crea un estado inicial, en el constructor para la clase WeatherScreen.

```
export default class WeatherScreen extends Component {
  constructor() {
    super();
    this.state = {
      weather: '',
    };
  }
```

2. Utiliza la declaración import – *importar*, - **import axios from 'axios';**

3. Crea una función llamada **getWeather** – *obtener clima*, que puede obtener la API meteorológica para obtener la información.
   - Mira la función como referencia.
   - La URL utilizada es https://fcc-weather-api.glitch.me/api/current?lat=35&lon=139

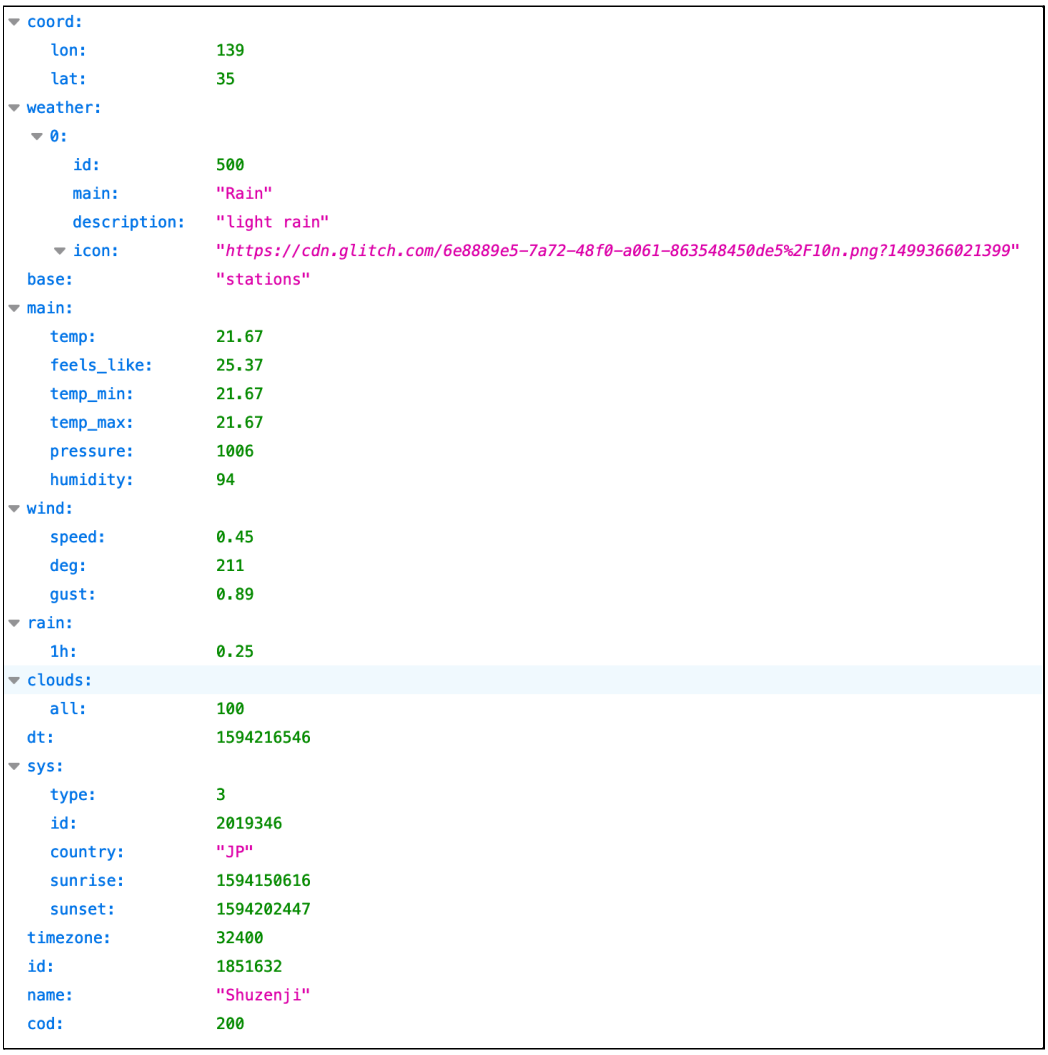
```
getWeather = async () => {
  //chnage latitude and longitude
  var url = 'https://fcc-weather-api.glitch.me/api/current?lat=35&lon=139';
  return fetch(url)
    .then(response => response.json())
    .then(responseJson => {
      this.setState({
        weather: responseJson,
      });
    })
    .catch(error => {
      console.error(error);
    });
};
```

4. Llama a esta función usando la palabra clave **this** en el componente didmount.

```
componentDidMount = () => {
  this.getWeather();
};
```

5. Si se obtiene la URL, muéstrala en la etiqueta **Text JSX** usando **this.state.weather.<data>.**
6. Si no se obtiene la URL, se muestra como cargando. Busca referencias al final.

***Nota:** Como referencia, así es como se ven los datos JSON:



7. Puedes elegir mostrar cualquier cosa; simplemente reemplaza los datos con el atributo que desees mostrar.

Referencia:

PROFESIONAL

APLICACIÓN BOLETÍN INFORMATIVO - 3

```
  render() {
    if (this.state.weather === '') {
      return (
        <View style={styles.container}>
          <Text>Loading...</Text>
        </View>
      );
    } else {
      return (
        <View style={styles.container}>
          <Text>Weather : {this.state.weather.weather[0].description}</Text>
          <Text>Wind Speed : {this.state.weather.wind.speed}</Text>
          <Text>Temprature : {this.state.weather.main.temp}</Text>
          <Text>Min Temprature : {this.state.weather.main.temp_min}</Text>
          <Text>Max Temprature : {this.state.weather.main.temp_max}</Text>
          <Text>Pressure : {this.state.weather.main.pressure}</Text>
          <Text>Humidity : {this.state.weather.main.humidity}</Text>
        </View>
      );
    }
  }
}
```

Pistas:

PROFESIONAL

APLICACIÓN BOLETÍN INFORMATIVO - 3

1. Ejecuta la función cuando se pulsan los botones Me gusta y No me gusta como se muestra a continuación.

```
<TouchableOpacity onPress ={this.likecount}>

  <Image
    style={{ width: 50, height: 50, marginLeft: 5 }}
    source={require('../assets/thumbs-up-hand-symbol.png')}
  />
</TouchableOpacity>
```

2. Puedes estructurar tus funciones likeCount – *cuenta de me gusta*, y dislikeCount – *cuenta de no me gusta*, como se muestra a continuación:

```
likecount=()=>{
  this.setState({like:this.state.like+1});
}
dislikecount=()=>{
  this.setState({dislike:this.state.dislike+1});
}
```

RECUERDA ... Haz tu mejor esfuerzo, eso es lo más importante.
Después de enviar tu proyecto, tu maestra te enviará comentarios sobre tu trabajo.

——————— XXX ——————— XXX ——————— XXX ——————— XXX ——————— XXX ———————