APLICACIONES MÓVILES MULTIPLATAFORMA

LABORATORIO N° 09

Acceso a la cámara

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

Criterio de Evaluación	*Excelente (4pts)*	*Bueno (3pts)*	*Requiere mejora (2pts)*	*No acept. (0pts)*	*Puntaje Logrado*
Agregar componentes de cámara					
Usar axios para subir archivos					
Consumir APIs de terceros					
Realiza con éxito lo propuesto en el laboratorio					
Es puntual y redacta el informe adecuadamente					

Laboratorio 09:
Acceso a la cámara

Objetivos:

Al finalizar el laboratorio el estudiante será capaz de:

- Entender el funcionamiento del componente Camera
- Desarrollar aplicaciones web enfocadas a componentes
- Manejar de manera básica axios para consumir APIs

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 de API**
   1.1. En el proyecto **dawa_api**, instalaremos **multer**:

```
>npm install --save multer
```

   1.2. Ahora, modificaremos las líneas del archivo server.js que hacen invocación de las rutas para que podamos invocar a multer

```
router.get('/', function(req, res) {
    res.json({ message: 'genial! bienvenido a nuestra api!' });
});

const userRouter = require('./routes/user');
const fileRouter = require('./routes/file');
router.use('/user', userRouter);
router.use('/file', fileRouter);

app.use('/api', router);
```

   1.3. Crearemos un archivo files.js dentro de la carpeta middlewares (en caso de no existir, crearla)

```
const multer = require('multer');

const storage = multer.diskStorage({
    destination: './temp',
    filename(req, file, cb) {
        cb(null, `${new Date()}-${file.originalname}`);
    },
});

module.exports = multer({ storage });
```

1.4. Finalmente, crearemos el archivo file.js dentro de routes

```
const express = require('express');
const multer = require('multer');

const router = express.Router();
const upload = multer({ dest: 'uploads/' });

router.post('/upload', upload.single('file'), (req, res) => {
    console.log('-----------------', req.file.path);
});

module.exports = router;
```

2. **Configuración de proyecto**

2.1. Copie el contenido del laboratorio 7 (la clase anterior) a excepción de la carpeta node_modules en una nueva carpeta llamada lab08 y reinstale todas las dependencias:

```
>npm install
```

2.2. En la nueva carpeta lab08, instalaremos las siguientes dependencias:

```
>npm install react-native-camera --save
```

```
>react-native link react-native-camera
```

2.3. Modificaremos el archivo android/app/build.gradle, agregando solamente la siguiente línea

```
android {
    compileSdkVersion rootProject.ext.compileSdkVersion

    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }

    defaultConfig {
        applicationId "com.lab07"
        minSdkVersion rootProject.ext.minSdkVersion
        targetSdkVersion rootProject.ext.targetSdkVersion
        versionCode 1
        versionName "1.0"
        missingDimensionStrategy 'react-native-camera', 'general'
    }
    splits {
```

2.4. Agregaremos la vista de la cámara en el enrutador de nuestro proyecto. Modicaremos App.js

```
import AuthLoadingScreen from './src/screens/AuthLoadingScreen/AuthLoadingScreen';
import SignInScreen from './src/screens/SignIn/SignIn';
import SignUpScreen from './src/screens/SignUp/SignUp';
import HomeScreen from './src/screens/Home/Home';
import ChatScreen from './src/screens/Chat/Chat';
import Location from './src/screens/Location/Location';
import Camera from './src/screens/Camera/Camera';

const AppStack = createDrawerNavigator({
    Home: HomeScreen,
    Chat: ChatScreen,
    Location: Location,
    Camera: Camera
});
const AuthStack = createBottomTabNavigator({
    SignIn: SignInScreen,
    SignUp: SignUpScreen
});
```

2.5. Agregaremos un botón en nuestro chat para tomar la foto en el render de Chat.js

```
render() {
    const user = { _id: this.state.userId || -1 };
    return (
        <Fragment>
            <Modal
                animationType="slide"
                transparent={false}
                visible={this.state.modalVisible}
            >
                <View>
                    <Button
                        onPress={this.chatHandler}
                        title="Regresar al Chat"
                        color="#841584"
                    />
                    <Button
                        onPress={this.cameraHandler}
                        title="Tomar foto"
                        color="green"
                    />
                    <Button
                        onPress={this.backHandler}
                        title="Regresar al Inicio"
                        color="red"
                    />
                </View>
            </Modal>
            <GiftedChat
                placeholder="Escribe algo..."
                renderActions={() => {
```

2.6. No nos olvidemos de agregar en Chat.js el evento para viajar a la pantalla de la cámara

```
backHandler = () => {
    this.setState({ modalVisible: false }, () => {
        this.props.navigation.navigate('Home');
    });
};
cameraHandler = () => {
    this.setState({ modalVisible: false }, () => {
        console.log('si hace click');
        this.props.navigation.navigate('Camera');
        //CameraRoll.saveToCameraRoll('file:///sdcard/img.png', 'photo')
    });
};
render() {
    const user = { _id: this.state.userId || -1 };
    return (
        <Fragment>
            <Modal
```

3. Creación de componente Cámara

3.1. Ahora sí, crearemos la carpeta Camera dentro de screens y dentro de ella el archivo Camera.js con el siguiente contenido.

```
import React, { Component } from 'react';
import { StyleSheet, Text, TouchableOpacity, View } from 'react-native';
import { RNCamera } from 'react-native-camera';

const PendingView = () => (
    <View
        style={{
            flex: 1,
            backgroundColor: 'lightgreen',
            justifyContent: 'center',
            alignItems: 'center'
        }}
    >
        <Text>Waiting</Text>
    </View>
);

class CameraScreen extends Component {
    render() {
        return (
            <View style={styles.container}>
                <RNCamera
                    style={styles.preview}
                    type={RNCamera.Constants.Type.back}
                    flashMode={RNCamera.Constants.FlashMode.on}
                    androidCameraPermissionOptions={{
                        title: 'Permission to use camera',
                        message: 'We need your permission to use your camera',
                        buttonPositive: 'Ok',
                        buttonNegative: 'Cancel'
                    }}
                    androidRecordAudioPermissionOptions={{
                        title: 'Permission to use audio recording',
                        message: 'We need your permission to use your audio',
                        buttonPositive: 'Ok',
                        buttonNegative: 'Cancel'
                    }}
                >
                    {({ camera, status, recordAudioPermissionStatus }) => {
                        if (status !== 'READY') return <PendingView />;
                        return (
```

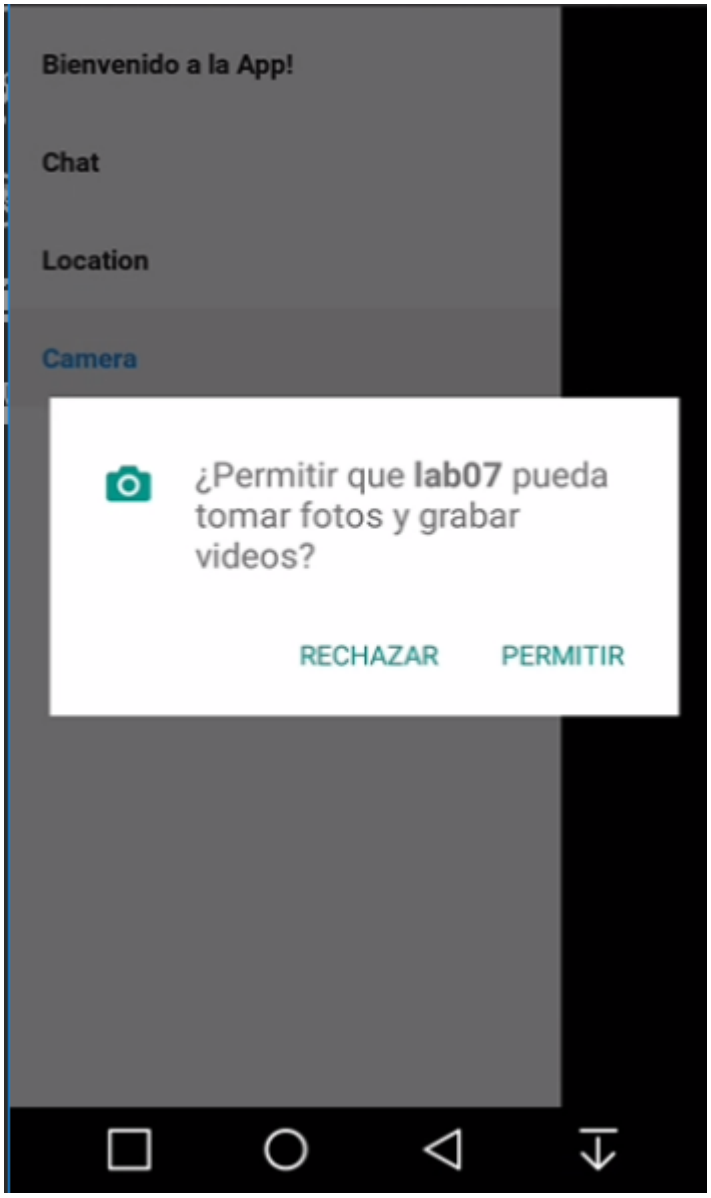
```
                    androidRecordAudioPermissionOptions={{
                        title: 'Permission to use audio recording',
                        message: 'We need your permission to use your audio',
                        buttonPositive: 'Ok',
                        buttonNegative: 'Cancel'
                    }}
                >
                    {({ camera, status, recordAudioPermissionStatus }) => {
                        if (status !== 'READY') return <PendingView />;
                        return (
                            <View
                                style={{
                                    flex: 0,
                                    flexDirection: 'row',
                                    justifyContent: 'center'
                                }}
                            >
                                <TouchableOpacity
                                    onPress={() => this.takePicture(camera)}
                                    style={styles.capture}
                                >
                                    <Text style={{ fontSize: 14 }}> SNAP </Text>
                                </TouchableOpacity>
                            </View>
                        );
                    }}
                </RNCamera>
            </View>
        );
    }

    takePicture = async function(camera) {
        const options = { quality: 0.5, base64: true };
        const data = await camera.takePictureAsync(options);
        //  eslint-disable-next-line
        console.log(data.uri);
    };
}

const styles = StyleSheet.create({
    container: {
        flex: 1,
        flexDirection: 'column',
        backgroundColor: 'black'
    },
    preview: {
        flex: 1,
        justifyContent: 'flex-end',
        alignItems: 'center'
    },
    capture: {
        flex: 0,
        backgroundColor: '#fff',
        borderRadius: 5,
        padding: 15,
        paddingHorizontal: 20,
        alignSelf: 'center',
        margin: 20
    }
});

export default CameraScreen;
```

3.2. Ejecute la aplicación. Al viajar a la vista de cámara, obtendrá la siguiente petición, que usted aceptará



3.3. Probemos que nuestro componente funciona, al tomar la foto (presionando el botón Snap) nos deberá lanzar una línea parecida en el **react-native log-android**

```
ReactNativeJS: file:///data/user/0/com.lab07/cache/Camera/108d63ce-2fcd-4bb5-8ef7-df35f838ea3f.jpg
```

4. Subiendo la imagen a nuestro servidor

4.1. Modificaremos la función takePicture de Camera.js para que haga el envío correspondiente al servidor express

```
takePicture = async function(camera) {
    const options = { quality: 0.5, base64: true };
    const token = await AsyncStorage.getItem('userToken');
    const data = await camera.takePictureAsync(options);

    console.log(data.uri);

    const formData = new FormData();
    formData.append('file', {
        uri: data.uri,
        name: 'my_photo.jpg',
        type: 'image/jpg'
    });
    axios({
        url: 'api/file/upload',
        method: 'POST',
        headers: {
            Authorization: token
        },
        data: formData
    })
        .then(response => {
            alert('Guardado con éxito!');
        })
        .catch(error => {
            alert('Hubo un error en el guardado!');
        });
};
```

4.2. Verifique que en su servidor express se ha creado una carpeta upload, y verifique que en la consola llega el mensaje de recepción del archivo.

5. **Finalizar la sesión**
   5.1. Apagar el equipo virtual
   5.2. Apagar el equipo

Conclusiones:

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