UNIVERSIDAD NACIONAL Y AUTÓNOMA DE MÉXICO
FACULTAD DE CONTADURÍA Y ADMINISTRACIÓN

“Fetch GET POST”

:: UNIDAD 4: Programación para Apple iOS

:: ACTIVIDAD COMPLEMENTARIA 3

MATERIA: **.:** PROGRAMACIÓN DE DISPOSITIVOS MÓVILES **:.**

CLAVE: 1668

LICENCIATURA EN INFORMÁTICA

PLAN 2012

REALIZO: Emmanuel Alejandro Pérez Hernández

No. 423142118

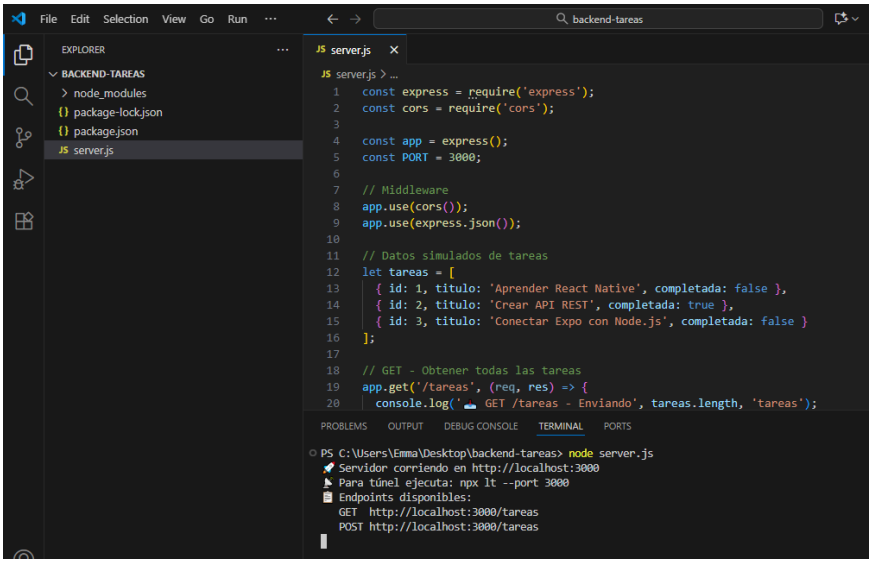
Grupo: 8691

ASESOR: MARTINEZ FERNANDEZ JUAN MANUEL

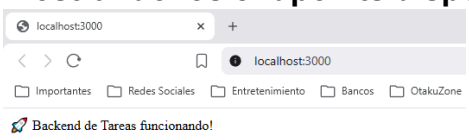
miércoles, 5 de noviembre de 2025

UNIDAD 4. Actividad Complementaria 3

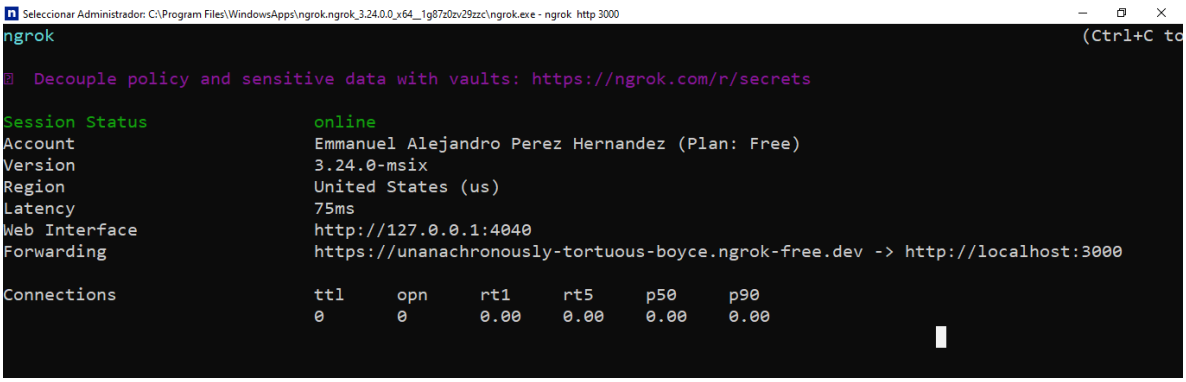
a) Ejecuta la app Node.js con el túnel.



Terminal de VS Code ejecutando el servidor Node.js en puerto 3000, mostrando los endpoints disponibles /tareas (GET y POST)



Backen funcionando



Terminal de ngrok creando túnel seguro, forwarding de URL pública https://unanachronously-tortuous-boyce.ngrok-free.dev hacia localhost:3000

b) Agrega la lógica a la app Expo para realizar un fetch de tipo GET.

Codigo en Snack.expo

```
import React, { useState, useEffect } from 'react';
import {
  View, Text, FlatList, TouchableOpacity, StyleSheet,
  SafeAreaView, StatusBar, Alert, ActivityIndicator
} from 'react-native';

//URL CORRECTA DE NGROK
const API_URL = 'https://unanachronously-tortuous-boyce.ngrok-free.dev';

const App = () => {
  const [tareas, setTareas] = useState([]);
  const [cargando, setCargando] = useState(false);

  //INCISO b) - FETCH GET para obtener tareas
  const obtenerTareas = async () => {
    try {
      setCargando(true);
      console.log('🔗 Obteniendo tareas de:', `${API_URL}/tareas`);

      const respuesta = await fetch(`${API_URL}/tareas`);
      const datos = await respuesta.json();

      console.log('✅ Tareas recibidas:', datos);
      setTareas(datos);
    } catch (error) {
      Alert.alert('Error', 'No se pudieron cargar las tareas');
      console.error('❌ Error:', error);
    } finally {
      setCargando(false);
    }
  };

  //INCISO d) - DOBLE FETCH: POST + GET
  const agregarTarea = async () => {
    try {
      const nuevaTarea = {
                titulo:   `Tarea   ${tareas.length   +   1}   -   ${new
Date().toLocaleTimeString()}`
      };

      console.log('📤 Enviando POST...');

      // PRIMER FETCH: POST para agregar
      const respuestaPost = await fetch(`${API_URL}/tareas`, {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
        },
```

```
      body: JSON.stringify(nuevaTarea),
    });

    if (respuestaPost.ok) {
      const tareaCreada = await respuestaPost.json();
      console.log('✅ Tarea agregada:', tareaCreada);

      Alert.alert('Éxito', 'Tarea agregada correctamente');

      // SEGUNDO FETCH: GET para actualizar lista
      console.log('🔄 Actualizando lista...');
      await obtenerTareas();
    }
  } catch (error) {
    Alert.alert('Error', 'No se pudo agregar la tarea');
    console.error('❌ Error:', error);
  }
};

// Cargar tareas al iniciar la app
useEffect(() => {
  obtenerTareas();
}, []);

// 🔷 INCISO c) - FLATLIST para mostrar tareas
const renderItem = ({ item }) => (
  <View style={[styles.item, item.completada && styles.itemCompletado]}>
    <Text style={styles.titulo}>{item.titulo}</Text>
    <Text style={styles.estado}>
      {item.completada ? '✅ Completada' : '⏳ Pendiente'}
    </Text>
    <Text style={styles.id}>ID: {item.id}</Text>
  </View>
);

return (
  <SafeAreaView style={styles.container}>
    <StatusBar backgroundColor="#4CAF50" />

    {/* Header */}
    <View style={styles.header}>
      <Text style={styles.tituloApp}>📋 Gestor de Tareas</Text>
      <Text style={styles.subtitulo}>Actividad 3 - Unidad 4</Text>
    </View>

    {/* Botón para agregar tarea - INCISO d) */}
    <TouchableOpacity style={styles.boton} onPress={agregarTarea}>
      <Text style={styles.textoBoton}>➕ Agregar Tarea</Text>
      <Text style={styles.textoSub}>POST + GET</Text>
    </TouchableOpacity>
```

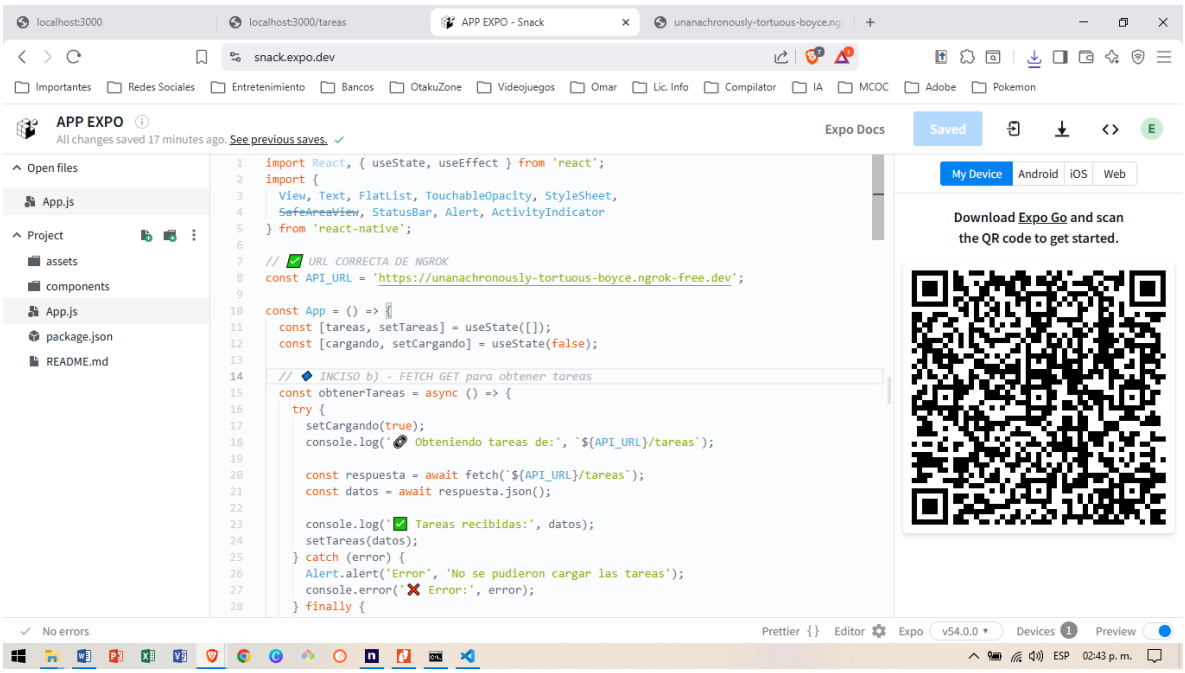
```
      {/* Lista de tareas - INCISO c) */}
      <View style={styles.listaContainer}>
        <Text style={styles.listaTitulo}>
          Lista de Tareas ({tareas.length})
        </Text>

        {cargando ? (
          <View style={styles.cargando}>
            <ActivityIndicator size="large" color="#4CAF50" />
                        <Text   style={styles.textoCargando}>Conectando   con
backend...</Text>
          </View>
        ) : (
          <FlatList
            data={tareas}
            renderItem={renderItem}
            keyExtractor={item => item.id.toString()}
            showsVerticalScrollIndicator={false}
            ListEmptyComponent={
                  <Text  style={styles.listaVacia}>No  hay  conexión  con  el
backend</Text>
            }
          />
        )}
      </View>
    </SafeAreaView>
  );
};

// Estilos
const styles = StyleSheet.create({
  container: { flex: 1, backgroundColor: '#f8f9fa' },
  header: { backgroundColor: '#4CAF50', padding: 20, paddingTop: 50 },
  tituloApp: { fontSize: 24, fontWeight: 'bold', color: 'white', textAlign:
'center' },
  subtitulo: { fontSize: 14, color: 'white', textAlign: 'center', marginTop:
5, opacity: 0.9 },
  boton: { backgroundColor: '#2196F3', padding: 15, margin: 20, borderRadius:
10, alignItems: 'center' },
  textoBoton: { color: 'white', fontSize: 18, fontWeight: 'bold' },
  textoSub: { color: '#E3F2FD', fontSize: 12, marginTop: 5 },
  listaContainer: { flex: 1, padding: 10 },
   listaTitulo:  {  fontSize:  16,  fontWeight:  'bold',  marginBottom:  10,
textAlign: 'center', color: '#333' },
  cargando: { alignItems: 'center', padding: 20 },
  textoCargando: { marginTop: 10, color: '#666' },
  listaVacia: { textAlign: 'center', marginTop: 50, fontSize: 16, color: '#999'
},
   item:  {  backgroundColor:  'white',  padding:  15,  marginVertical:  8,
marginHorizontal: 16, borderRadius: 8, borderLeftWidth: 4, borderLeftColor:
'#FF9800' },
  itemCompletado: { borderLeftColor: '#4CAF50', opacity: 0.8 },
```

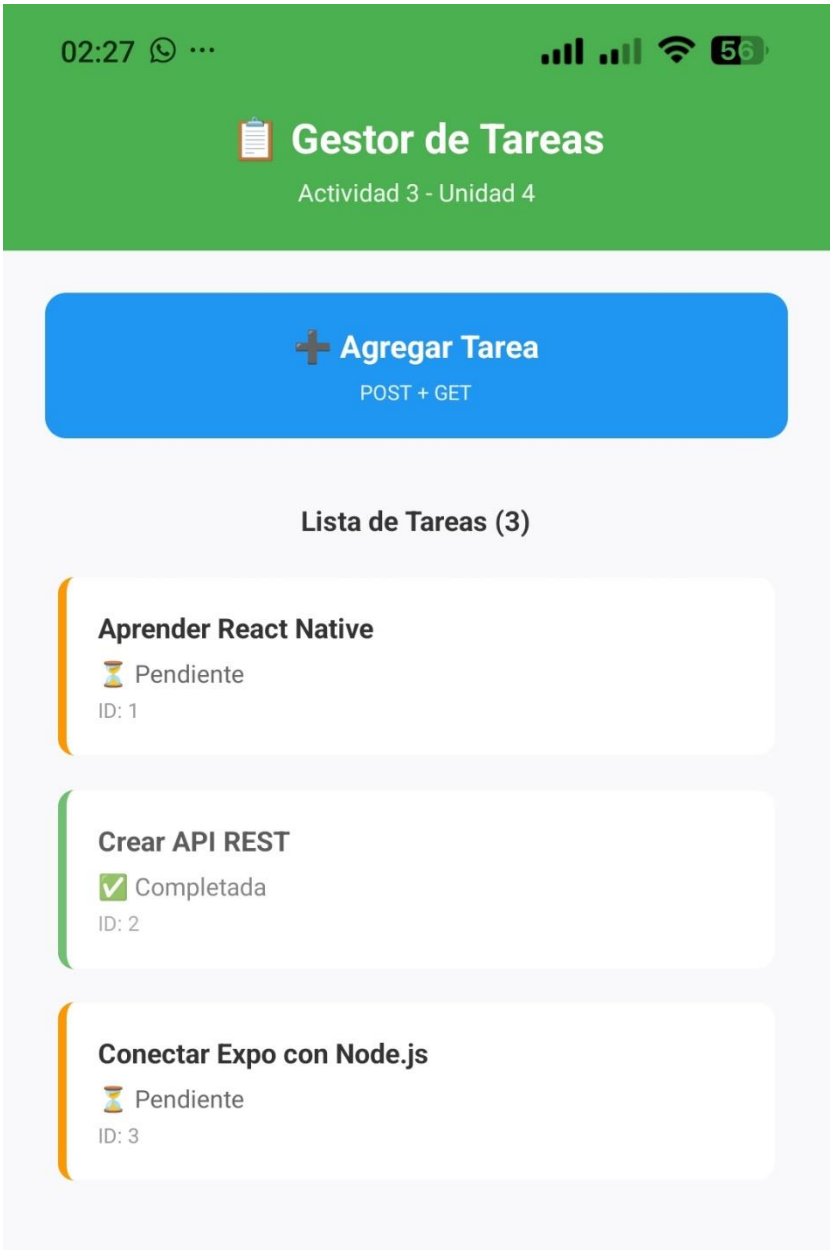
```
  titulo: { fontSize: 16, fontWeight: 'bold', color: '#333' },
  estado: { fontSize: 14, color: '#666', marginTop: 5 },
  id: { fontSize: 12, color: '#999', marginTop: 3 },
});

export default App;
```



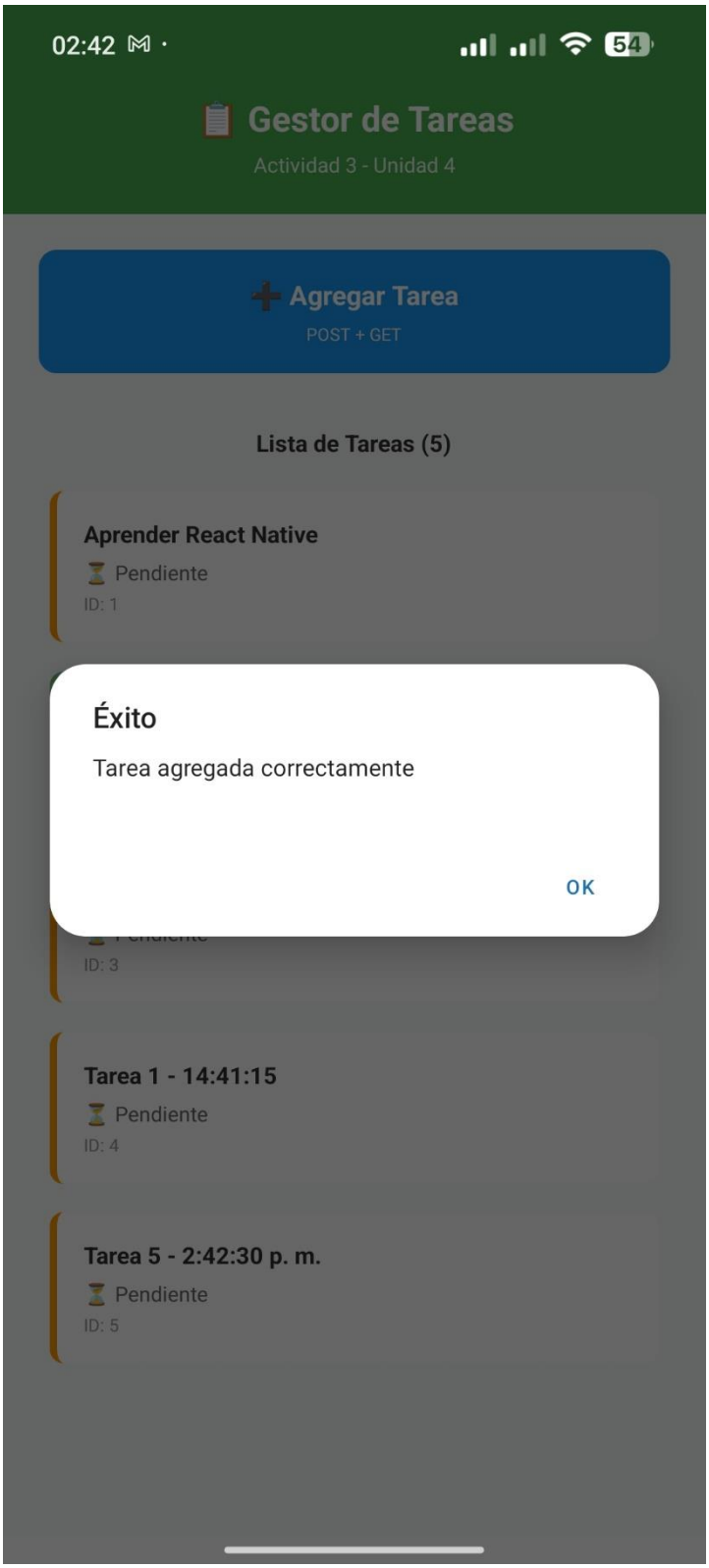
Editor Snack.expo.dev con código React Native implementando fetch GET, FlatList y función de doble fetch (POST + GET) para agregar tareas

c) Muestra el listado de tareas en el FlatList.

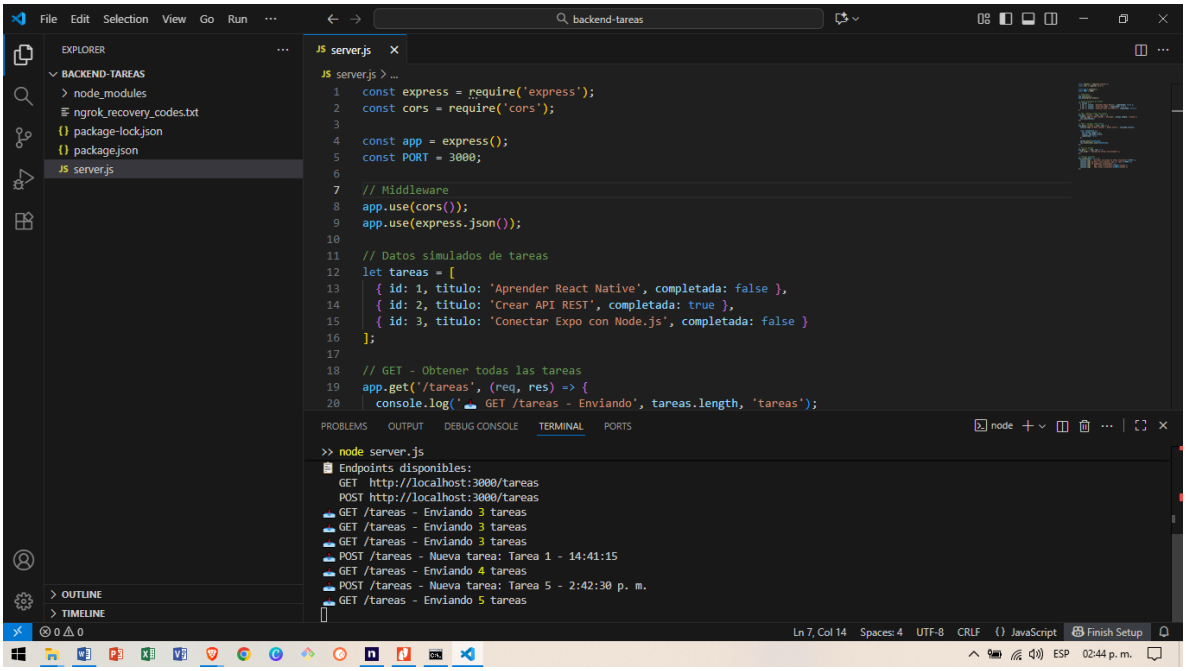


Aplicación Expo Go en dispositivo móvil mostrando FlatList con 3 tareas obtenidas mediante fetch GET, interfaz con header verde y botón azul

d) **Al hacer clic en el botón, realiza dos fetch: uno para agregar y otro para leer.**



Aplicación móvil después de hacer clic en 'Agregar Tarea', mostrando 4 tareas (3 iniciales + 1 nueva) demostrando el doble fetch: POST para crear y GET para actualizar



Código en VSCode con las actualizaciones del túnel y la app.

I. REFERENCIAS BIBLIOGRÁFICAS

crowdbotics. (2021, October 6). *How to Add a Search Bar in a FlatList in React Native Apps — Crowdbotics*. Crowdbotics. https://crowdbotics.com/posts/blog/add-search-bar-flatlist-react-native-apps/