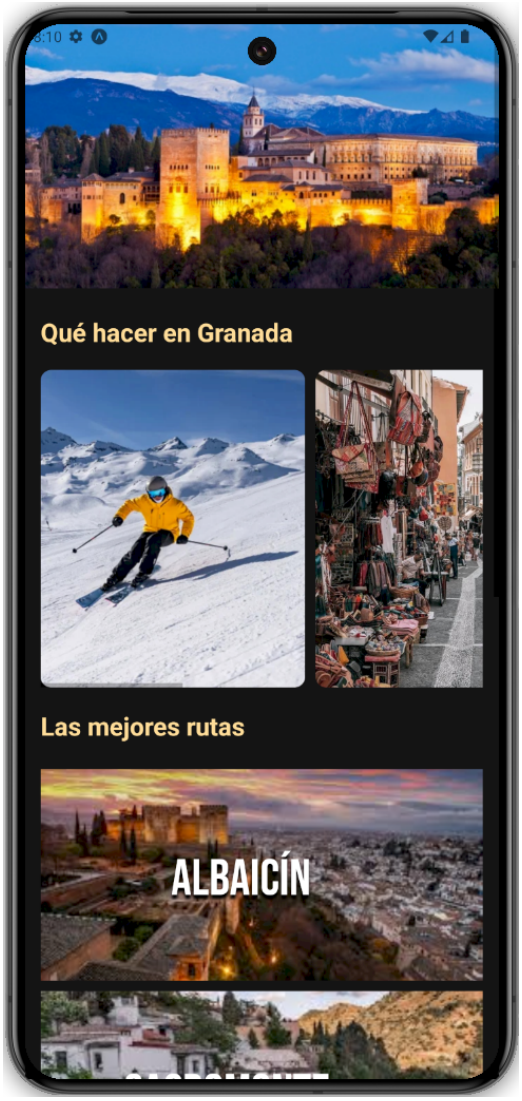
PROYECTO 1

QUÉ HACER EN GRANADA



- Número de personas: 1
- Tutoriales: 1 - 6
- Tipo de proyecto: ejercicio guiado con pasos

☉ ***¿Qué hace la aplicación?*** *El proyecto que vas a realizar consiste en la programación de una app que muestra información de Granada. Se trata simplemente de una aplicación informativa, en la que el usuario se limita a navegar por la interfaz. La aplicación tiene un carrusel de fotos y varias zonas con información sobre la ciudad. Algunas de las imágenes contienen texto en su interior. Al finalizar, pondremos un tema claro y un tema oscuro.*

1. Crea un proyecto llamado **Granada_app** y súbelo a GitHub
2. Copia las imágenes adjuntas a la carpeta **assets** del proyecto
3. Entra en **Google Fonts** y descarga el tipo de letra **Bebas Neue**
4. Descomprime en la carpeta **assests** el archivo **BebasNeue-Regular.ttf** de la descarga
5. Abre **App.tsx** y borra todo su contenido
6. En ese archivo, teclea **rnfs + Intro** para obtener una plantilla de inicio
7. Puedes borrar el **Text**, pero deja el **View**
8. Abre la terminal de **Visual Studio Code** e instala la librería **expo-image**
9. Crea una rama (y pasa a trabajar sobre ella) llamada **desarrollo_interfaz**
10. Haz un commit titulado “inicio proyecto”

☞ ***Punto de explicación:*** *Vamos a utilizar variables para guardar en ellas los principales colores usados por la aplicación. Haremos esto para que la modificación de colores sea sencilla, y así facilitar el mantenimiento de la app.*

11. Dentro de la función **App**, crea las siguientes constantes:

```
export default function App() {
  const COLOR_FONDO="#121212" // color de fondo de la pantalla
  const COLOR_TITULO="#ffdd99" // color para los títulos principales
  const COLOR_TEXTO_FOTO="#ffffff" // color para el texto que hay dentro de las fotos
  return (
    <View>
    </View>
  )
}
```

*Usarás esas constantes cada vez que aparezca uno de esos colores. Observa que al estar definidas dentro de la función **App**, no puedes usarlas fuera de ella, concretamente, en el objeto **styles**.*

12. Crea un estilo llamado **contenedorPrincipal** y ponle que sea un contenedor **flex** que ocupe toda la pantalla, y distribuya en columnas su contenido.
13. Pon al **View** principal una lista con dos estilos:
    - el estilo **contenedorPrincipal**
    - la regla de estilo **{backgroundColor: COLOR_FONDO}**

*Es necesario hacer esto porque como **COLOR_FONDO** solo “vive” en la función **App** y no puede ser usado en el objeto **styles***

14. Como la interfaz que vamos a mostrar es muy grande, dentro del **View** coloca un **ScrollView**, donde se diseñará todo lo que vendrá en los siguientes puntos.
15. Coloca la imagen **granada.jpg** de manera que su ancho sea toda la pantalla, su altura **250** y se escale hasta cubrir dicho tamaño, pero manteniendo su relación de aspecto
16. Haz un commit titulado “cabecera finalizada”

☑ ***Punto de comprobación:*** *Ejecuta la app y comprueba que se ve la imagen del fondo*

17. Tras la foto añade un **View** y ponle un estilo llamado **contenedorSecundario** con estas características: es un contenedor **flex,** tiene margen horizontal de **10** y padding de **5**
18. Añade un estilo llamado **titulo**, que tenga tamaño de texto **24**, negrita y añada un margen vertical de **20**
19. Dentro del contenedor secundario, añade un **Text** que ponga "¿Qué hacer en Granada?" y tenga una lista con estos dos estilos:
    - El estilo **titulo**
    - La regla **{color:COLOR_TITULO}**
20. Crea un estilo llamado **fotoCarrusel** que tenga anchura **250,** altura **300,** margen por la derecha **10** y borde del radio **10**
21. Tras el texto anterior, añade la imagen **actividad1.jpg** con el estilo **fotoCarrusel**, nuevamente escalándola hasta mantener su relación de aspecto
22. Tras la imagen anterior, añade las imágenes **actividad2.jpg** hasta **actividad5.jpg**, todas con el estilo **fotoCarrusel** y escalándolas hasta mantener su relación de aspecto

☑ ***Punto de comprobación:*** *Comprueba que todas las fotos se ven hacia abajo y la pantalla se puede desplazar hacia abajo*

23. Vamos a hacer un carrusel en el que las fotos se distribuyan de izquierda a derecha y podamos desplazar la pantalla en ese sentido. Para ello, simplemente encierra todas las fotos en un **ScrollView** que tenga su prop **horizontal** con valor **true**

☑ ***Punto de comprobación:*** *Comprueba que las fotos se muestran de izquierda a derecha y es posible desplazar la pantalla en ese sentido para verlas*

24. Haz un commit titulado "carrusel finalizado"
25. Tras el **ScrollView** del carrusel de fotos, añade un nuevo **Text** que ponga "Las mejores rutas" y tenga el mismo estilo del texto del punto 19
26. Añade un estilo llamado **fotoRuta** que tenga de ancho toda la pantalla, altura **200** y margen vertical de **5**
27. Tras el último **Text** que has puesto, añade las imágenes **mejores1.jpg, mejores2.jpg** y **mejores3.jpg**, todas con el estilo **fotoRuta** y escalándolas hasta mantener la relación de aspecto

*El componente **Image** no puede mostrar texto en su interior, pero el componente **ImageBackground** (incluido en la librería **expo-image**) si*

28. Cambia las etiquetas **Image** de "las mejores rutas" por etiquetas **ImageBackground**, dejando el resto de sus props iguales
29. Haz que el primer **ImageBackground** no se cierre dentro de su misma etiqueta, sino que tenga una etiqueta que lo abra y otra **</ImageBackground>** que lo cierre. La etiqueta que lo abre mantiene sus props.

```
1.  <ImageBackground // props omitidos >
2.  <ImageBackground>
```

30. Entre las etiquetas de inicio y cierre del **ImageBackground**, coloca un **Text** que diga "Albaicín"

☑ ***Punto de comprobación:*** *Comprueba que dentro de la primera imagen de "las mejores rutas" aparece la palabra Albaicín*

31. Al principio de la función **App,** justo después de crear las constantes, usa la función **useFonts** para crear un tipo de letra llamado **bebasNeue** usando el archivo **BebasNeue-Regular.ttf**
32. Crea un estilo llamado **textoFoto** que centre el texto horizontal y verticalmente, use el tipo de letra **bebasNeue,** tenga tamaño de letra **48** y una sombra de elevación **5**

33. Aplica al **Text** "Albaicín" una lista con dos estilos:
    - El estilo **textoFoto**
    - La regla **{color:COLOR_TEXTO_FOTO}**

☑ ***Punto de comprobación:*** *Comprueba que la palabra Albaicín aparece centrada en la primera foto, usa el tipo de letra Bebas Neue y tiene una sombra*

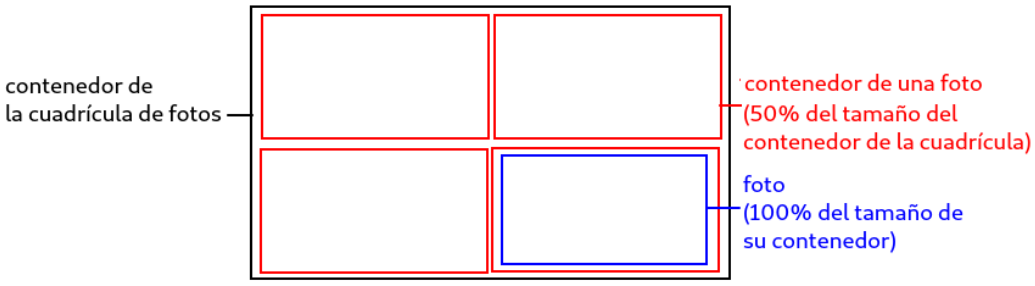
34. Repite los pasos anteriores para otras dos fotos de las rutas. En la segunda foto añade el texto "Sacromonte" y en la tercera "El centro". Sus estilos son los del paso 33

☑ ***Punto de comprobación:*** *Comprueba que las tres fotos de "las mejores rutas" tienen su texto centrado, con tipo de letra diferente y sombra*

35. Haz un commit titulado "mejores rutas finalizadas"
36. Tras las fotos anteriores añade un **Text** que ponga "Los mejores alojamientos", con el estilo de letra del punto 19

☞ ***Punto de explicación:*** *En los siguientes puntos vamos a hacer la cuadrícula de fotos y queremos conseguir que se distribuyan en cuadrícula de forma automática (sin dar tamaños "a mano"), ocupando cada foto la mitad de la pantalla.*

*La idea es meter cada **foto** en un **View**, de forma que la **foto** ocupe el 100% del tamaño de ese **View**, y a su vez, el **View** ocupe el 50% de su **contenedor**.*



Es necesario hacer este "truco" porque el tamaño de las fotos no funciona bien con flex, y lo mejor es quitarnos de problemas dándoles un tamaño del 100% de su contenedor, y usar el tamaño de dicho contenedor para poder dar el tamaño deseado a la foto.

37. Crea un estilo llamado **contenedorFotosAlojamiento**, que sea un contenedor flex, distribuya en filas sus elementos, dejando una separación de **5** entre las filas y columnas. Además, repartirá las fotos pegándolas a los bordes de la pantalla y tendrá un margen inferior de **15**
38. Crea un estilo llamado **fotoAlojamiento**, que defina una anchura y altura igual al 100% de su contenedor
39. Crea un estilo llamado **contenedorAlojamiento**, que tenga un ancho del 49% de su contenedor (el 1% restante se dejará como espacio para separar las fotos) y la regla **aspectRatio : 1**

```
1.  contenedorAlojamiento: {
2.    width: "49%",
3.    aspectRatio: 1,
4.  },
```

> *La regla **aspectRatio:1** permite calcular la altura del **View** automáticamente para que se mantenga la relación de aspecto de la foto*

40. Tras el **Text** "los mejores alojamientos" añade un **View** con el estilo **contenedorFotosAlojamiento**
41. Dentro del **View** anterior, coloca un **View** con estilo **contenedorAlojamiento** y dentro de él, la imagen **alojamiento1.jpg** con estilo **fotoAlojamiento**, escalándola hasta mantener la relación de aspecto
42. Repite los pasos anteriores con el resto de fotos de alojamiento

> ☑ ***Punto de comprobación:** Comprueba que se muestra correctamente la cuadrícula con las 4 fotos de alojamientos*

43. Haz un commit titulado "alojamientos finalizado"

> ☞ ***Punto de explicación:** Vamos a ver cómo las variables que hemos creado con los colores nos permiten cambiar fácilmente la apariencia de la app*

44. Modifica las variables de los colores con las siguientes combinaciones, guardando los cambios tras cada una y observando el resultado.

COLOR_FONDO	COLOR_TITULO	COLOR_TEXTO_TITULO
#F5F5F5	#2C3E50	#FFFFFF
#FFF8F0	8C1C1E	#FFFAFA
#E8F5E9	#1B5E20	#FFFFFF
#FEF9EF	#DB5461	#FFFFFF

> ☞ ***Punto de explicación**: Si te das cuenta, cada combinación de colores es un **tema** que podemos usar en app. Vamos a crear dos objetos con dos combinaciones de colores, una para el tema claro (**light)** y otra para el tema oscuro **(dark)**. Usaremos la función **useColorScheme** para obtener el tema del dispositivo*

45. En el proyecto, crea una carpeta llamada **themes** y dentro de ella un archivo llamado **Temas.ts**

46. En el archivo **Temas.ts** crea (y exporta) un objeto llamado **TEMA_OSCURO** con los tres colores que definimos en **App**. Ese objeto tendrá como claves: **colorFondo, colorTitulo, colorTextoFoto** y **logo**. Esta última guardará la imagen que se mostrará en el tema oscuro, y por tanto, contendrá la llamada a la función **require** para cargar la foto **granada_dark.jpg**

```
const TEMA_OSCURO = {
    colorFondo:"#121212",
    colorTitulo:"#ffdd99",
    colorTextoFoto: "#ffffff",
    logo:require("../assets/granada_dark.jpg")
}
export {TEMA_OSCURO}
```

47. En el mismo archivo, a continuación del objeto **TEMA_OSCURO,** crea y exporta otro objeto llamado **TEMA_CLARO** que tenga los mismos campos, y sus valores sean la combinación de colores del punto 44 que más te haya gustado. En el campo **logo**, tendrás que poner **granada_light**
48. En la función **App**, borra las variables de los colores (porque ahora las cogeremos del tema)
49. Abre el archivo **app.json** y modifica el campo **userInterfaceStyle** para que sea **automatic**

Por defecto, las apps siempre usan el tema claro (por motivos de eficiencia). Con la modificación que has hecho, se podrá detectar el tema claro del móvil

50. Al principio de la función **App**, llama a la función **useColorScheme** (no necesita parámetros de entrada) y recoge en una variable llamada **temaActivo** el string que nos devuelve.

*La función **useColorScheme()** nos devuelve un **string** que vale **light** si el tema del dispositivo es claro, y **dark** si es oscuro*

51. Crea una variable llamada **tema** que sea igual a **TEMA_OSCURO** si **temaActivo** es **dark,** o **TEMA_CLARO** si **temaActivo** es **light**
52. Modifica los estilos que usaban las variables de colores (y que ahora dan error) para que usen los colores que hay dentro de la variable **tema**
53. Modifica la imagen de la cabecera para que el nombre del archivo se tome de **tema.logo**

☑ ***Punto de comprobación:*** *Comprueba que al cambiar de tema (se recomienda probarlo en un móvil real porque es más fácil cambiar de tema) cambia el esquema de colores*

54. **Voluntario:** Modifica **Temas.ts** para que no haya dos variables, sino un solo objeto llamado **Temas** cuyas claves sean los string **light** y **dark** y sus valores los objetos con los colores de cada tema. No olvides exportar el objeto **Temas**
55. **Voluntario:** Modifica **App** para que la variable **tema** se rellene directamente a partir del objeto **Temas**, consultando sobre él el valor del campo **temaActivo**