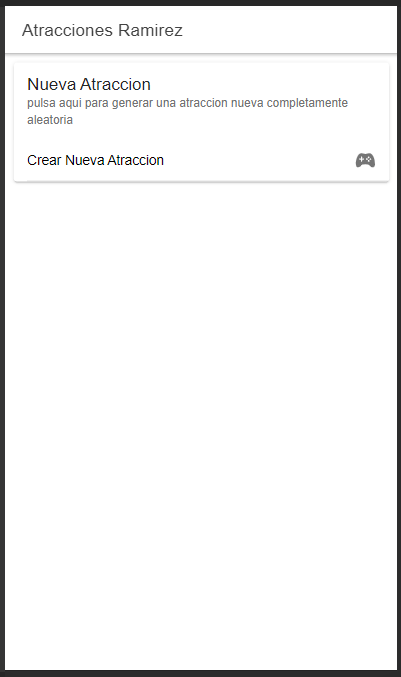
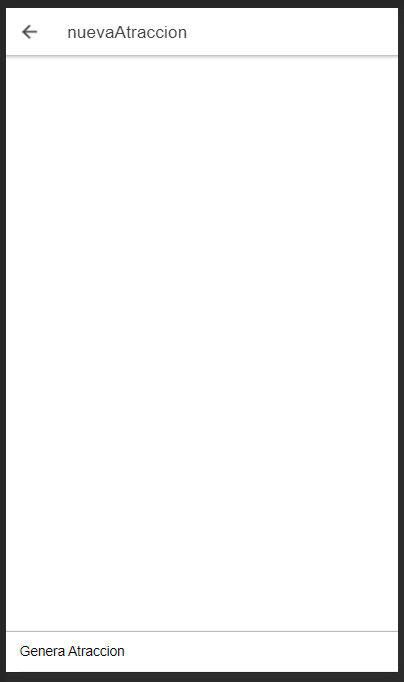
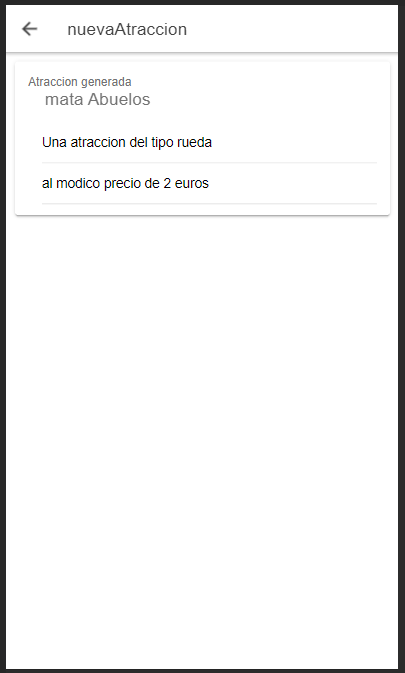
Vamos a hacer un examen por partes. En este enunciado aparecen los botones en el orden en que pueden programarse, es decir, a medida que planteo preguntas aparecen los componentes necesarios.

1º) Vamos a hacer la aplicación de un parque de atracciones.

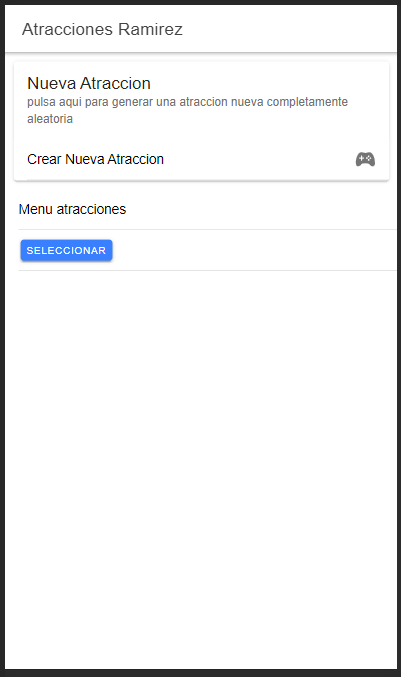
Aquí puedes ver la primera pregunta. Es una card, puedes hacer click sobre “crear Nueva Atraccion” y te lleva a otra página

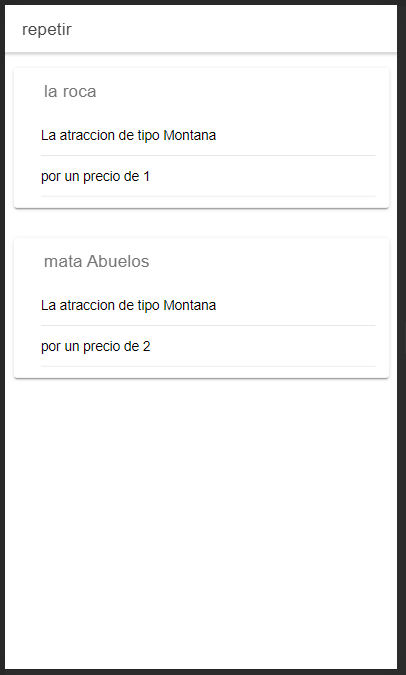


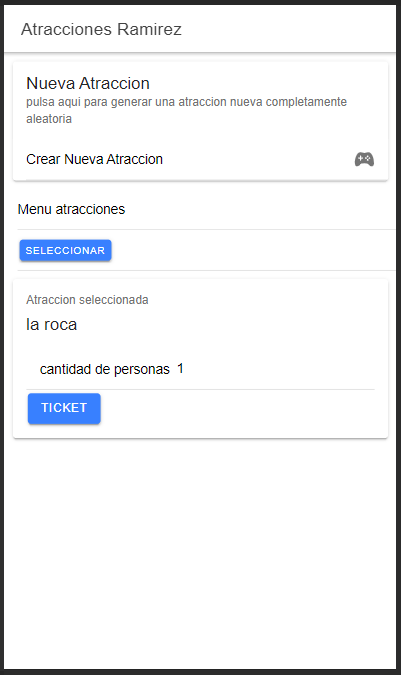
Observa como la pagina ofrece la posibilidad, en pantalla, de volver a la página anterior. Abajo tenéis un elemento que podéis clickar y que aleatoriamente genera una Atracción. Para conseguir esto debes usar el servicio que se os pasa “genera-atraccion-service” y las clases de “core/model” “Atraccion” y “TipoAtraccion”. Lo que debe mostrar al pulsar sobre “genera Atraccion” es:

Observa como desaparece el elemento clicable de la parte inferior de la pantalla. Observa que en el campo “tipo de atracción” aparece el nombre del tipo y no su número.

Esta pregunta vale 1 punto

2º) La siguiente pregunta se basa en la primera para poder hacerse. Observa cómo cambia la página principal.

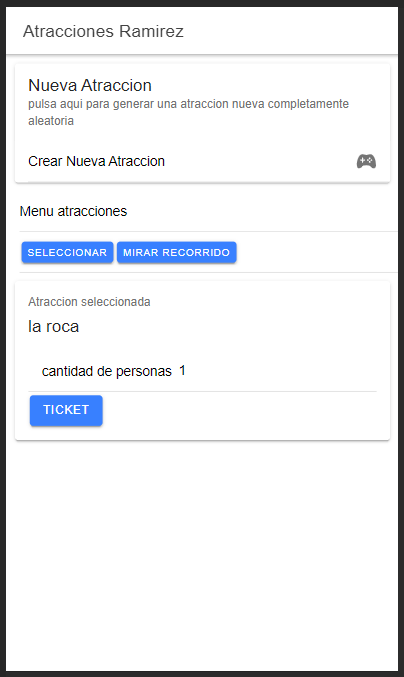
Imaginamos que hemos creado un par de nuevas atracciones. Tendremos, por lo tanto, una lista de nuevas atracciones. Al pulsar sobre el botón “seleccionar” debe salir algo como esto:

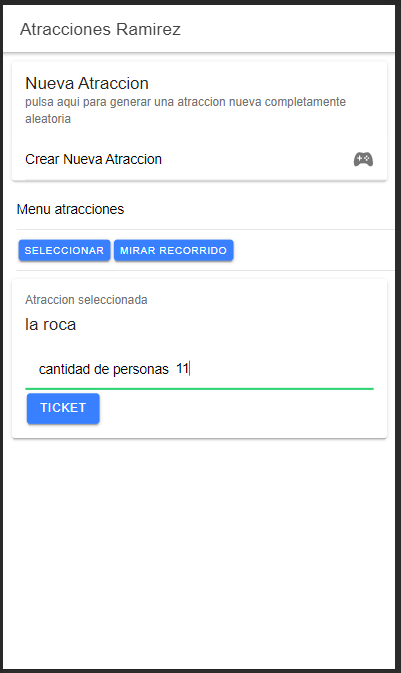
Una lista de las atracciones disponibles. Observa que no hay ningún control de navegación. No se puede ir atrás ni hay botones. Hacer clic sobre uno de los elementos significa que lo has seleccionado y esa acción te lleva a la página principal, de nuevo, que ahora muestra este aspecto.

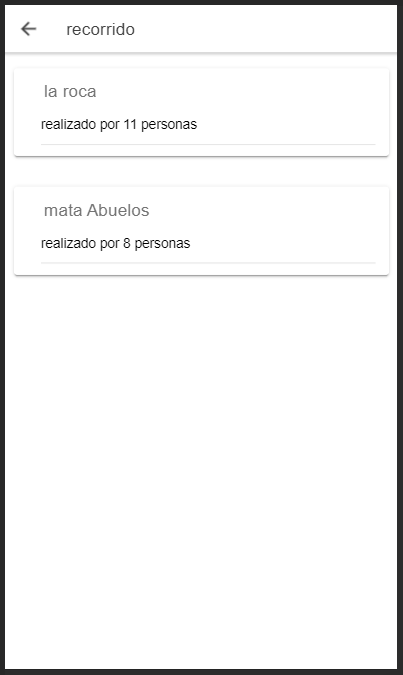
Ahora tenemos una página que añade la atracción seleccionada en este momento. En la misma podemos escribir el número de personas que queramos, como mínimo 1.

EL botón ticket **NO HACE NADA** en esta pregunta

Esta pregunta son dos puntos.

3º) Aquí empezamos viendo que hay un nuevo botón en la página principal: “Mirar Recorrido”

Comenzamos haciendo la implementación del botón “ticket” que genera un viaje. Para poder saber cómo es debes usar las clases que se encuentran en el archivo “recorrido.ts” en “core/model”. Vamos a entender la clase “Recorrido” como un conjunto de viajes. Cuando haces clic sobre el botón “Ticket” se debe generar un nuevo “Viaje” del “Recorrido”.

El nuevo botón “Mirar recorrido” nos debe llevar a una nueva página donde veremos algo así:

Una lista de los Viajes que se han hecho en el recorrido. Observa que hay un botón de retroceso en la cabecera, único elemento clicable, que permite la navegación hacia atrás.

Esta pregunta vale 1.5 puntos.

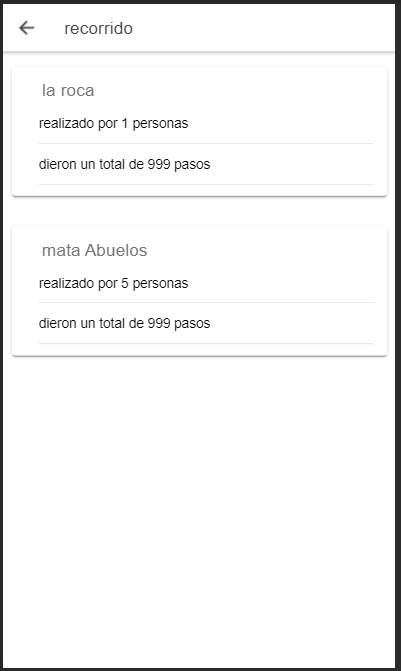
Hasta aquí, si todo está bien hecho, tendrás un 4.5. Supongo que con la práctica final y tu actitud en clase, normalmente excelente, no tendrás problemas para aprobar el módulo, pero siempre te quedará la cosa de que no aprobaste el examen, porque para aprobar este examen hay que saber algo sobre nativos, mocks, specs y promesas.

Las siguientes preguntas son independientes y se puede contestar una de ellas o las dos.

4º) En el proyecto que se te pasa está instalado el plugin y native “stepcounter”. Hemos tenido que añadir la plataforma android para que permitiese su instalación. Sin embargo, como sabéis, no nos interesa probar la app en móvil, en esta etapa del desarrollo, por lo tanto usamos un mock que permite la depuración en el navegador.

Se te pasa el mock, es decir tendrás el archivo index.ts con el contenido del native mencionado.

Será necesario que sepas configurar la inyección para poder usarlo en un servicio.

Debes crear un servicio que use el native “StepCounter” con un método que se llame “getSteps()” que te da un valor, aleatorio, entre 100 y 600 (ambos incluidos). Para obtener este valor debes usar el método “getTodayStepCount():Promise<number|any>”. Debes utilizar este valor para poder visualizarlo en la page que llamamos al presionar el botón “mirar recorrido”. Ahora deberá aparecer un valor aleatorio como se ve en la foto.

Esta pregunta vale 1.5 puntos

4ºa) La segunda parte de la pregunta te pide que configures el archivo spec de este servicio para comprobar que el método funciona “getSteps()” funciona, es decir, que da valores entre 100 y 600, incluidos estos dos. Y estos son 0.5 puntos (y yo lo haría al final, si me sobra media hora)

5º) Quedan por dar 3.5 puntos que son los que valen esta pregunta. Existe otro nativo que también tienes instalado ya. Ya está configurado en el module, así que puedes usarlo libremente. Se llama “Storage”. Este nativo permite guardar elementos usando un mecanismo de persistencia invisible al programador. Para almacenar algo hay que hacerlo insertando elementos de tipo “clave/valor” en un mapa. Como es un nativo, funciona con promesas.

Los métodos que podemos usar son

set(key: string, value: any): Promise<any>

Con un valor en key ya existente sustituye el antiguo value por este nuevo.

keys(): Promise<string[]>

Que te entrega una promesa con las keys almacenadas en el mapa

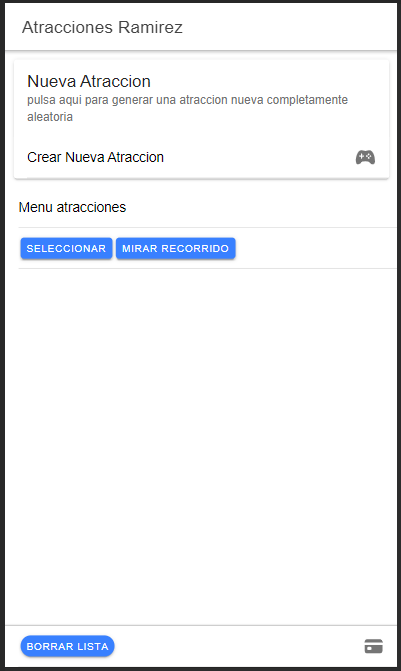
get(key: string): Promise<any>;

Que te entrega una promesa del tipo any que será el valor asociado al parámetro key, en caso de que exista

clear(): Promise<void>; borra todos los elementos del mapa

Con estos métodos ya se puede hacer la pregunta. Siéntete libre de usar otros si los necesitas.

Debes implementar el servicio “atracciones-service” para que almacene las atracciones que se generan aleatoriamente en el elemento “crear atraccion”. Esta pregunta está bien contestada si, al reiniciar la aplicación, las atracciones siguen estando disponibles para el botón “seleccionar” de la pagina “home”, es decir, funciona la persistencia. Esto vale 3 puntos.

5ºa) Una última cosa. Implementa el botón “borrar lista” que elimina todas las atracciones almacenadas en el mapa anterior.

**Aunque sea un native, Storage no necesita de plugin para funcionar. Por lo tanto no necesitas un mock para usarlo.**

Veamos

1º) 1 punto

2º) 2 puntos

3º) 1,5 puntos

4º) 1,5 + 0,5 puntos

5º) 3 + 0,5 puntos

El orden propuesto es el aconsejado. Lo que debes entregar es un archivo comprimido, que contenga tu nombre en el suyo y que contenga la carpeta src de tu proyecto terminado. Asegúrate de no estropear cosas que ya te funcionan, es decir, para cada pregunta crea una rama que contenga las preguntas anteriores y la que estas contestando en ese momento. Entrega la carpeta src de la rama en que funcione todo. Por ejemplo si has intentado la rama con la pregunta cinco pero no has podido terminarla, entrega la rama con la respuesta cuatro (si funciona). Tranquilos y buena suerte.

Comenzamos importando el proyecto almacenado en git.

https://github.com/ionicharnina2021/exaAluMarzo21

Ejecutalo en android. Si te funciona (mirar los logs, debe aparecer que hay objeto Storage y StepCounter) puedes empezar el examen, si no, llama al profesor.