EcoHarmony Park - App

TDD - PRUEBAS A EJECUTAR

Guia de pruebas a desarrollar y ejecutar para la implementación de la funcionalidad de inscribirse a actividad

ECOHARMONY PARK

ISW - GRUPO 4

DOCENTES:

- Ing. Laura Covaro
- Ing. Mickaela Crespo
- Ing. Constanza Garnero

INTEGRANTES:

- 86657 - Juan Salvador Barbera.
- 90297 - Francisco Cornejo.
- 83009 - Mateo Romero Plaza.
- 85291 - Nicolás Ranalli.

US A DESARROLLAR: Inscribirme a Actividad

Inscribirme a actividad **COMO visitante QUIERO inscribirme a una actividad PARA reservar mi lugar en la misma.**	3

Criterios de Aceptación:

- Debe requerir seleccionar una actividad del conjunto de actividades de la lista de "Tirolesa", "Safari", "Palestra" y "Jardinería", siempre y cuando tengan cupos disponibles para el horario seleccionado
- Debe requerir seleccionar el horario dentro de los disponibles
- Debe indicar la cantidad de personas que participaran de la actividad
- Para cada persona que participa, debe ingresar los datos del visitante: nombre, DNI, edad y talla de vestimenta si la actividad lo demanda
- Debe requerir aceptar los términos y condiciones específicos de la actividad en la que participarán.

Pruebas de usuario:

- Probar inscribirse a una actividad del listado que poseen cupos disponibles, seleccionando un horario, ingresando los datos del visitante (nombre, DNI, edad, talla de la vestimenta si la actividad lo requiere) y aceptando los términos y condiciones (pasa)
- Probar inscribirse a una actividad que no tiene cupo para el horario seleccionado (falla)
- Probar inscribirse a una actividad sin ingresar talle de vestimenta porque la actividad no lo requiere (pasa)
- Probar inscribirse a una actividad seleccionando un horario en el cual el parque está cerrado o la actividad no está disponible (falla)
- Probar inscribirse a una actividad sin aceptar los términos y condiciones de la actividad (falla)
- Probar inscribirse a una actividad sin ingresar el talle de la vestimenta requerido por la actividad (falla)

CÓDIGO:

Función principal para inscribirse a una actividad

```
func inscribirseAActividad(fechaSeleccionada, actividad,
horario, cantCuposSeleccionados, visitantes[],
terminosYCondiciones)
{
if not validarFecha(fechaSeleccionada){
throw Exception FechaSeleccionadaInvalida('La actividad no se
encuentra disponible para la fecha seleccionada.')
}

if not validarDisponibilidadCupos(fechaSeleccionada, horario,
cantCuposSeleccionados, actividad){
     throw Exception CuposNoDisponibles('No existen cupos
disponibles para dicha actividad en el horario seleccionado')
}

for visitante in visitantes{
if not validarDatosMinimos(actividad, visitante) {
     // le paso la actividad para que sepa que tiene que
corroborar de cada act
throw Exception DatosVisitanteInvalidos('El visitante no cumple
con los requerimientos para la actividad.')
}
if actividad.requiereTalle and visitante.talle == Null {
     throw Exception TalleRequerido('La actividad seleccionada
requiere que se indique el talle de vestimenta.')
}
}

if not terminosYCondiciones {
throw Exception TerminosYCondicionesFalse('Debe aceptar los
términos y condiciones de la actividad.')
}
```

```
//resultados
actividad.cuposDisponibles = actividad.cuposDisponibles -
cantCupos

//Se crea la inscripción a la actividad
new Inscripcion(fechaSeleccionada, actividad, horario,
cantCuposSeleccionados, visitantes[])

return 'Su inscripción a la actividad se ha realizado con
éxito.'
}
```

Funciones auxiliares dentro de la principal

```
func validarFecha(fechaSeleccionada){
     if fechaSeleccionada >= sysdate.fechaActual() and
fechaSeleccionada.dia not in ('Martes', 'Jueves'){
         return True
      }
      else {
          return False
      }
}
```

```
func validarDisponibilidadCupos(fechaSeleccionada, horario,
cantCuposSeleccionados, actividad) {
    // Supongamos que actividad tiene una estructura como:
    // actividad.cuposDisponiblesPorHorario = {
    //     '2025-06-23': { '14:00': 3, '16:00': 5 }
    // }

    fechaStr = fechaSeleccionada.toString() // e.g. '2025-06-23'

    if fechaStr in actividad.cuposDisponiblesPorHorario {
        if horario in
actividad.cuposDisponiblesPorHorario[fechaStr] {
```

```
            cuposDisponibles =
actividad.cuposDisponiblesPorHorario[fechaStr][horario]

            if cuposDisponibles >= cantCuposSeleccionados {
                return True
            }
        }
    }

    return False
}
```

```
func validarDatosMinimos(actividad, visitante) {
    // Supongamos que cada actividad tiene requisitos como:
    // actividad.edadMinima, actividad.edadMaxima
    // actividad.alturaMinima, actividad.alturaMaxima (opcional)

    if visitante.edad < actividad.edadMinima or visitante.edad >
actividad.edadMaxima {
        return False
    }

    // Si la actividad requiere altura, la validamos también
    if actividad.tieneRequisitoAltura {
        if visitante.altura < actividad.alturaMinima or
visitante.altura > actividad.alturaMaxima {
            return False
        }
    }

    return True
}
```

TESTS

```
func test_inscribirseAActividad_success(){
// se entiende que la fecha actual es Lunes 24/6/2025 y hay
cupos para tirolesa en el horario de 14:00

     fechaSeleccionada = new Date() // crea la fecha y es igual
a Lunes 24/6/2025
     actividad = new Actividad(nombre = 'Tirolesa', edadMinima
= 10, edadMaxima = 90,  alturaMinima = null, requiereTalle =
True, alturaMaxima = null, tieneRequisitoAltura = False)
     horario = '14:00'
     cantCuposSeleccionados = 1
     visitantes = [Visitante(nombre= 'Moria Casan', edad = 80,
dni = 12345678, talle = 'S')]
     terminosYCondiciones = True

      assert.equals(inscribirseAActividad(fechaSeleccionada,
actividad, horario, cantCuposSeleccionados, visitantes[],
terminosYCondiciones),'Su inscripción a la actividad se ha
realizado con éxito')
}
```

```
func test_inscribirseAActividad_TYC_fail(){
// se entiende que la fecha actual es Lunes 23/6/2025 y hay
cupos para tirolesa en el horario de 14:00

     fechaSeleccionada = new Date() // crea la fecha y es igual
a Lunes 23/6/2025
     actividad = new Actividad(nombre = 'Tirolesa', edadMinima =
10, edadMaxima = 90,  alturaMinima = null, requiereTalle = True,
alturaMaxima = null, tieneRequisitoAltura = False)
     horario = '14:00'
     cantCuposSeleccionados = 1
     visitantes = [Visitante(nombre= 'Moria Casan', edad = 80,
dni = 12345678, talle = 'S')]
```

```
     terminosYCondiciones = False

      try {
          inscribirseAActividad(fechaSeleccionada, actividad,
horario, cantCuposSeleccionados, visitantes[],
terminosYCondiciones)
      }
      catch (ExceptionType TerminosYCondicionesFalse ex){
           assert.equals(ex,'Debe aceptar los términos y
condiciones de la actividad.')
    }
}
```

```
func test_inscribirseAActividad_Date_fail(){
     fechaSeleccionada = new Date('18/06/2022') //Fecha
anterior a la actual
     actividad = new Actividad(nombre = 'Tirolesa', edadMinima
= 10, edadMaxima = 90,  alturaMinima = null, requiereTalle =
True, alturaMaxima = null, tieneRequisitoAltura = False)
     horario = '14:00'
     cantCuposSeleccionados = 1
     visitantes = [Visitante(nombre= 'Moria Casan', edad = 80,
dni = 12345678, talle = 'S')]
     terminosYCondiciones = True

      try {
          inscribirseAActividad(fechaSeleccionada, actividad,
horario, cantCuposSeleccionados, visitantes[],
terminosYCondiciones)
      }
      catch (ExceptionType FechaSeleccionadaInvalida ex){
             assert.equals(ex,'La actividad no se encuentra
disponible para la fecha seleccionada.')
      }
}
```

```
func test_inscribirseAActividad_sin_cupo_fail() {
    // Asumimos que para este horario y fecha hay menos de 5
cupos disponibles
    fechaSeleccionada = new Date('2025-06-23')
    actividad = new Actividad(nombre = 'Tirolesa', edadMinima =
10, edadMaxima = 90,  alturaMinima = null, requiereTalle = True,
alturaMaxima = null, tieneRequisitoAltura = False)
    horario = '14:00'
    cantCuposSeleccionados = 5
    terminosYCondiciones = True

    visitantes = [
        Visitante(nombre='Moria Casan', edad=80, dni=11111111,
talle='S'),
        Visitante(nombre='Susana Giménez', edad=79,
dni=22222222, talle='M'),
        Visitante(nombre='Mirtha Legrand', edad=89,
dni=33333333, talle='S'),
        Visitante(nombre='Carmen Barbieri', edad=68,
dni=44444444, talle='L'),
        Visitante(nombre='Nacha Guevara', edad=83, dni=55555555,
talle='M')
    ]

    terminosYCondiciones = true

    try {
        inscribirseAActividad(fechaSeleccionada, actividad,
horario, cantCuposSeleccionados, visitantes,
terminosYCondiciones)
    } catch (ExceptionType CuposNoDisponibles ex) {
        assert.equals(ex, 'No existen cupos disponibles para
dicha actividad en el horario seleccionado')
    }
}
```

```
func test_inscribirseAActividad_sin_talle_y_es_requerido_fail()
{
    fechaSeleccionada = new Date('2025-06-23')
    actividad = new Actividad(nombre = 'Tirolesa', edadMinima =
10, edadMaxima = 90,  alturaMinima = null, requiereTalle = True,
alturaMaxima = null, tieneRequisitoAltura = False)
    horario = '14:00'
    cantCuposSeleccionados = 1
    visitantes = [Visitante(nombre='Franco Colapinto"', edad=24,
dni=12345678, talle=null)]
    terminosYCondiciones = True

    try {
        inscribirseAActividad(fechaSeleccionada, actividad,
horario, cantCuposSeleccionados, visitantes,
terminosYCondiciones)
    } catch (ExceptionType TalleRequerido ex) {
        assert.equals(ex, 'La actividad seleccionada requiere
que se indique el talle de vestimenta.')
    }
}
```

```
func test_inscribirseAActividad_no_cumple_requerimientos_fail(){
    fechaSeleccionada = new Date('2025-06-23')
    actividad = new Actividad(nombre = 'Tirolesa', edadMinima =
10, edadMaxima = 90,  alturaMinima = 1.75 , requiereTalle =
False, alturaMaxima = 2.10, tieneRequisitoAltura = True)
    horario = '14:00'
    cantCuposSeleccionados = 1
    visitantes = [Visitante(nombre='Franco Colapinto"', edad=24,
dni=12345678, altura=1.60)]
    terminosYCondiciones = True
    try{
        inscribirseAActividad(fechaSeleccionada, actividad,
horario, cantCuposSeleccionados, visitantes,
```

```
terminosYCondiciones)
    }
    catch (ExceptionType DatosVisitanteInvalidos ex){
        assert.equals(ex, 'El visitante no cumple con los
requerimientos para la actividad.')
    }
}
```

TESTS FUNCIONES AUXILIARES

```
func test_validarFecha_success() {
   fechaSeleccionada = new Date('2025-06-23') // crea la fecha y
es igual a Lunes 23/6/2025

   assert.equals(validarFecha(fechaSeleccionada), True)
}
```

```
func test_validarFecha_fecha_pasada_fail() {
     fechaSeleccionada = new Date('2025-06-22')
     sysdate.setFechaActual(new Date('2025-06-23'))

     assert.equals(validarFecha(fechaSeleccionada), False)
}
```

```
func test_validarFecha_fail(){
     fechaSeleccionada = new Date('2025-06-24') // crea la
fecha y es igual a Martes 24/6/2025
     assert.equals(validarFecha(fechaSeleccionada), False)
}
```

```
func test_validarFecha_formato_incorrecto_fail(){
     fechaSeleccionada = 'Lunes 2025-06-23'
     try {
        validarFecha(fechaSeleccionada)
     }
     catch (ExceptionType FechaTipoDate ex) {
        assert.equals(ex, 'El formato y tipo de fecha no
corresponde al del tipo Date')
     }
}
```

```
func test_validarDisponibilidadCupos_success(){
     fechaSeleccionada = new Date('2025-06-23')
     cantCuposSeleccionados = 2
     horario = '14:00'
     actividad.cuposDisponiblePorHorario = {
          '2025-06-23': {'14:00': 3, '16:00': 5}
     }

     assert.equals(validarDisponiblidadCupos(fechaSeleccionada,
cantCuposSeleccionados, horario, actividad), True)
}
```

```
func test_validarDisponibilidadCupos_sin_cupos_fail() {
     actividad.cuposDisponiblePorHorario = {
     '2025-06-23': {'16:00': 2}
     }

     fechaSeleccionada = new Date('2025-06-23')
     horario = '16:00'
     cantCuposSeleccionados = 4

     assert.equals(validarDisponibilidadCupos(fechaSeleccionada,
cantCuposSeleccionados, horario, actividad), False)
}
```

```
func test_validarDisponibilidadCupos_fecha_incorrecta_fail(){
     fechaSeleccionada = 'Martes 24-06-2025'
     cantCuposSeleccionados = 2
     horario = '14:00'
     actividad.cuposDisponiblePorHorario = {
          '2025-06-23': {'14:00': 3, '16:00': 5}
     }
     try{
         validarDisponibilidadCupos(fechaSeleccionada,
cantCuposSeleccionados, horario, actividad)
    }
    catch (ExceptionType FormatoFechaIncorrecto ex) {
        assert.equals(ex, 'El formato y tipo de fecha no
corresponde al del tipo Date')
    }
}
```

```
func test_validarDatos_succes(){
     actividad = new Actividad(nombre = 'Tirolesa', edadMinima
= 10, edadMaxima = 90,  alturaMinima = null, requiereTalle =
True, alturaMaxima = null)
     visitante = Visitante(nombre='Nacha Guevara', edad=83,
dni=55555555, talle='M')

     assert.equals(validarDatos(actividad , visitante), True)
}
```

```
func test_validarDatos_fail(){
     actividad = new Actividad(nombre = 'Tirolesa', edadMinima
= 10, edadMaxima = 90,  alturaMinima = null, requiereTalle =
True, alturaMaxima = null, tieneRequisitosAltura = False)
     visitante = Visitante(nombre='Mirtha Legrand', edad=100,
dni=55555555, talle='M')

     assert.equals(validarDatos(actividad , visitante), False)
}
```

```
func test_validarDatos_altura_int_fail(){
     actividad = new Actividad(nombre = 'Palestra', edadMinima
= 10, edadMaxima = 90,  alturaMinima = null, requiereTalle =
True, alturaMaxima = null, tieneRequisitosAltura = True)
     visitante = Visitante(nombre='Mirtha Legrand', edad=100,
dni=55555555, talle='L', altura = 2)

     try{
         validarDatos(actividad,visitante)
     }
     catch (ExceptionType FormatoAltura ex){
         assert.equals(ex, 'El formato de la altura no
corresponde al del tipo Float')
     }
}
```

ToDO: Todas las pruebas que al ejecutarse den error, deberían de implicar que se actualice el código correspondiente, contemplando todas las excepciones pertinentes.