UNIVERSIDAD ANTONIO NARIÑO

ELECTIVA III

M.Sc. Claudia Hernández

MANEJO DE RESERVAS DEL COSTA BRAVA

Leydy J. Pachón Alarcón : *11161223477*
Carlos A. Garzón Trujillo : *11161217454*

June 6, 2016

Contents

1 INTRODUCCIÓN

A continuación se expone el proyecto desarrollado y el seguimiento que se llevó a cabo para ejecutar pruebas, control de versiones y administración de defectos.

2 PLANTEAMIENTO DEL PROBLEMA HOTEL

El hotel Costa Brava tiene 50 habitaciones disponibles para prestar servicio, de las cuales 8 son de clase suite presidencial y 42 de clase Premium. Costa Brava cuenta con una zona de piscinas y con dos zonas de jardines ubicadas en las partes laterales del hotel.

Las suites presidenciales están ubicadas en filas de 4 habitaciones, separadas en el medio por la zona de piscinas. De este modo se tiene habitaciones suite presidencial con vista a la piscina o con vista al jardín.

Las Premium están ubicadas en filas de seis habitaciones, de cada fila hay tres habitaciones a cada lado de la zona de piscinas. De este modo se tiene habitaciones Premium con vista a la piscina, al jardín o sin vista privilegiada.

Este es el esquema del Costa Brava:

Figure 1: Estructura del Hotel Costa Brava

Cuando un huésped llega a solicitar una habitación este indica sus datos personales

y sus preferencias respecto a la ubicación de la habitación.

A Costa Brava le interesa el nombre y el número de cédula del cliente, como única identificación. Además, si hay disponibilidad, el huésped puede elegir:

Suite presidencial Vista a la piscina

Suite presidencial Vista al jardín

Premium Vista a la piscina

Premium Sin vista privilegiada

Premium Vista al jardín La asignación de las habitaciones se realiza por orden de llegada y teniendo en cuenta las preferencias del cliente. Un Huésped solo puede solicitar una habitación.

El huésped no puede elegir la numeración de la habitación, se asigna la contigua al número de habitación que está reservada.

Se hace necesario crear un sistema de control de reservas que se adecue a la política del Costa Brava.

Estos son los diagramas de la abstracción del problema:

Figure 2: Diagramas de clases

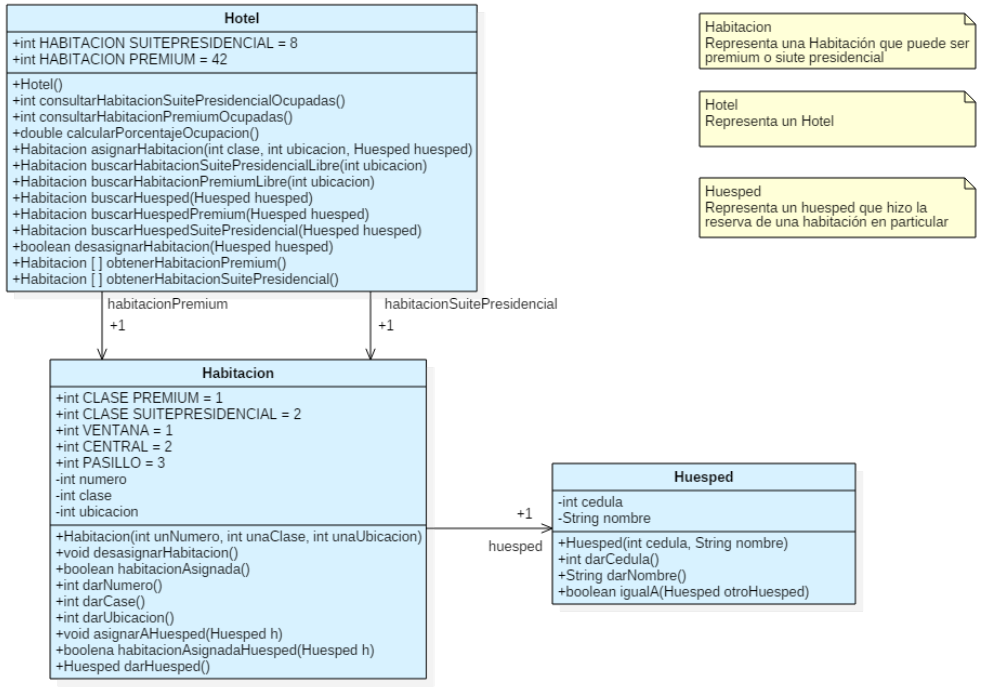
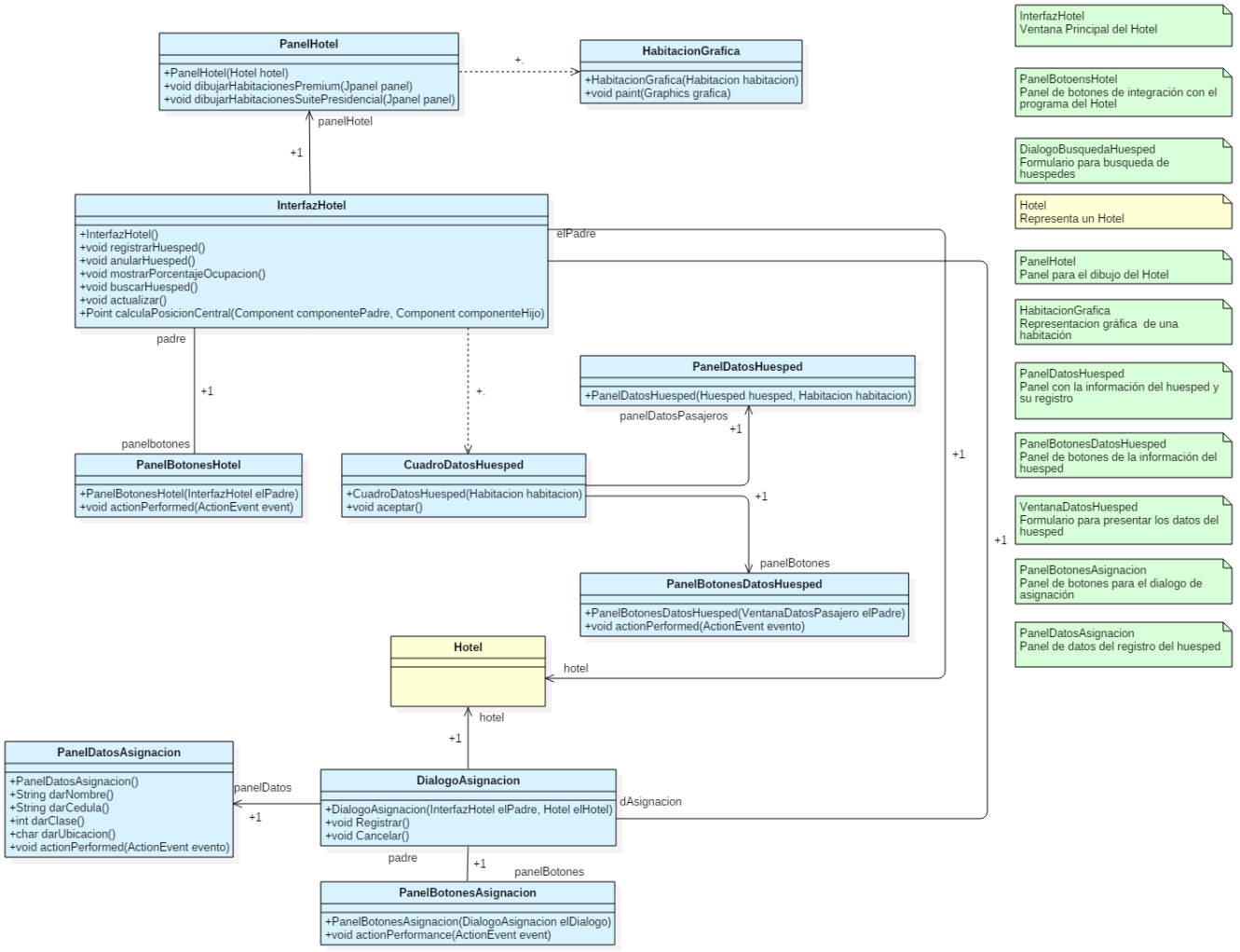


Figure 3: Diagramas de la GUI



3 REQUERIMIENTOS

Listado de requerimientos del problema expuesto, con sus respectivos casos de uso:

Figure 4: Requerimiento 1

Nombre	Asignar una habitación a un huésped
Descripción	Se requiere asignar una habitación según las preferencias del huésped, estas son clase: *Suite presidencial* y *Premiun;* ubicación: vista piscina, vista jardín y sin vista. Se debe dligenciar los datos del huésped.
Entradas	
Nombre	**Descripción**
Nombre	Nombre del huésped
Cédula	Cédula del huésped
Clase	Clase de habitación que desea
Ubicación	Ubicación de la habitación que desea
Resultados	
Nombre	**Descripción**
Asignación de habitación	Si existe una habitación como el huésped la desea
Manejo de errores	
Situación de error	**Acción**
Si no hay habitaciones disponicles con las preferencas del huésped	Se muestra un mensaje informando que no hay habitaciones con esas especificaciones
Si el usuario ya tiene asignada una habitación	No se permite asignar la habtación y se muestra un mensaje indicando que ya tine u ahabitacion asignada

Figure 5: Casos de uso Requerimiento 1

Nombre	Asignar una habitación a un huésped	
Pasos	**Resultado**	
Nombre	**Caso exitoso**	**Caso fallido**
Registrar huésped: dar click en registrar huésped	Aparece un cuadro de diálogo para ingresar los datos	Mensaje de error en la consola indicando que hay un problema que impide el funcionamiento de la applet
Ingresar número de cédula		
Ingresar nombre del huésped		
Escoger clase de la habitación		
Escoger ubicación de la habitación		
Aceptar	La habitación seleccionada es asignada al huésped registrado	La cédula tiene que ser númerica
		Si algún campo está vacío, se indica que ese campo es requerido y se regresa al cuadro de ingreso de datos
		Si el huésped ya tiene una habitación se le indica que otra asignación no es factible
		Si no hay habitaciones con esas especificaciones se indica mediante un mensaje
Cancelar	Asignación no realizada	Fallo del aplicativo

Figure 6: Requerimiento 2

Nombre	Desasignar una habitación a un huésped
Descripción	Con el número de la cédula se elimina la reservación del huésped
Entradas	
Nombre	**Descripción**
Cédula	Cédula del huésped
Resultados	
Nombre	**Descripción**
habitación	La habitación que tenía el huésped está libre
Manejo de errores	
Situación de error	**Acción**
El huésped no tenía ninguna habitación asignada	Se muestra un mensaje indicando que el huésped no tenía habitación asignada

Figure 7: Casos de uso Requerimiento 2

Nombre	Desasignar una habitación a un huésped	
Pasos	**Resultado**	
Nombre	**Caso exitoso**	**Caso fallido**
Eliminar huésped: dar click en eliminar huésped	Aparece un cuadro de diálogo para ingresar el número de cédula del huésped	Mensaje de error en la consola indicando que hay un problema que impide el funcionamiento de la applet
Aceptar	La habitación del huésped ingresado es desasignada	Mensaje: error en el número de cédula
		Mensaje: el huésped no tiene habitación asignada
Cancelar	Mensaje: error en el número de cédula	Fallo del aplicativo

Figure 8: Requerimiento 3

Nombre	Conocer los datos de reserva de un huésped
Descripción	Con el número de cédula de un huésped, si tiene reservación, se visualiza sus datos.
Entradas	
Nombre	**Descripción**
Cédula	Cédula del huésped
Resultados	
Nombre	**Descripción**
Nombre	Nombre del huésped
Cédula	Cédula del huésped
Habitación	Número de la habitación
Clase	Clase de habitación
Ubicación	Ubicación de la habitación
Manejo de errores	
Situación de error	**Acción**
Si el huésped no tiene habitación asignada	Se muestra un mensaje informando que el huésped no tiene habitación asignada

Figure 9: Casos de uso Requerimiento 3

Nombre	Conocer los datos de reserva de un huésped	
Pasos	**Resultado**	
Nombre	**Caso exitoso**	**Caso fallido**
Buscar huésped: dar click en buscar huésped	Aparece un cuadro de diálogo para ingresar el número de cédula del huésped	Mensaje de error en la consola indicando que hay un problema que impide el funcionamiento de la applet
Aceptar	Muestra los datos del huésped: Nombre, cédula, clase, habitación y ubicación	Mensaje: error en el número de cédula
		Mensaje: el huésped no se encuentra registrado
Cancelar	Mensaje: error en el número de cédula	Fallo del aplicativo
		Mensaje: error en el número de cédula

Figure 10: Requerimiento 4

Nombre	Conocer los datos de reserva de un huésped
Descripción	Con el número de cédula de un huésped, si tiene reservación, se visualiza sus datos.
Entradas	
Nombre	**Descripción**
Cédula	Cédula del huésped
Resultados	
Nombre	**Descripción**
Nombre	Nombre del huésped
Cédula	Cédula del huésped
Habitación	Número de la habitación
Clase	Clase de habitación
Ubicación	Ubicación de la habitación
Manejo de errores	
Situación de error	**Acción**
Si el huésped no tiene habitación asignada	Se muestra un mensaje informando que el huésped no tiene habitación asignada

Figure 11: Casos de uso Requerimiento 4

Nombre	Conocer el porcentaje de ocupación del Costa Brava	
Pasos	Resultado	
Nombre	Caso exitoso	Caso fallido
Conocer porcentaje de ubicación: dar click en porcentaje ocupación	Aparece un cuadro de diálogo mostrando el porcentaje de ocupación del hotel	Mensaje de error en la consola indicando que hay un problema que impide el funcionamiento de la applet
Aceptar	Libera la pantalla	

4 PRUEBAS

Para alcanzar calidad en el producto de software (SW), es indispensable la ejecución de pruebas durante el proceso de desarrollo, en este caso se trabaja con la siguiente estructura de pruebas:

4.1 PRUEBAS UNITARIAS

Se diseña y ejecuta una prueba para cada uno de los métodos de las clases del package MUNDO (Habitacion, Huesped y Hotel), para tener la certeza de que el desarrollo no presenta incidencias. Se utiliza la estrategía de pruebas de caja blanca.

Se crea una carpeta llamada test/src y el package se denomina pruebasUnitarias, las cuales se ejecutan utilizando JUnit.

PRUEBAS INDIVIDUALES:

TestAsignarDesasignar

```
package pruebasUnitarias;

import junit.framework.TestCase;
import CostaBravaMUNDO.Hotel;
import CostaBravaMUNDO.Huesped;
import CostaBravaMUNDO.Habitacion;

public class TestAsignarDesasignar extends TestCase {
```

```
//ATRIBUTOS

    //Hotel
    private Hotel hotel;

    //Huésped 1
    private Huesped h1;

    //Huésped 2
    private Huesped h2;

    // Huésped 3
    private Huesped h3;

    //Huésped 4
    private Huesped h4;

    //Nombre del Huésped 1
    private String nombre1;

    // Cédula del Huésped 1
    private int cedula1;

    // Nombre del Huésped 2
    private String nombre2;

    //Cédula del Huésped 2
    private int cedula2;

    //MÉTODO PARA CREAR ESCENARIOS

    private void setupEscenario1( )
    {
        //Crea el hotel
        hotel = new Hotel( );

        //Prepara los nombres y cédulas
        nombre1 = "Camilo Pérez";
        cedula1 = 12345;
        nombre2 = "Fernando Santander";
```

```
        cedula2 = 23456;

        //Crea los huéspedes
       h1 = new Huesped( cedula1, nombre1 );
       h2 = new Huesped( cedula2, nombre2 );

        //Asigna el primer huésped en una habitación suite presidencial del jardín
        hotel.asignarHabitacion( Habitacion.CLASE_SUITEPRESIDENCIAL, Habitacion.JAR

        //Asigna al segundo huésped en una habitacion premium con vista a la piscin
        hotel.asignarHabitacion( Habitacion.CLASE_PREMIUM, Habitacion.PISCINA, h2 )

    }

    //Prepara los datos de prueba para probar el hotel
    //Se crean dos huéspedes, uno de ellos se asigna a una habitación suite preside
    private void setupEscenario2( )
    {
        String nombre;
        int cedula;

        //Usa el escenario 1
        setupEscenario1( );

        //Crea los huéspedes
        nombre = "Clara Martínez";
        cedula = 34567;
       h3 = new Huesped( cedula, nombre );
        nombre = "Sonia Osorio";
        cedula = 56789;
       h4 = new Huesped( cedula, nombre );

    }

  //Verifica que la asignación de una habitación suite presidencial haya sido corre
    public void testAsignarHabitacion1( )
    {
        Habitacion habitacionH1;
        Huesped h;
```

```
        //Configura los datos de prueba
        setupEscenario1( );

        habitacionH1 = hotel.buscarHuesped( h1 );

        //El huésped 1 se hospeda en suite presidencial
        assertEquals( Habitacion.CLASE_SUITEPRESIDENCIAL, habitacionH1.darClase( )

        //El huésped 1 se hospeda con vista al jardín
        assertEquals( Habitacion.JARDIN, habitacionH1.darUbicacion( ) );

        //La primera habitación suite presidencial en jardín es la número 1
        assertEquals( 1, habitacionH1.darNumero( ) );

        //El huésped debe ser el mismo
        h = habitacionH1.darHuesped( );
        assertTrue( h1.igualA( h ) );

    }

  //Verifica que la asignación de una habitación premium haya sido correcta
    public void testAsignarHabitacion( )
    {
        Habitacion habitacionH2;
        Huesped h;

        //Configura los datos de prueba
        setupEscenario1( );

        habitacionH2 = hotel.buscarHuesped( h2 );

        //El huésped 2 reserva en premium
        assertEquals( Habitacion.CLASE_PREMIUM, habitacionH2.darClase( ) );

        //El huésped 2 reserva con vista  a la piscina
        assertEquals( Habitacion.PISCINA, habitacionH2.darUbicacion( ) );

        //La primera habitación premium en con vista a las piscinas es la número 11
        assertEquals( 11, habitacionH2.darNumero( ) );
```

```
        //El huésped debe ser el mismo
        h = habitacionH2.darHuesped( );
        assertTrue( h2.igualA( h ) );

    }

  //Verifica la desasignación de habitaciones
    public void testDesasignarHabitacion1( )
    {
        Habitacion s;

        //Configura los datos de prueba
        setupEscenario1( );

        hotel.desasignarHabitacion( h1 );

        //Ya el huésped no debe estar en el hotel
        s = hotel.buscarHuesped( h1 );

        if( s == null )
            assertTrue( true );
        else
            fail( "El huésped no debería estar" );
    }
}
```

TestCalcularPorcentaje

```
package pruebasUnitarias;

import junit.framework.TestCase;
import CostaBravaMUNDO.Hotel;
import CostaBravaMUNDO.Huesped;
import CostaBravaMUNDO.Habitacion;

public class TestCalcularPorcentaje extends TestCase {

//ATRIBUTOS
```

```
//Hotel
private Hotel hotel;

//Huésped 1
private Huesped h1;

//Huésped 2
private Huesped h2;

// Huésped 3
private Huesped h3;

//Huésped 4
private Huesped h4;

//Nombre del Huésped 1
private String nombre1;

// Cédula del Huésped 1
private int cedula1;

// Nombre del Huésped 2
private String nombre2;

//Cédula del Huésped 2
private int cedula2;

//MÉTODO PARA CREAR ESCENARIOS

private void setupEscenario1( )
{
    //Crea el avión
    hotel = new Hotel( );

    //Prepara los nombres y cédulas
    nombre1 = "Camilo Pérez";
    cedula1 = 12345;
    nombre2 = "Fernando Santander";
    cedula2 = 23456;
```

```
        //Crea los huéspedes
       h1 = new Huesped( cedula1, nombre1 );
       h2 = new Huesped( cedula2, nombre2 );

        //Asigna el primer huésped en una habitación suite presidencial del jardín
        hotel.asignarHabitacion( Habitacion.CLASE_SUITEPRESIDENCIAL, Habitacion.JARI

        //Asigna al segundo huésped en una habitacion premium con vista a la piscin
        hotel.asignarHabitacion( Habitacion.CLASE_PREMIUM, Habitacion.PISCINA, h2 )

    }

    //Prepara los datos de prueba para probar el hotel
    //Se crean dos huéspedes, uno de ellos se asigna a una habitación suite preside
    private void setupEscenario2( )
    {
        String nombre;
        int cedula;

        //Usa el escenario 1
        setupEscenario1( );

        //Crea los huéspedes
        nombre = "Clara Martínez";
        cedula = 34567;
       h3 = new Huesped( cedula, nombre );
        nombre = "Sonia Osorio";
        cedula = 56789;
       h4 = new Huesped( cedula, nombre );

    }

  //Prueba el porcentaje de ocupación
    public void testCalcularPorcentajeOcupacion1( )
    {
        double porcentajeEsperado, porcentaje;

        //Configura los datos de prueba
        setupEscenario2( );
```

```
        //inicialmente el porcentaje de ocupación es igual a 2*100/50 porque hay do
        porcentajeEsperado = ( 2 * 100 ) / 50;
        porcentaje = hotel.calcularPorcentajeOcupacion( );
        assertEquals( porcentajeEsperado, porcentaje, 0 );

        //Asigno otros dos huéspedes
        hotel.asignarHabitacion( Habitacion.CLASE_PREMIUM, Habitacion.SINVISTA, h3
        hotel.asignarHabitacion( Habitacion.CLASE_SUITEPRESIDENCIAL, Habitacion.JAR

        //Ahora el porcentaje es 4*100/50
        porcentajeEsperado = ( 4 * 100 ) / 50;
        porcentaje = hotel.calcularPorcentajeOcupacion( );
        assertEquals( porcentajeEsperado, porcentaje, 0 );
    }
}
```

TestBuscarHabitacion

```
package pruebasUnitarias;

import junit.framework.TestCase;
import CostaBravaMUNDO.Hotel;
import CostaBravaMUNDO.Huesped;
import CostaBravaMUNDO.Habitacion;

public class TestBuscarHabitacion extends TestCase {

//ATRIBUTOS

    //Hotel
    private Hotel hotel;

    //Huésped 1
    private Huesped h1;

    //Huésped 2
    private Huesped h2;

    // Huésped 3
    private Huesped h3;
```

```
//Huésped 4
private Huesped h4;

//Nombre del Huésped 1
private String nombre1;

// Cédula del Huésped 1
private int cedula1;

// Nombre del Huésped 2
private String nombre2;

//Cédula del Huésped 2
private int cedula2;

//MÉTODO PARA CREAR ESCENARIOS

private void setupEscenario1( )
{
    //Crea el avión
    hotel = new Hotel( );

    //Prepara los nombres y cédulas
    nombre1 = "Camilo Pérez";
    cedula1 = 12345;
    nombre2 = "Fernando Santander";
    cedula2 = 23456;

    //Crea los huéspedes
   h1 = new Huesped( cedula1, nombre1 );
   h2 = new Huesped( cedula2, nombre2 );

    //Asigna el primer huésped en una habitación suite presidencial del jardín
    hotel.asignarHabitacion( Habitacion.CLASE_SUITEPRESIDENCIAL, Habitacion.JAR

    //Asigna al segundo huésped en una habitacion premium con vista a la piscin
    hotel.asignarHabitacion( Habitacion.CLASE_PREMIUM, Habitacion.PISCINA, h2 )

}
```

```
  //Prepara los datos de prueba para probar el hotel
  //Se crean dos huéspedes, uno de ellos se asigna a una habitación suite preside
  private void setupEscenario2( )
  {
      String nombre;
      int cedula;

      //Usa el escenario 1
      setupEscenario1( );

      //Crea los huéspedes
      nombre = "Clara Martínez";
      cedula = 34567;
     h3 = new Huesped( cedula, nombre );
      nombre = "Sonia Osorio";
      cedula = 56789;
     h4 = new Huesped( cedula, nombre );

  }

//Busca la siguiente habitación premium libre
  public void testBuscarHabitacionPremiumLibre1( )
  {
      Habitacion s;

      //Configura los datos de prueba
      setupEscenario1( );

      //La siguiente habitación premium de con vista a las piscinas libre es la 1
      s = hotel.buscarHabitacionPremiumLibre( Habitacion.PISCINA );
      assertEquals( 12, s.darNumero( ) );

      //La siguiente habitación premium de jardín libre es la 9
      s = hotel.buscarHabitacionPremiumLibre( Habitacion.JARDIN );
      assertEquals( 9, s.darNumero( ) );

      //La siguiente habitación premium sin vista libre es la 10
      s = hotel.buscarHabitacionPremiumLibre( Habitacion.PISCINA );
      assertEquals( 12, s.darNumero( ) );
```

```
    }

    //Busca la siguiente habitación suite presidencial libre
    public void testBuscarHabitacionSuitePresidencialLibre1( )
    {
        Habitacion s;

        //Configura los datos de prueba
        setupEscenario1( );

        //La siguiente habitación suite presidencial con vista a las piscinas libre
        s = hotel.buscarhabitacionSuitePresidencialLibre( Habitacion.PISCINA );
        assertEquals( 2, s.darNumero( ) );

        //La siguiente habitación suite presidencial con vista al jardín libre es l
        s = hotel.buscarhabitacionSuitePresidencialLibre( Habitacion.JARDIN );
        assertEquals( 4, s.darNumero( ) );

    }

}
```

TestBuscarHuesped

```
package pruebasUnitarias;

import junit.framework.TestCase;
import CostaBravaMUNDO.Hotel;
import CostaBravaMUNDO.Huesped;
import CostaBravaMUNDO.Habitacion;

public class TestBuscarHuesped extends TestCase {

//ATRIBUTOS

    //Hotel
    private Hotel hotel;

    //Huésped 1
```

```
private Huesped h1;

//Huésped 2
private Huesped h2;

// Huésped 3
private Huesped h3;

//Huésped 4
private Huesped h4;

//Nombre del Huésped 1
private String nombre1;

// Cédula del Huésped 1
private int cedula1;

// Nombre del Huésped 2
private String nombre2;

//Cédula del Huésped 2
private int cedula2;

//MÉTODO PARA CREAR ESCENARIOS

private void setupEscenario1( )
{
    //Crea el avión
    hotel = new Hotel( );

    //Prepara los nombres y cédulas
    nombre1 = "Camilo Pérez";
    cedula1 = 12345;
    nombre2 = "Fernando Santander";
    cedula2 = 23456;

    //Crea los huéspedes
   h1 = new Huesped( cedula1, nombre1 );
   h2 = new Huesped( cedula2, nombre2 );
```

```
        //Asigna el primer huésped en una habitación suite presidencial del jardín
        hotel.asignarHabitacion( Habitacion.CLASE_SUITEPRESIDENCIAL, Habitacion.JAR

        //Asigna al segundo huésped en una habitacion premium con vista a la piscin
        hotel.asignarHabitacion( Habitacion.CLASE_PREMIUM, Habitacion.PISCINA, h2 )

    }

    //Prepara los datos de prueba para probar el hotel
    //Se crean dos huéspedes, uno de ellos se asigna a una habitación suite preside
    private void setupEscenario2( )
    {
        String nombre;
        int cedula;

        //Usa el escenario 1
        setupEscenario1( );

        //Crea los huéspedes
        nombre = "Clara Martínez";
        cedula = 34567;
       h3 = new Huesped( cedula, nombre );
        nombre = "Sonia Osorio";
        cedula = 56789;
       h4 = new Huesped( cedula, nombre );

    }

  //Verifica la búsqueda de un huésped económico que existe
    public void testBuscarHuesped1( )
    {
        Huesped h;
        Habitacion s;

        //Configura los datos de prueba
        setupEscenario1( );

        s = hotel.buscarHuespedPremium( h2 );
        if( s == null )
            fail( "El huésped debería existir" );
```

```
        else
        {
            h = s.darHuesped( );
            assertEquals( cedula2, h.darCedula( ) );
            assertEquals( nombre2, h.darNombre( ) );
        }
    }

  //Verifica la búsqueda de un huésped prtemium que no existe
    public void testBuscarHuesped2( )
    {
        Habitacion s;

        //Configura los datos de prueba
        setupEscenario1( );

        s = hotel.buscarHuespedPremium( h1 );
        if( s == null )
            assertTrue( true );
        else
        {
            fail( "El huésped NO debería existir" );
        }
    }

  //Verifica la búsqueda de un huésped ejecutivo que existe
    public void testBuscarHuesped3( )
    {
        Huesped h;
        Habitacion s;

        //Configura los datos de prueba
        setupEscenario1( );

        s = hotel.buscarHuespedSuitepresidencial( h1 );
        if( s == null )
            fail( "El huésped debería existir" );
        else
        {
            h = s.darHuesped( );
```

```
            assertEquals( cedula1, h.darCedula( ) );
            assertEquals( nombre1, h.darNombre( ) );
        }
    }

 // Verifica la búsqueda de un huésped suite presidencial que no existe
    public void testBuscarHuesped4( )
    {
        Habitacion s;

        //Configura los datos de prueba
        setupEscenario1( );

        s = hotel.buscarHuespedSuitepresidencial( h2 );
        if( s == null )
            assertTrue( true );
        else
        {
            fail( "El huésped NO debería existir" );
        }
    }
}
```

TestContarHabitaciones

```
package pruebasUnitarias;

import junit.framework.TestCase;
import CostaBravaMUNDO.Hotel;
import CostaBravaMUNDO.Huesped;
import CostaBravaMUNDO.Habitacion;

public class TestContarHabitaciones extends TestCase {

//ATRIBUTOS

    //Hotel
    private Hotel hotel;

    //Huésped 1
```

```
private Huesped h1;

//Huésped 2
private Huesped h2;

// Huésped 3
private Huesped h3;

//Huésped 4
private Huesped h4;

//Nombre del Huésped 1
private String nombre1;

// Cédula del Huésped 1
private int cedula1;

// Nombre del Huésped 2
private String nombre2;

//Cédula del Huésped 2
private int cedula2;

//MÉTODO PARA CREAR ESCENARIOS

private void setupEscenario1( )
{
    //Crea el avión
    hotel = new Hotel( );

    //Prepara los nombres y cédulas
    nombre1 = "Camilo Pérez";
    cedula1 = 12345;
    nombre2 = "Fernando Santander";
    cedula2 = 23456;

    //Crea los huéspedes
   h1 = new Huesped( cedula1, nombre1 );
   h2 = new Huesped( cedula2, nombre2 );
```

```
        //Asigna el primer huésped en una habitación suite presidencial del jardín
        hotel.asignarHabitacion( Habitacion.CLASE_SUITEPRESIDENCIAL, Habitacion.JAR

        //Asigna al segundo huésped en una habitacion premium con vista a la piscin
        hotel.asignarHabitacion( Habitacion.CLASE_PREMIUM, Habitacion.PISCINA, h2 )

    }

    //Prepara los datos de prueba para probar el hotel
    //Se crean dos huéspedes, uno de ellos se asigna a una habitación suite preside
    private void setupEscenario2( )
    {
        String nombre;
        int cedula;

        //Usa el escenario 1
        setupEscenario1( );

        //Crea los huéspedes
        nombre = "Clara Martínez";
        cedula = 34567;
       h3 = new Huesped( cedula, nombre );
        nombre = "Sonia Osorio";
        cedula = 56789;
       h4 = new Huesped( cedula, nombre );

    }

  //Cuenta las habitaciones premium ocupadas
    public void testContarHabitacionesPremiumOcupadas1( )
    {

        //Configura los datos de prueba
        setupEscenario2( );

        //Inicialmente las habitaciones premium ocupadas son:  1
        assertEquals( 1, hotel.contarHabitacionesPremiumOcupadas( ) );

        //Asignando otros dos huéspedes
        hotel.asignarHabitacion( Habitacion.CLASE_PREMIUM, Habitacion.SINVISTA, h3
```

```
        hotel.asignarHabitacion( Habitacion.CLASE_PREMIUM, Habitacion.JARDIN, h4 );

        //Ahora el número de habitaciones ocupadas es 3
        assertEquals( 3, hotel.contarHabitacionesPremiumOcupadas( ) );
    }

  //Cuenta las habitaciones suite presidencial ocupadas
    public void testContarHabitacionesSuitepresidencialOcupadas1( )
    {

        Habitacion s;

        //Configura los datos de prueba
        setupEscenario2( );

        //Inicialmente las habitaciones suite presidencial ocupadas son: 1
        assertEquals( 1, hotel.contarHabitacionesSuitePresidencialOcupadas( ) );

        //Asignando otros dos huéspedes
        s = hotel.asignarHabitacion( Habitacion.CLASE_SUITEPRESIDENCIAL, Habitacion
        if( s == null )
            fail( "Debió asignar alguna habitación 1" );

        s = hotel.asignarHabitacion( Habitacion.CLASE_SUITEPRESIDENCIAL, Habitacion
        if( s == null )
            fail( "Debió asignar alguna habitación 2" );

        //Ahora el número de habitaciones ocupadas es 3
        assertEquals( 3, hotel.contarHabitacionesSuitePresidencialOcupadas( ) );
    }
}
```

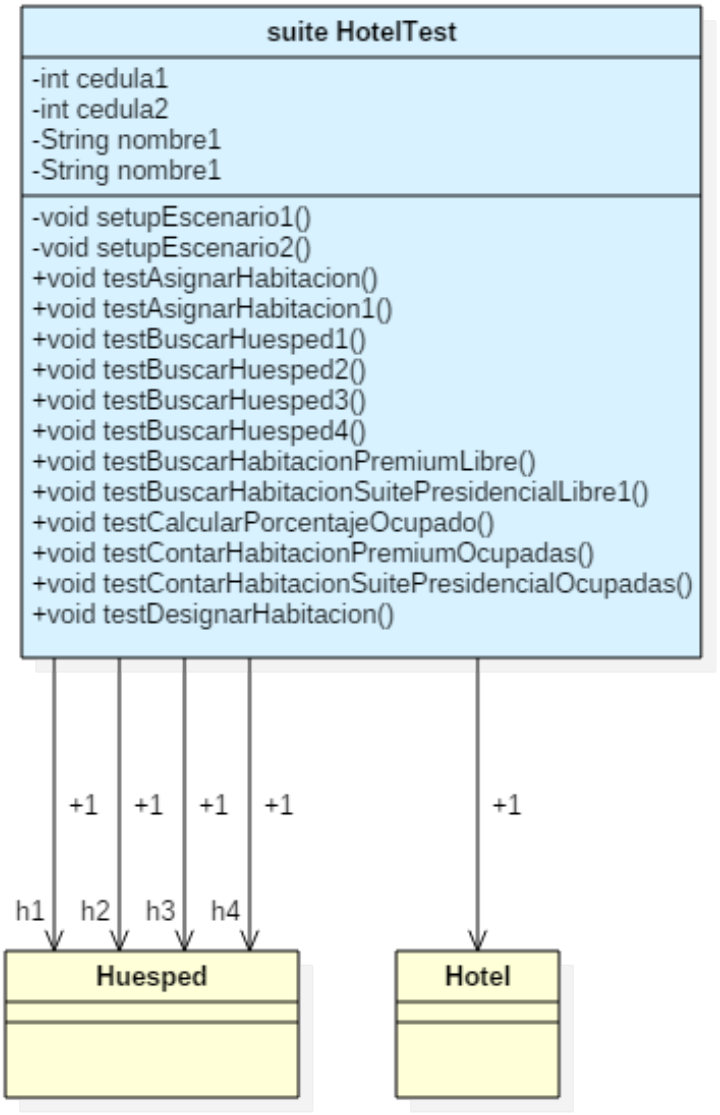
4.2 PRUEBAS SUITE

Se construyó una suite de pruebas con JUnit, llamada SuiteTest donde se incluye a todas las clases de prueba de la sección 4.1 pruebas unitarias.

Esto también se puede revisar en el repositorio en GitHub en la rama de pruebas con JUnit.

H

Figure 12: Diagrama de la suite de pruebas



SUITE DE PRUEBAS

SuiteTest

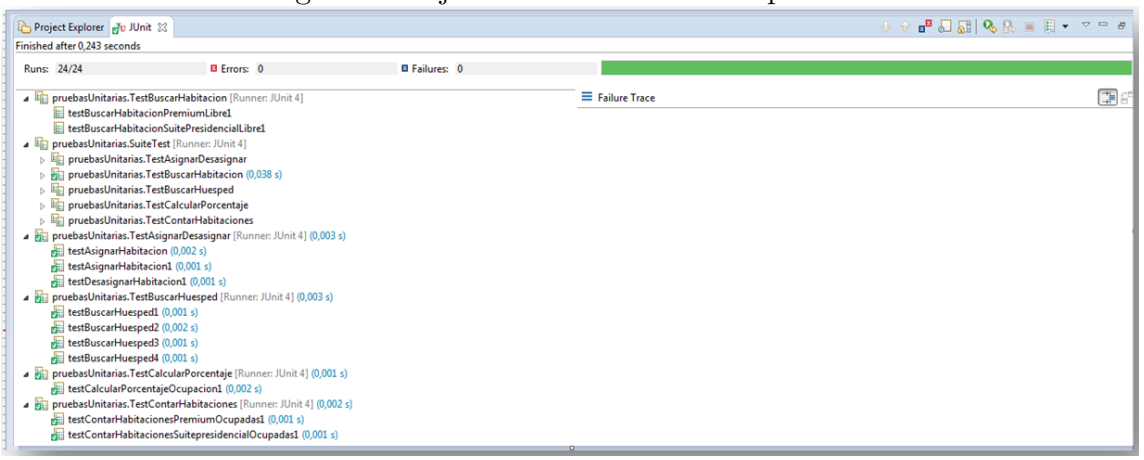
```
package pruebasUnitarias;

import org.junit.runner.RunWith;
import org.junit.runners.Suite;
import org.junit.runners.Suite.SuiteClasses;

@RunWith(Suite.class)
@SuiteClasses({
TestAsignarDesasignar.class,
TestBuscarHabitacion.class,
TestBuscarHuesped.class,
TestCalcularPorcentaje.class,
TestContarHabitaciones.class })
public class SuiteTest {
}
```

La figura 13. evidencia la ejecucción la suite de pruebas, donde se observa que todas se ejecutan sin reportar errores.

Figure 13: Ejecucción de la suite de pruebas



4.3 PRUEBAS SELENIUM

Estas pruebas no se realizaron porque el desarrollo es StandAlone y no se encontró una herramienta semejante para implementar pruebas de integración a aplicaciones de esta clasificación.

5 CASOS DE PRUEBA

En esta sección se dan a conocer todos los datos para que el tester lleve a cabo pruebas, donde los usuarios son personas, semejantes al ususrio final, ajenas al ambiente de desarrollo. En este caso la estrategía implementada son pruebas de caja negra.

Al ingresar los datos predefinidos se obtienen resultados, que deben ser los esperados, de lo contrario el producto aún no es de calidad. Se procede a revisar y corregir el problema que dificulta el funcionamiento idóneo.

El tiempo es un gran limitante, puesto que al momento de hacer estas pruebas se supone que el programa debe funcionar eficazmente y la fecha d entrega está muy cercana.

Figure 14: CASOS DE PRUEBA

CASOS DE PRUEBA	
DATOS PREDEFINIDOS EN EL PROGRAMA	
Escenario 1	Nombre: Camilo Pérez Cédula = 12345 Habitación: 1 Clase: Suite presidencial Ubicación: con vista al jardín
	Nombre: Fernando Santander Cédula = 23456 Habitación: 11 Clase: Premium Ubicación: piscina
Escenario 2	Además de los datos del escenario 1, tiene dos huéspedes:
	"Clara Martínez" Cédula Clara Martínez = 34567
	"Sonia Osorio" Cédula Sonia Osorio = 56789
	Estos huéspedes no tienen habitación asignada

PRUEBA 1		
Buscar huésped		
Datos a ingresar	**Resultado esperado**	**Caso de fallo**
Cédula: 12345	Nombre: Camilo Pérez Cédula = 12345 Habitación: 1 Clase: Suite presidencial Ubicación: con vista al jardín	_Opción 1: Número de cédula mal digitado
		_Opción 2: El número de cédula está bien digitado, reportar el error, adjuntando los datos ingresados, el resultado obtenido, la fecha de hallazgo, datos de responsables.
Cédula = 23456	Nombre: Fernando Santander Cédula = 23456 Habitación: 11 Clase: Premium Ubicación: piscina	_Opción 1: Número de cédula mal digitado
		_Opción 2: El número de cédula está bien digitado, reportar el error, adjuntando los datos ingresados, el resultado obtenido, la fecha de hallazgo, datos de responsables.
PRUEBA 2		
Asignar habitación		
Datos a ingresar	**Resultado esperado**	**Caso de fallo**
Nombre: Clara Martínez **Cédula = 34567** **Clase: premium** **Ubicación: Piscina**	Habitación asignada: Habitación: 12 Clase: Premium Ubicación: Piscina	_Opción 1: Número de cédula mal digitado
		_Opción 2: El número de cédula está bien digitado, reportar el error, adjuntando los datos ingresados, el resultado obtenido, la fecha de hallazgo, datos de responsables.
		_Opción 3: Debió asignar la siguiente habitación premium disponible
Nombre: Sonia Osorio **Cédula = 56789** **Clase: Suite Presidencial** **Ubicación: ventana**	Habitación asignada: Habitación: 4 Clase: Suite Presidencial Ubicación: Ventana	_Opción 1: Número de cédula mal digitado
		_Opción 2: El número de cédula está bien digitado, reportar el error, adjuntando los datos ingresados, el resultado obtenido, la fecha de hallazgo, datos de responsables.
		_Opción 3: Debió asignar la siguiente habitación suite presidencial disponible
PRUEBA 3		
Desasignar habitación (Eliminar huésped)		
Datos a ingresar	**Resultado esperado**	**Caso de fallo**
Cédula = 34567	Habitación desasignada: Habitación: 12 Clase: Premium Ubicación: Piscina	_Opción 1: Número de cédula mal digitado
		_Opción 2: El número de cédula está bien digitado, reportar el error, adjuntando los datos ingresados, el resultado obtenido, la fecha de hallazgo, datos de responsables.
		_Opción 3: El huésped no tiene reserva
		_Opción 4: la habitación no debería seguir reservada
Cédula = 56789	Habitación desasignada: Habitación: 4 Clase: Suite Presidencial Ubicación: Ventana	_Opción 1: Número de cédula mal digitado
		_Opción 2: El número de cédula está bien digitado, reportar el error, adjuntando los datos ingresados, el resultado obtenido, la fecha de hallazgo, datos de responsables.
		_Opción 3: El huésped no tiene reserva
		_Opción 4: La habitación no debería seguir reservada

PRUEBA 4		
Buscar Huéspedes clase premium		
Datos a ingresar	**Resultado esperado**	**Caso de fallo**
Cédula = 34567	Nombre: Clara Martínez Cédula = 34567 Habitación: 12 Clase: Premium Ubicación: Piscina	_Opción 1: El huésped debería existir
		_Opción 2: Número de cédula mal digitado
Cédula = 23456	Nombre: Fernando Santander Cédula = 23456 Habitación: 11 Clase: Premium Ubicación: piscina	_Opción 1: El huésped debería existir
		_Opción 2: Número de cédula mal digitado

PRUEBA 5		
Buscar Huéspedes clase suite presidencial		
Datos a ingresar	**Resultado esperado**	**Caso de fallo**
Cédula = 12345	Nombre: Camilo Pérez Cédula = 12345 Habitación: 1 Clase: Suite presidencial Ubicación: con vista al jardín	_Opción 1: El huésped debería existir
		_Opción 2: Número de cédula mal digitado
Cédula = 56789	Nombre: Sonia Osorio Cédula = 56789 Habitación: 4 Clase: Suite Presidencial Ubicación: Ventana	_Opción 1: El huésped debería existir
		_Opción 2: Número de cédula mal digitado

PRUEBA 6		
Buscar Huéspedes		
Datos a ingresar	**Resultado esperado**	**Caso de fallo**
Cédula = 12345	Nombre: Camilo Pérez Cédula = 12345 Habitación: 1 Clase: Suite presidencial Ubicación: con vista al jardín	_Opción 1: El huésped debería existir
		_Opción 2: Número de cédula mal digitado
Cédula = 56789	Nombre: Sonia Osorio Cédula = 56789 Habitación: 4 Clase: Suite Presidencial Ubicación: Ventana	_Opción 1: El huésped debería existir
		_Opción 2: Número de cédula mal digitado
Cédula = 34567	Nombre: Clara Martínez Cédula = 34567 Habitación: 12 Clase: Premium Ubicación: Piscina	_Opción 1: El huésped debería existir
		_Opción 2: Número de cédula mal digitado
Cédula = 23456	Nombre: Fernando Santander Cédula = 23456 Habitación: 11 Clase: Premium Ubicación: piscina	_Opción 1: El huésped debería existir
		_Opción 2: Número de cédula mal digitado

PRUEBA 7		
Buscar Huésped que no existe		
Datos a ingresar	**Resultado esperado**	**Caso de fallo**
Cédula = 98765	El huésped no se encuentra registrado	_Opción 1: El huésped no debería existir
Cédula = 43210	El huésped no se encuentra registrado	_Opción 2: El huésped no debería existir

PRUEBA 8		
Calcular porcentaje de ocupación		
Datos a ingresar	**Resultado esperado**	**Caso de fallo**
Utilizando escenario vacío	0%	_Opción: Hay un problema interno (en alguno de los métodos), que no permite calcular el porcentaje de forma correcta
Utilizando escenario 1	2%	
Utilizando escenario 2	4%	
Utilizando escenario lleno	100%	

6 REGISTRO DE BUGS

Para llevar a cabo la administración de defectos, de acuerdo a la etapa de detección del incidente se vio que es más fácil darle solución a un problema, dependiendo del críterio de mayor prioridad, así se dio orden a la solución de incidencias.

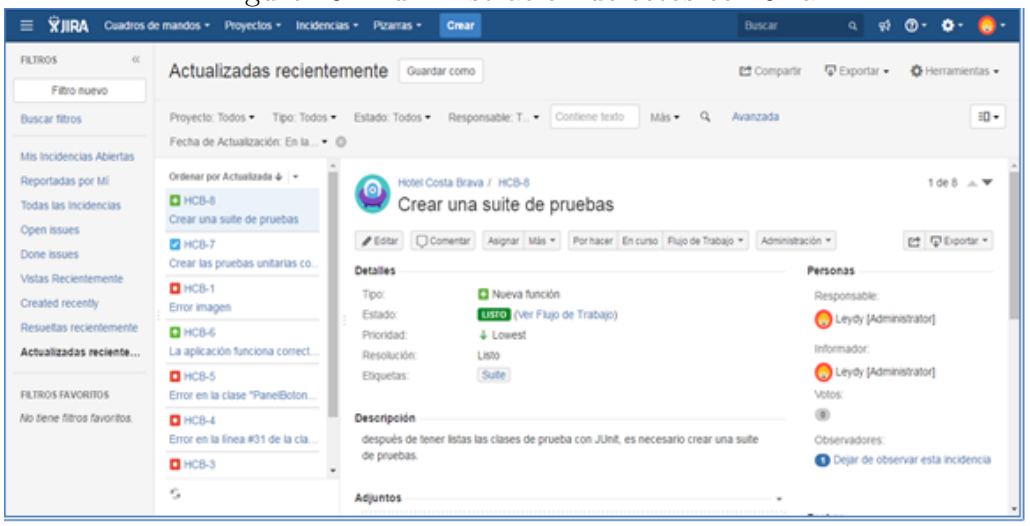
6.1 BUGZILLA

Bugzilla presenta una desventaja significativa y es que hay congestión y la página no siempre está disponible.

6.2 JIRA

Se elige Jira para administrar los defectos del proyecto, donde se tratan las incidencias de acuerdo a su prioridad. Se crearon issues de tipo Bug, Tarea, Mejora, Nueva Función.

Figure 15: Administración defectos con Jira



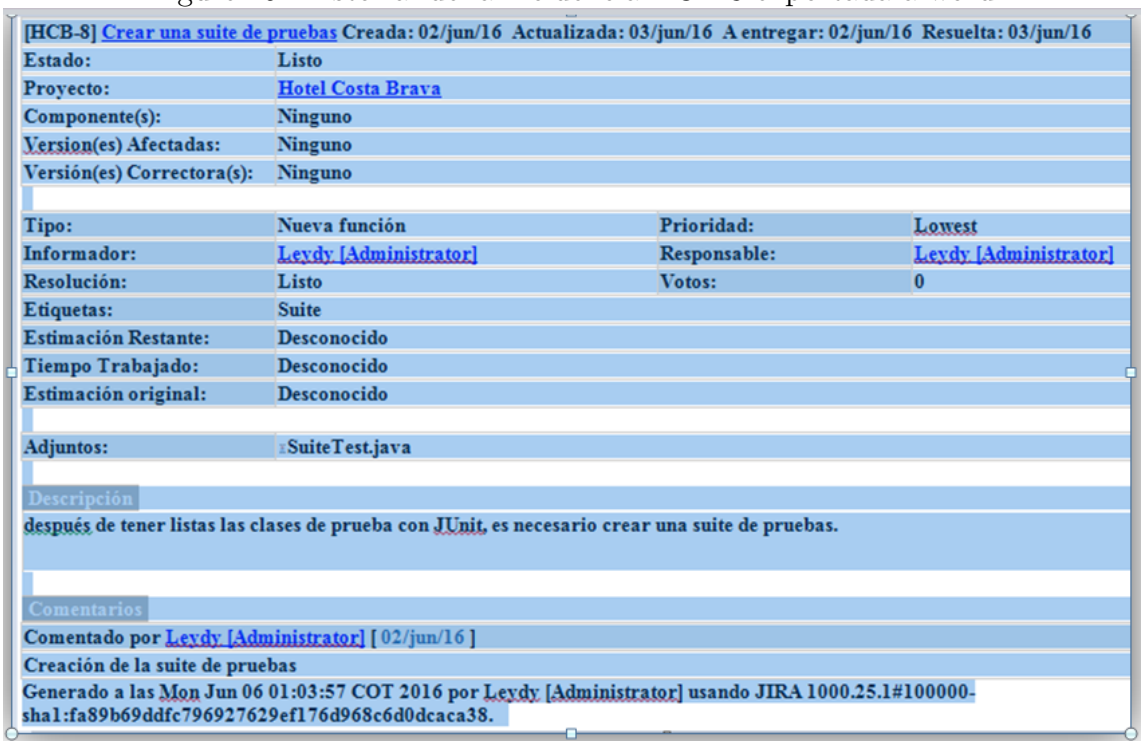
Para consultar el seguimiento de las incidencias, en la parte de herramientas se puede usar el link del proyecto junto con el usuario y la contraseña.

Pasos:

- Jira ofrece un uso de prueba que priva al usuario de algunos privilegios ya que no está pagando el servicio. Se accede a `https://www.atlassian.com/software/jira/` y se empieza a usar la prueba gratis.
- Se crea el nombre del proyecto
- Se colocan los datos del correo y contraseña del proyecto.
- Se empieza a usar Jira.
- Se inicia con la creación de issues.
- Se hace seguimiento a la incidencia.
- se hacen cambios de acuerdo al progreso.
- se añaden colaboradores del equipo de trabajo.
- Jira ofrece variedad de opciones para controlar y seguir el progreso de la solución de incidencias, por ejemplo asignar un rol a cada colaborador.

En Jira también es posible cargar todo el seguimiento de una incidencia en GitHub, además permite exportar a Word cada incidencia.

Figure 16: Historial de la incidencia HCB-8 exportada a word



[HCB-8] Crear una suite de pruebas Creada: 02/jun/16 Actualizada: 03/jun/16 A entregar: 02/jun/16 Resuelta: 03/jun/16

Estado:	Listo		
Proyecto:	Hotel Costa Brava		
Componente(s):	Ninguno		
Version(es) Afectadas:	Ninguno		
Versión(es) Correctora(s):	Ninguno		
Tipo:	Nueva función	Prioridad:	Lowest
Informador:	Leydy [Administrator]	Responsable:	Leydy [Administrator]
Resolución:	Listo	Votos:	0
Etiquetas:	Suite		
Estimación Restante:	Desconocido		
Tiempo Trabajado:	Desconocido		
Estimación original:	Desconocido		
Adjuntos:	SuiteTest.java		

Descripción

después de tener listas las clases de prueba con JUnit, es necesario crear una suite de pruebas.

Comentarios

Comentado por Leydy [Administrator] [02/jun/16]

Creación de la suite de pruebas

Generado a las Mon Jun 06 01:03:57 COT 2016 por Leydy [Administrator] usando JIRA 1000.25.1#100000-sha1:fa89b69ddfc796927629ef176d968c6d0dcaca38.

7 CONTROL DE VERSIONES

El control de versiones se manejó con GITHUB, de entorno local y de entorno remoto.

7.1 MANEJO DE REPOSITORIO

Se creó el repositorio en la nube y se clonó en el git local , trabajando de forma colaborativa con dos usuarios.

Pasos:

- se descarga el Git local dependiendo del sistema operativo y otros requerimientos a seguir en esta página : `https://git-scm.com/`
- Se crea una cuenta en GitHub remoto `https://github.com/`
- GitHub local tiene dos formas de trabajar: por consola o por inrtefaz gráfica, siendo la segunda seleccionada en este caso. La cual se configura colocando los datos de la cuenta que se tiene en el remoto.

- En el remoto se crea un nuevo repositorio, este va a ser vacio.
- A continuación aparece una serie de instrucciones de comandos para ejecutar en el gitHub local por medio de consola. des esta forma ya está creado el repositorio en GitHub remoto.
- Desde el git local se clona los repositorios del remoto.
- Se elige un browser del equipo local.
- Se desarrolla el proyecto en el equipo y los cambios se ven reflejados en la interfaz del local, allí se da click en cambios y para q se actualicen hay que agregar un resumen y realizar el commit.
- Para sincronizar los cambios del local con el remoto, se debe dar click en sincronizar del local.
- Se puede observar en el remoto que los cambios han sido efectuados.

Tanto GitHub local como remoto muestran un historial de commits, además el remoto muestra gráficos, usuarios colaboradores, ramas, entre otras características.

7.2 HISTORIA DE ARTEFACTOS

En la sección de herramientas se anexa la dirección, usuario y contraseña para realizar la revisión del historial de artefactos del repositorio en GitHub remoto, donde cada commit corresponde a un cambio o actualización del proyecto, sincronizado desde el git local.

Figure 17: Historial GitHub remoto

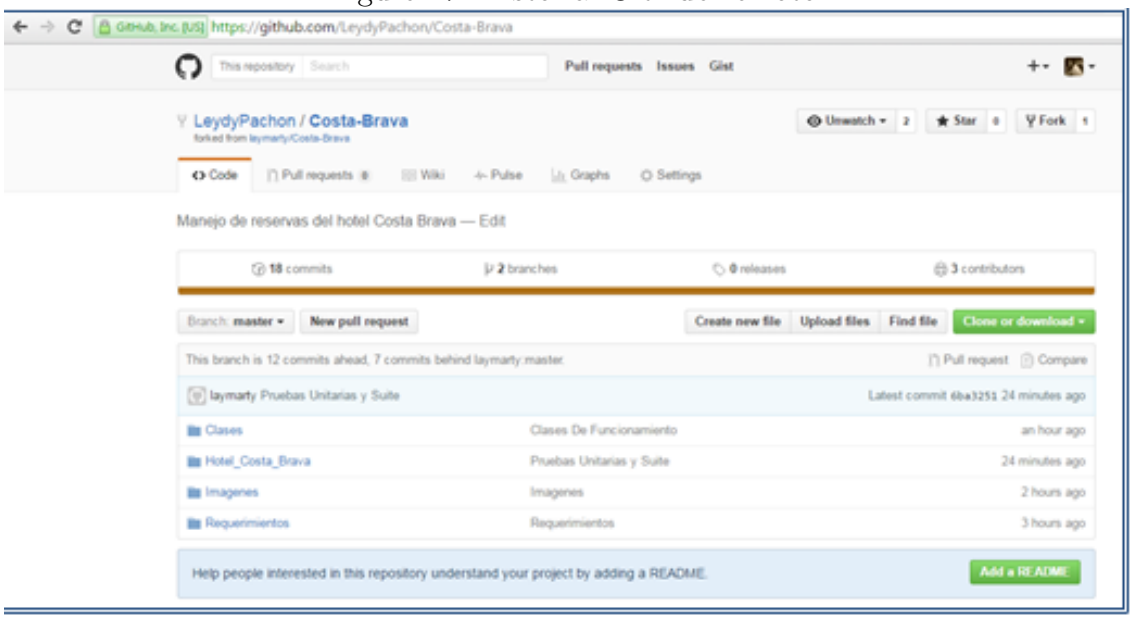


Figure 18: Historial GitHub local 1

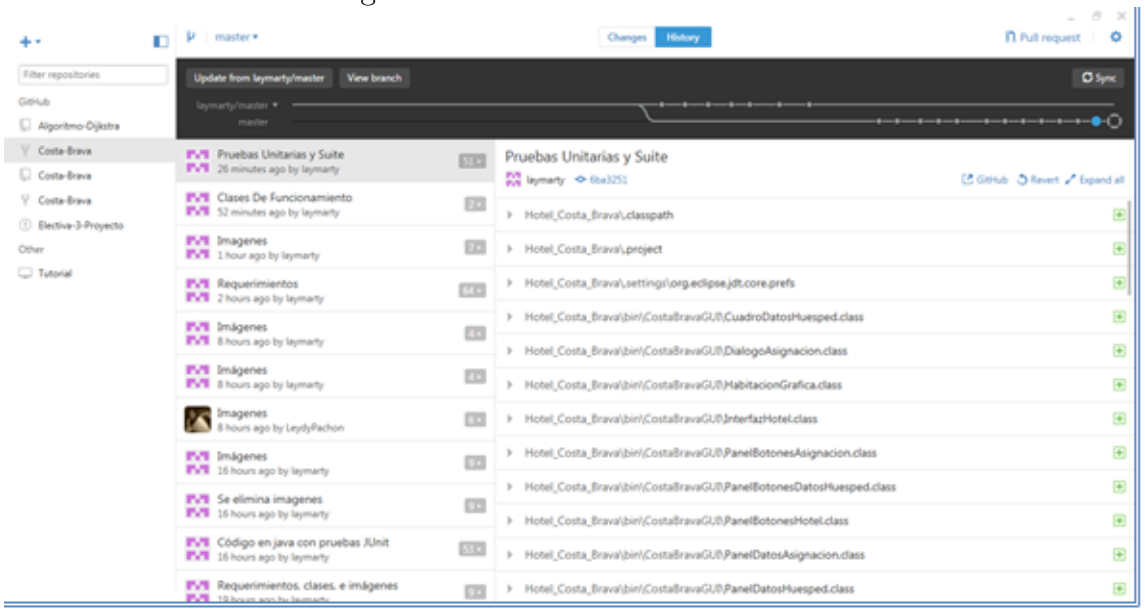
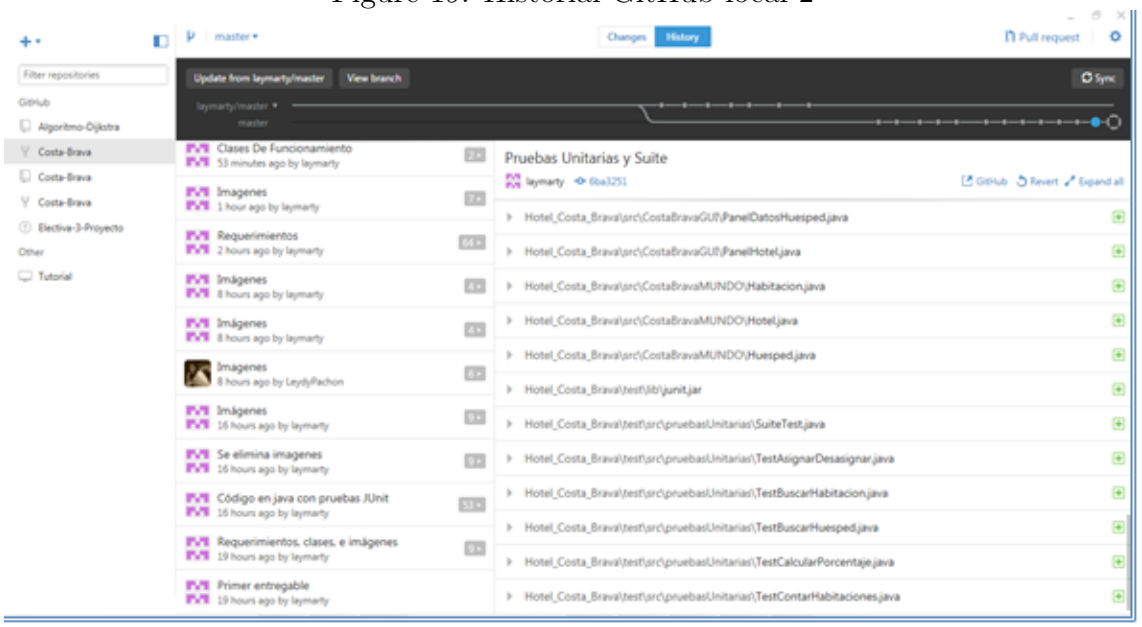


Figure 19: Historial GitHub local 2



8 HERRAMIENTAS

- **Overleaf**

  El documento de este informe se encuentra disponible en:

  `https://git.overleaf.com/5387672bktzkp` Clonar con Git.

  `https://www.overleaf.com/read/dbvxycfbvzzz` Solo lectura.

  `https://www.overleaf.com/5387672bktzkp` Lectura y edición.

  Manejo de reservas del Costa Brava

- **Jira**

  El sitio se llama hotelcostabrava.atlassian.net clave: HCB y se encuentra disponible en:

  `https://HotelCostaBrava.atlassian.net`

  Usuario: leypachon@uan.edu.co

  contraseña: 110103jaider

- **GitHub**

  GitHub Remoto

  El proyecto se llama LeydyPachon/Costa-brava y se encuentra disponible en:

  `https://github.com/LeydyPachon/Costa-Brava`

  Usuario: leypachon@uan.edu.co / laymarty2004@yahoo.es

  Contraseña: 110103Jaider< / 110103jaider

  GitHub Local

  El repositorio se pude clonar de forma local siempre y cuando se conceda permiso, para efectos prácticos se anexa la contraseña del remoto.

- **Eclipse**

  Se usó como editor de código

- **Java**

  Desarrollo completamente en lenguaje Java y el código está publicado en GIT.

  El archivo ejecutable junto con el código se añade en la entrega a través de correo a la ingeniera Claudia hernández.

- **JUnit**

  junit-4.12.jar

  hamcrest-core-1.3.jar

REFERENCIAS

- https://git-scm.com/
- https://github.com/
- http://junit.org/junit4/
- https://jira.atlassian.com/secure/Dashboard.jspa
- https://www.oracle.com/es/java/index.html
- https://eclipse.org/downloads/
- https://www.overleaf.com/dash