# Entrega 5

Calificación de la documentación entrega 5: Disciplina de Implementación: 9.9 puntos / 15 puntos

# Componente de Administración Tour: 6.1 puntos / 10 Puntos

**Componente de recorrido:** 3.4 / 4 Puntos

**Presentación de resultados en la especificación de los casos de prueba:** 0.4 puntos / 1 Puntos

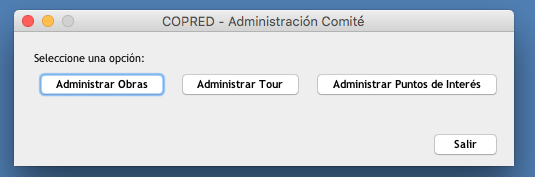
# Detalles de la Nota del Equipo #1:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **DOCUMENTACIÓN ENTREGA 5: DISCIPLINA IMPLEMENTACION** | | | | **9,900** |
| **MODELO DE IMPLEMENTACION** | | | | **9,900** |
|  | ***Componente de Administración de Tours*** | | | ***6,100*** |
|  |  | Implementación de las clases interfaz (refieren a administrar los tours) utilizando las facilidades de NetBeans | | 1,400 |
|  |  |  | Correspondencia entre el codigo y el diagrama de clase de diseño | 0,200 |
|  |  | implementación de atributos y métodos en las clases entidad (Tour y Tours) y la implementación de la clase de control para administrar los tour (puede ser la implementación 1:N entre la clase control y Tour si no está la clase Tours o la implementación de la asociación con la clase Tours). Implementación utilizando el patron Singleton en las clases de control y Tours tambien | | 1,400 |
|  |  |  | Correspondencia entre el codigo y el diagrama de clase de diseño | 0,300 |
|  |  | implementación del método crear tour | | 1,500 |
|  |  | Implementación del metodo modificar tour para agregar un punto de interés a la secuencia | | 1,300 |
|  |  | Implementación del método modificar tour para eliminar un punto de interés de la secuencia del tour | | 0,800 |
|  |  | Implementación del método eliminar tour | | 1,000 |
|  | ***Componente de Recorrido*** | | | ***3,400*** |
|  |  | implementación de las clases interfaz (correspondientes a consultar tours disponibles y recorrer las obras asociadas a los puntos interés del tour) utilizando las facilidades de NetBeans | | 1,500 |
|  |  |  | Correspondencia entre el codigo y el diagrama de clase de diseño | 0,200 |
|  |  | implementación de atributos y métodos (consultar los tour disponibles, avanzar, visitar) en la clase entidad (Recorrido) y la implementación de la clase de control para el orden de despliegue de las clases interfaz y re-transmitir los envíos de mensajes a la clase Recorrido. Implementación utilizando el patron Singleton | | 1,400 |
|  |  |  | Correspondencia entre el codigo y el diagrama de clase de diseño | 0,300 |
|  | **Presentación de resultados en la especificación de los casos de prueba** | | | **0,400** |

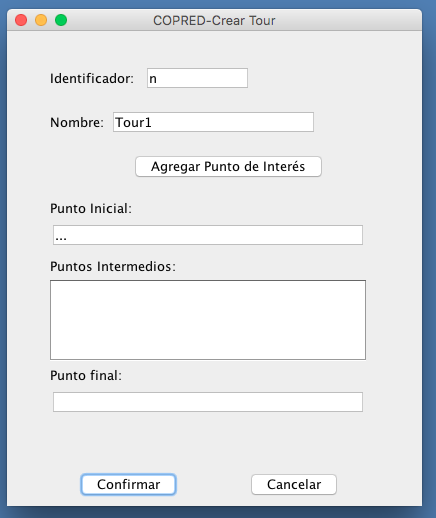
## Revisión de ejecución

El código esta operativo, con algunos detalles que se encontraron, a saber:

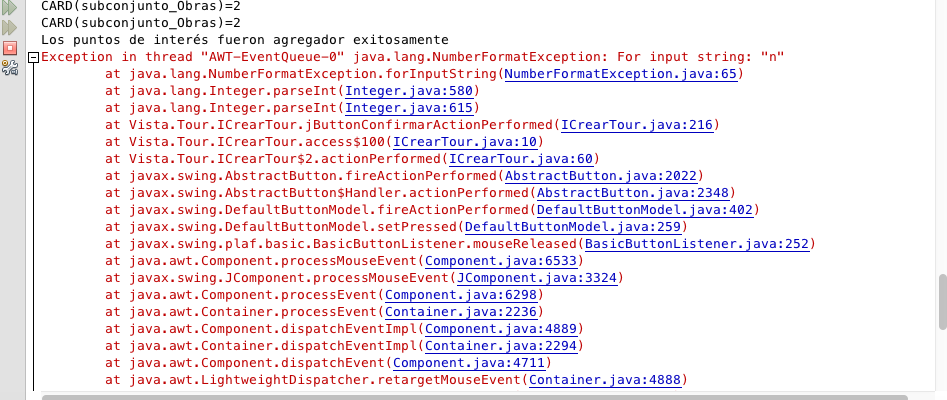
1. **Pantalla de Administrar Comité**: El botón de administrar tour debió estar de ultimo, primero administrar obras, luego punto de interés y de ultimo tour. Esto debido a que se necesitan puntos de interés.



1. **Crear Tour**:
   1. Cuando escojo los puntos de interés por primera vez, en el primer tour que se va a crear, al volver a la pantalla de Crear Tour, no muestra la información correctamente, pone solo unos puntos. Esto no ocurre si quiero crear un segundo Tour



* 1. Si coloco un carácter no numérico en el identificador da un error, no muestra mensaje de error y no crea el tour, dejando sin saber que pasa al usuario. Puedes cambiarlo por el numero y crear el tour, pero el usuario tiene que adivinar el error.



* 1. Luego del mensaje de error por no numérico, si creo un segundo tour con los tres puntos de interés correctos, el tour se crea correctamente, pero le pone disponibilidad “N”, cuando debería ser “S”. El resultado por consola es:

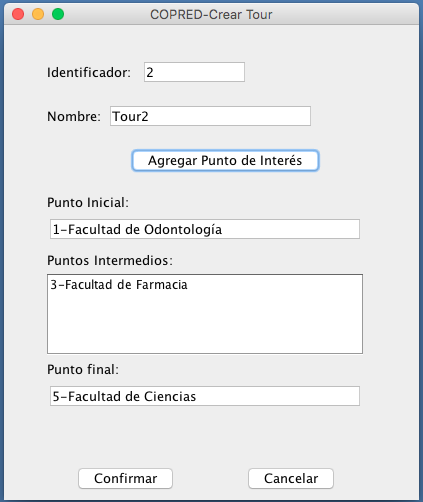
Los puntos de interés fueron agregador exitosamente

CARD(conjunto\_Tour)=2

Disponibilidad(Tour)=N

Se ha creado el Tour exitosamente

Las imágenes al crear son:



* 1. La creación de un tercer tour le pasa exactamente lo mismo, se queda con disponibilidad “N” aunque este bien

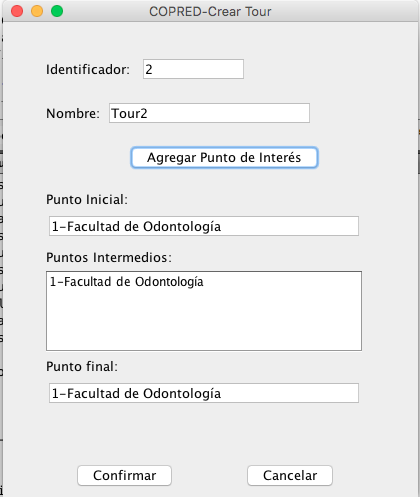
Los puntos de interés fueron agregador exitosamente

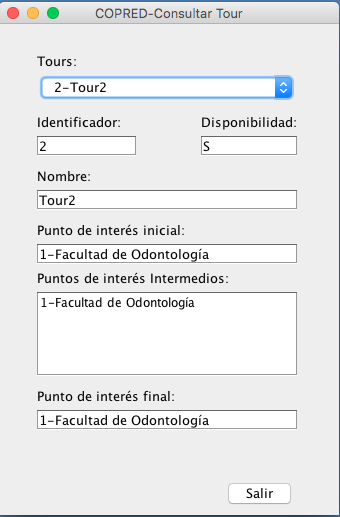
CARD(conjunto\_Tour)=3

Disponibilidad(Tour)=N

Se ha creado el Tour exitosamente

* 1. Para poder crear tours con la disponibilidad correcta hay que salir del programa y volver a crearlos, sin el error de numero en el identificador
  2. Cuando creas un tour te deja colocar el mismo punto de interés en varias posiciones, eso no es correcto.

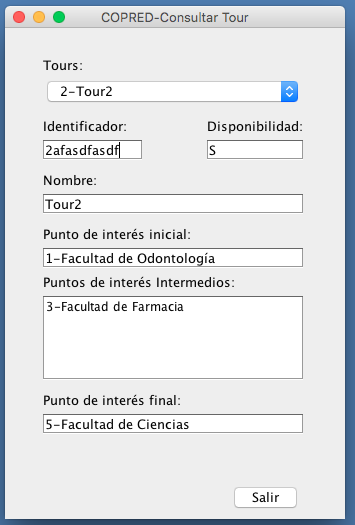




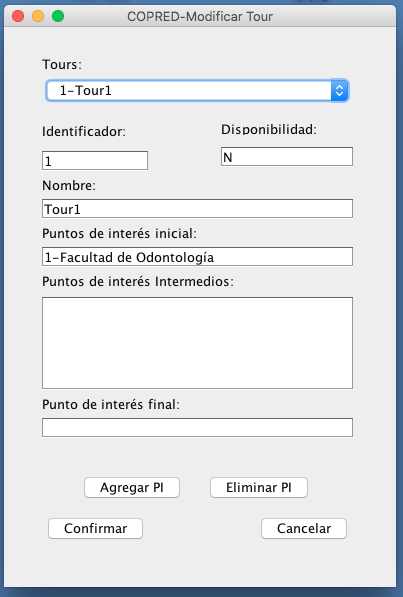
* 1. A veces da error la pantalla “Agregar punto de interés a un tour” da un error de NullPointException y ya no te deja agregar puntos. Cuando vuelves a la pantalla de crear tour y le das de nuevo al botón de agregar te dice que ya agregaste los puntos. Por lo tanto, no lo puedes crear. Si sales de esa pantalla y vuelves a entrar y creas un tour correctamente, lo hace bien, pero le pone la disponilibidad en “N”. En otras palabras, cada vez que se genera un error por debajo, la disponibilidad queda fija en “N” no importa si lo haces bien o no.

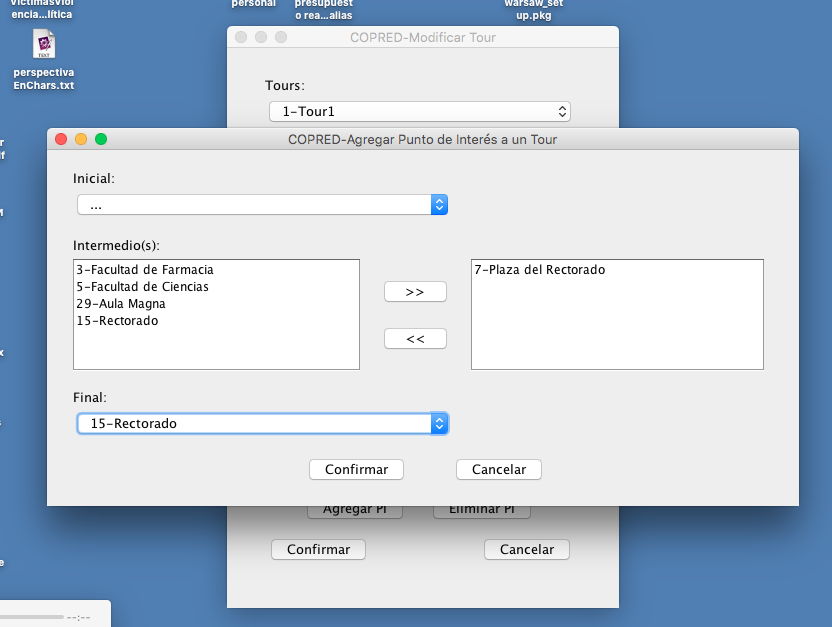


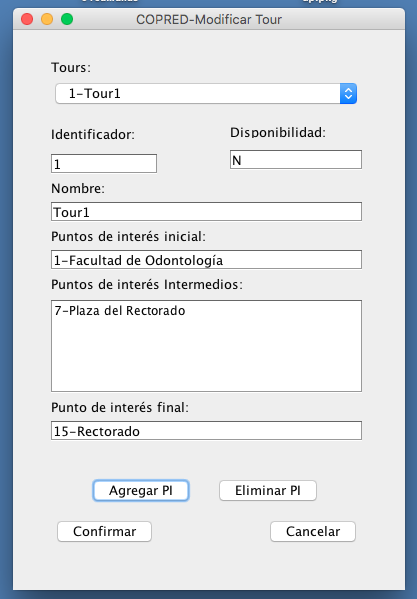
1. **Consultar Tour:**
   1. En esta pantalla ningún campo es modificable, por lo tanto debieron estar deshabilitados o ser solo texto

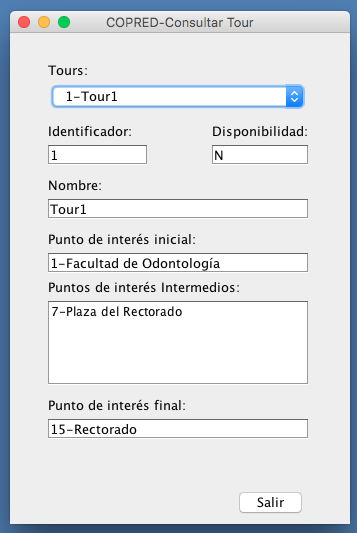


1. **Modificar Tour**:
   1. Si lo modifico un tour con disponibilidad “N” y le agrego los puntos de interés faltantes nunca cambia a disponibilidad “S”. Además, los botones no pueden llamarse “Agregar PI” y “Modificar PI”, PI no le dice nada al usuario, debí ser “Punto de interés”

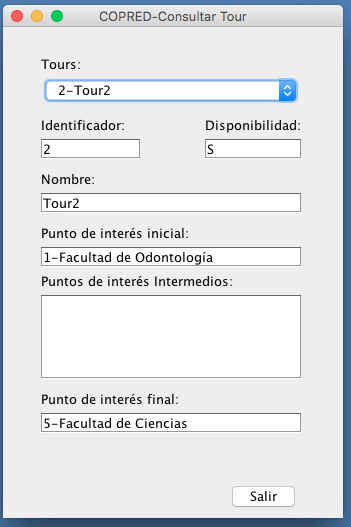




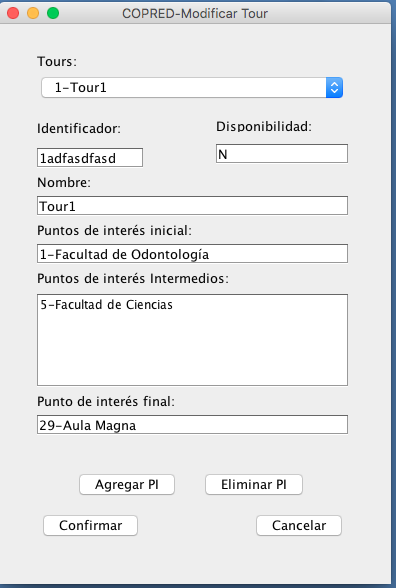




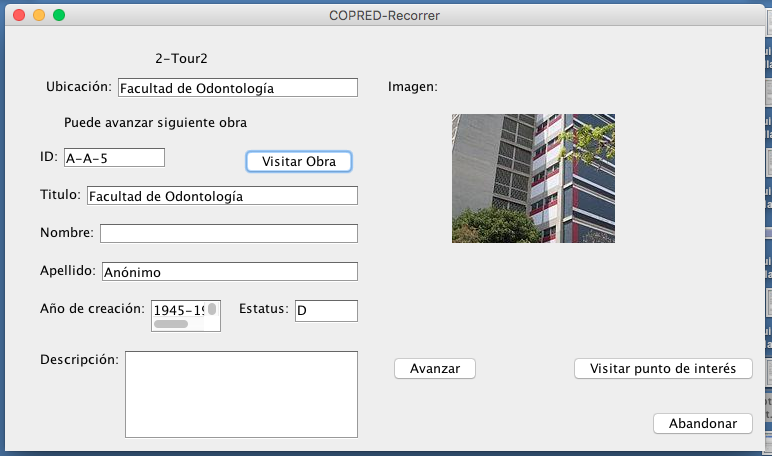
* 1. Si modifico un tour con disponibilidad “S” y le quito los tours intermedios, la disponibilidad no cambia a “N”



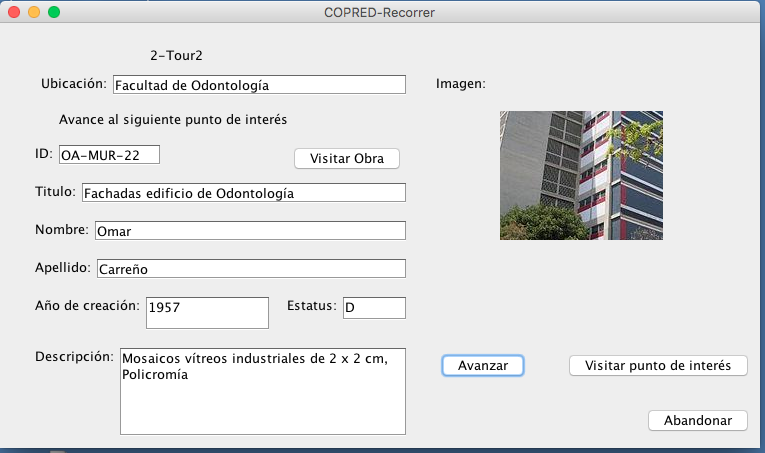
* 1. Los campos que no se modifican, como el identificador, el nombre y la disponibilidad NO pueden ser editables, confunde al usuario, se puede escribir (ver identificador en la imagen), pero eso no afecta al confirmar



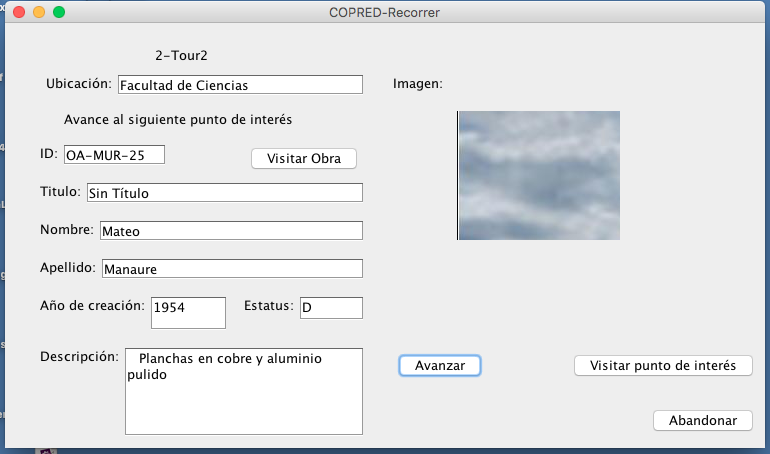
1. Recorrer tour
   1. El especio para la fecha no es suficiente, no se ve bien

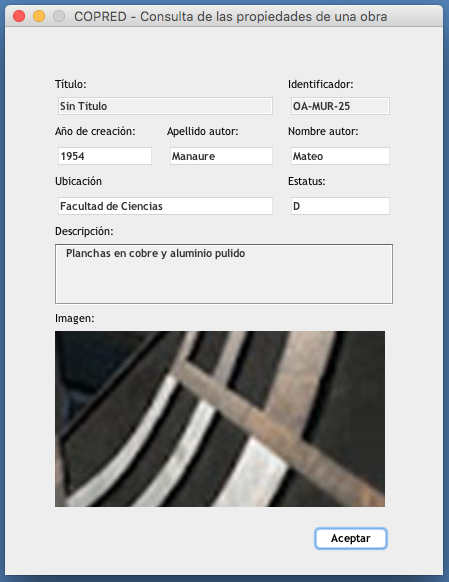


* 1. La pantalla se va adaptando a la información

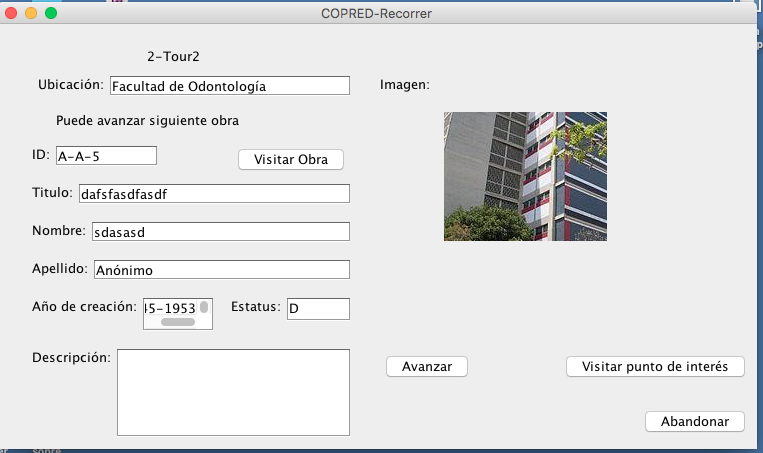


* 1. La imagen no cambia, se queda siempre la imagen de la primera obra del punto de interés

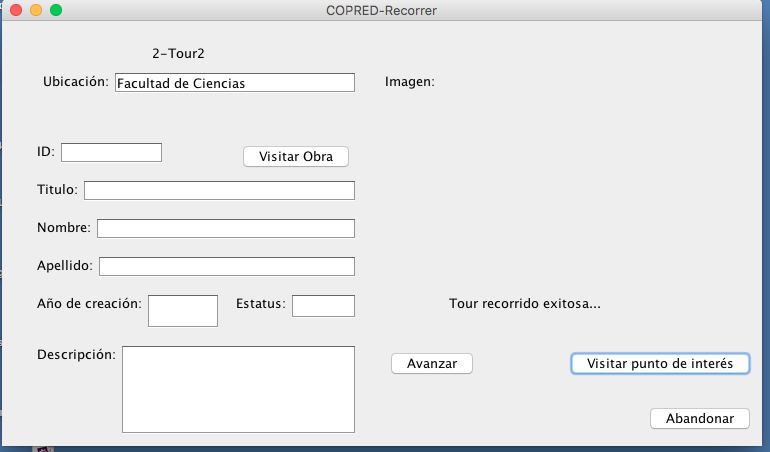




* 1. Los campos en recorrido deberían solo mostrar información, ustedes colocaron cajas de textos, el usuario puede modificar esa información, eso no debería ocurrir

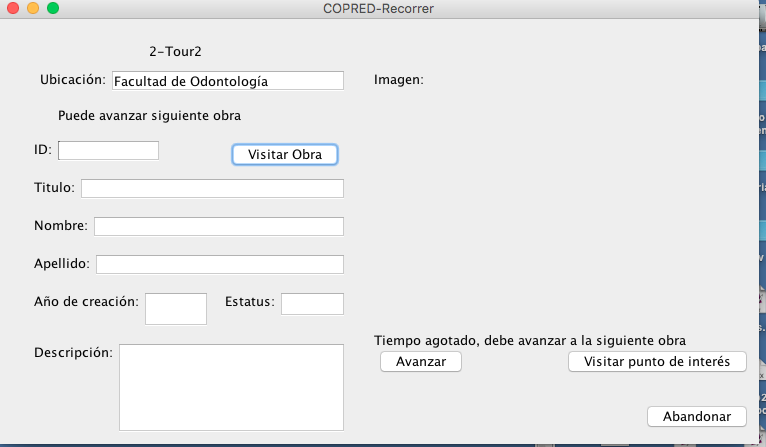


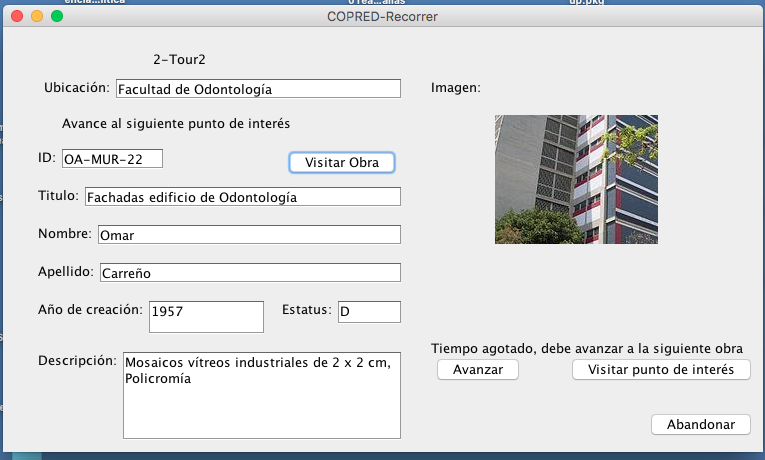
* 1. Hay un mensaje al final del recorrido que no esta en la descripción del UC, “Tour Recorrido exitosa”?. También hay otros mensajes que no coinciden con lo que pusieron, en la exposición hay que explicar por qué.



* 1. Cuando se agota el tiempo aparece el mensaje de avanzar a siguiente obra, pero te borra todos los campos. Luego si le das al botón avanzar, el mensaje se queda fijo.

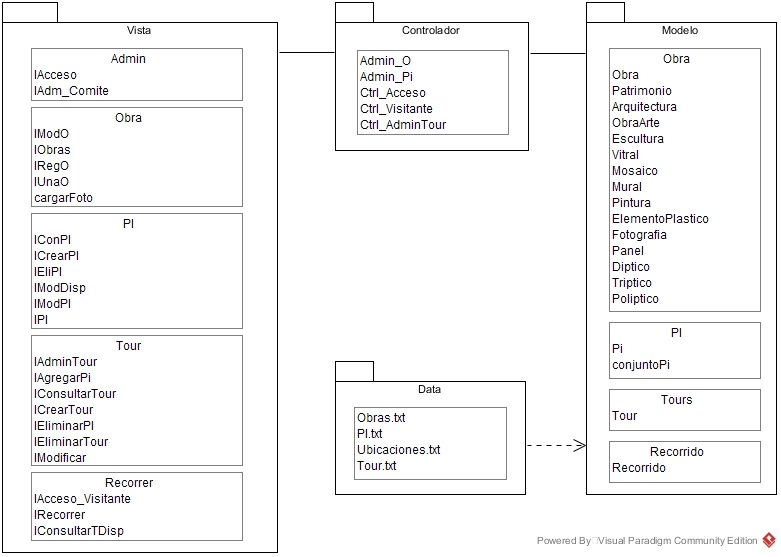






## Revisión de Código

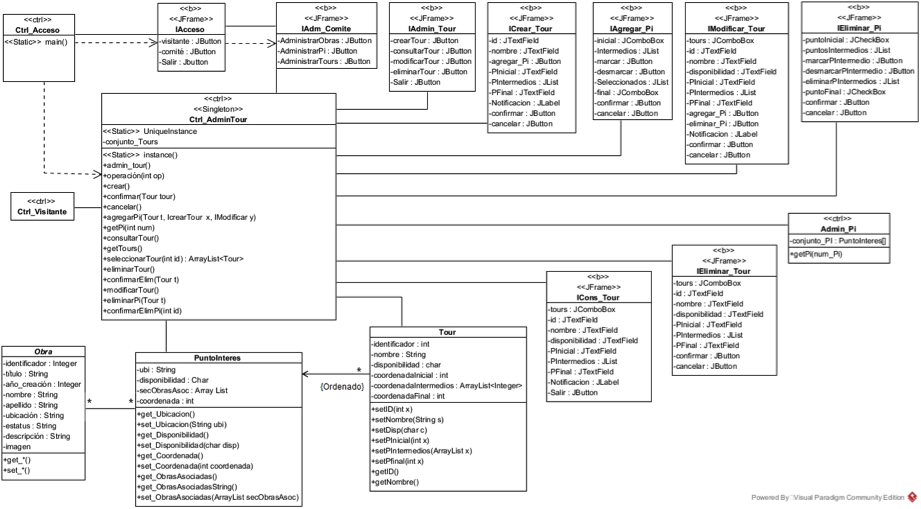
1. Concordancia diagrama de paquete:



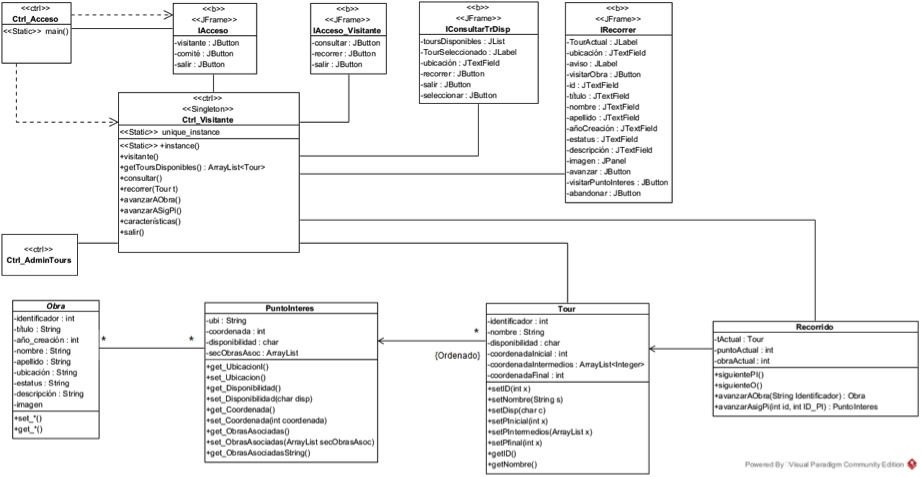
El código java tiene clasificadas sus clases tal cual el diagrama de paquete, solo unos detalles:

* 1. La clase IConsultarTrDisp tiene mal el nombre en este diagrama IConsutlarTDisp (falta la r)
  2. En el paquete Data falta la clase Data.java
  3. En el paquete Modelo, el sub paquete es Tour y no Tours
  4. El sub paquete Recorrido de Modelo no esta dentro de esa carpeta, esta afuera, un paquete llamado Recorrido

1. Concordancia diagrama de clases de diseño:



* 1. En IAdmin\_Comite falto el botón salir
  2. En la clase Ctrl\_Admin\_Tours hay un método getTours() que no esta en el código. También faltan todos los retornos de los métodos y los tipos del atributo. Los métodos guardarTour(), cargarTours() están en el código pero no en el diagrama de clases. Por último, en esta clase hay mucho métodos vacíos que no tienen código, no se para que están, debieron borrarlos. En esta clase tampoco esta la relación con la clase PuntoInteres. La relación de esta clase con Admin\_PI no existe en el código.
  3. En la clase tour faltan los métodos: getDips, getPInicial, getPIntermedios, getPFinal
  4. Todas las clases Interfaz tienen nombres distintos, en el diagrama de clases llevan un “\_” que no lo tienen en el código. Además, la de IModificar\_Tour se llama IModificar.
  5. La clase ICrearTour se relaciona con Modelo.Tour.Tour y Modelo.PI.\*, y NO debería hacerlo, era una restricción del diagrama de clases que no se implemento.
  6. La clase IAgregarPI se relaciona con Modelo.Tour.Tour y Modelo.PI.\*, y NO debería hacerlo, era una restricción del diagrama de clases que no se implemento. También se relaciona con las clases IModificar y ICrearTour y eso no esta reflejado en el diagrama de clases.
  7. La clase IModificar se relaciona con Modelo.Tour.Tour y Modelo.PI.\*, y NO debería hacerlo, era una restricción del diagrama de clases que no se implemento.
  8. La clase IModificar se relaciona con Modelo.Tour.Tour y Modelo.PI.PuntoInteres, y NO debería hacerlo, era una restricción del diagrama de clases que no se implemento.
  9. En la clase IEliminarTour falto el campo “Notificar” y se relaciona con Modelo.Tour.Tour y Modelo.PI.PuntoInteres, y NO debería hacerlo, era una restricción del diagrama de clases que no se implemento.
  10. La clase IConsultarTour tiene en el diagrama de clases un campo “Notificar” que no esta en el código, y se relaciona con Modelo.Tour.Tour y NO debería hacerlo, era una restricción del diagrama de clases que no se implemento.



* 1. En el código la clase Ctrl\_Visitante tiene el método getPI() que no esta en el diagrama de clases. El método avanzarAObra() del diagrama de clases no esta en el código. La relación con la clase Recorrido no esta en el código y falta en el diagrama de clases la relación con Modelo.PI.ConjuntoPI
  2. En el diagrama de clases le faltan atributos a la clase Recorrido, además faltan los métodos getPActual, getObras, existeSiguiente, existePi. Además, el nombre del método siguienteObra esta diferente en el diagrama de clases, y los métodos avanzarAObra y avanzaAsigPI no están en el código. Por otro lado la relación de Recorrido con ConjuntoPI no esta representado en el diagrama de clases.
  3. La clase IConsultarTrDisp tiene relación con Modelo.Tour.Tour y NO debería hacerlo, era una restricción del diagrama de clases que no se implemento.
  4. La clase IRecorrer tiene relación con Modelo.Obra.\*, Modelo.Tour.\*, Modelo.PI.\* y Recorrido.Recorrido y NO debería hacerlo, era una restricción del diagrama de clases que no se implemento.

1. Código como tal:
   1. En las clases singleton les falto hacer privadas las constructoras, así como esta ahora implementado cualquiera puede hacer new de esas clases y no cumple con singleton
   2. Las clases de Interfaz se están relacionando con las de Modelo y eso NO se puede, era una restricción de implementación
   3. Al código le faltó documentación, sobre todo explicar un poco el funcionamiento de los métodos.

## Revisión de las plantillas de prueba

1. La prueba número 3 era sobre el método seleccionarTour de Ctrl\_AdminTour, ese método esta vacío en la clase, no hace nada y no retorna ningún ArrayList. Supongo que la prueba la hicieron sobre la clase IConsultarTour y de ahí sacaron lo que se imprime en la consola. Debieron actualizar la plantilla de prueba o en realidad arreglar el código, por que esta mal que esa clase lo haga.
2. La prueba número 4, los parámetros no están según el código, en el código están (Tour t,ICrearTour x, IModificar y). Luego en el parte de casos de prueba ponen (1,” Tour1”, -12, null, 5), no coincide nada. En Resultado también falta imprimir la cardinalidad de conjunto\_Tours para coincidir con lo que dice la parte de “Valor esperado”. Por otro lado, el método que esta imprimiendo lo que ponen en Resultado es jButtonConfirmarActionPerformed de la clase IAgregarPI.
3. La prueba número 5, los parámetros no están según el código, en el código dice confirmarElimPi(int id) y además es un método vacío. Esta pasando lo mismo que en la prueba anterior. Lo que se imprime a consola esta en otra clase. También falta colocar en la prueba la impresión de la cardinalidad que se indica en valor esperado.
4. En la prueba 7 pasa lo mismo con el método que en las dos pruebas anteriores.
5. En la prueba 8 se hace referenci a un método de la clase Ctrl\_Visitante que no existe en la clase, esta pasando lo mismo que en las otras pruebas. Falta el resultado de algunas pruebas. Es más, no consigo ningún método de ese código que imprima “El punto de interés no tiene más obras avance al siguiente punto de interés” y retorne el objeto Obra, como lo especifican ahí.