**Universidad Central de Venezuela**

**Facultad de Ciencias**

**Escuela de Computación**

**Ingeniería de Software**

**Semestre 1-2017**

**Proyecto**

**Entrega 4:**

**Disciplina de Prueba**

**Equipo 1**

Integrantes:

Alejandra Vento

Laura Zambrano

Patrizia Guarente

Jesús Martínez

Jesús Pacheco

# Modelo de Prueba

Diagrama de clases de diseño de comité:

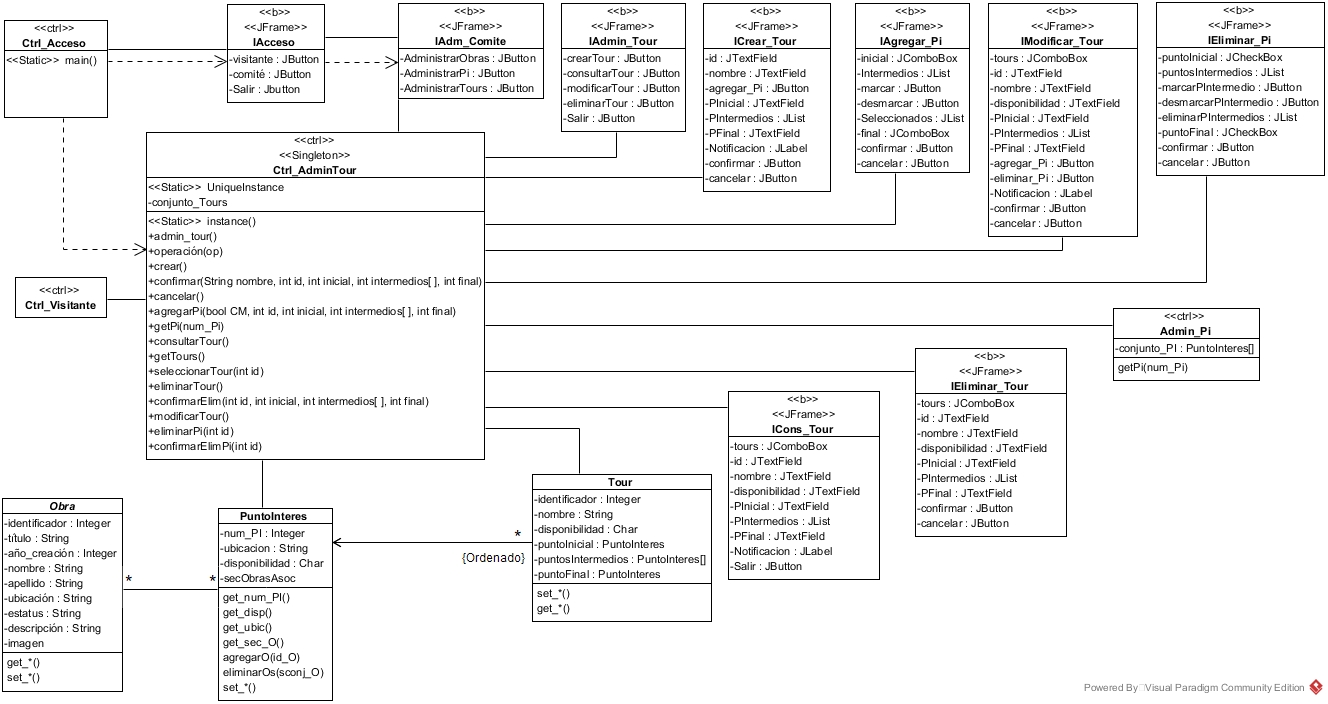
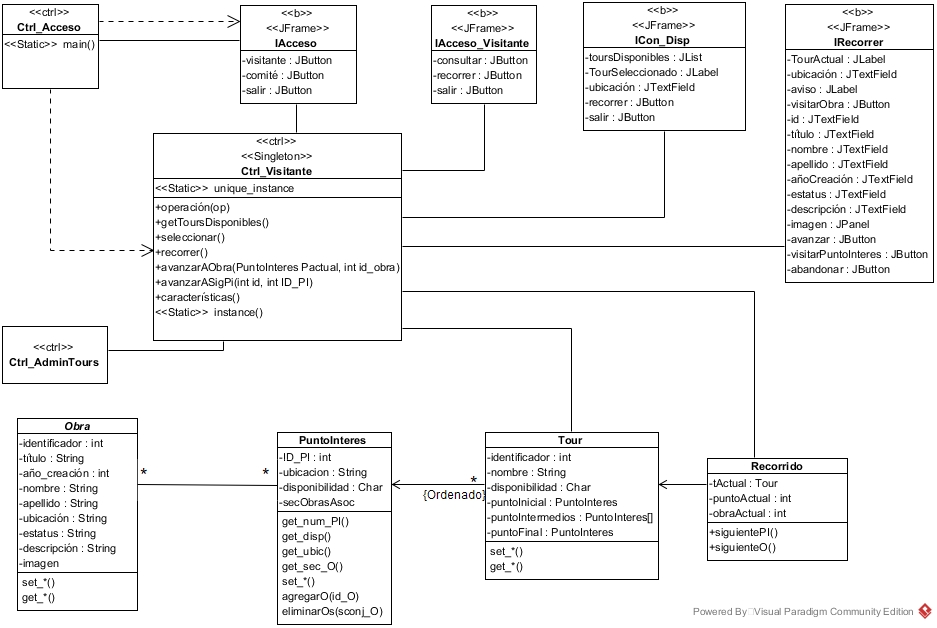


Diagrama de clases de diseño de visitante:



## Especificación de los casos de prueba

**Id Prueba**: 1

**Tipo de Prueba**: Unitaria

**Descripción**: Se crea un tour con sus datos como parámetros recibidos por la función, siempre y cuando el ID dado no coincida con el ID de un tour ya creado

**Clase**: Ctrl\_AdminTour

**Método**: Confirmar(String nombre, int id, int inicial, int intermedios[ ], int final) **Tipo de retorno**: void

**Pre-condición**: -.

**Post-condición**: CARD(conjunto\_Tours) = CARD(conjunto\_Tours)’+1, CARD(conjunto\_Tours) > 0

**Casos de prueba**: Sea T ⊆D y T ={(t1), (t2), (t3), (t4), (t5)};

t1= (“Tour1”, 1, 1, {3,29,7}, 5),

t2= (“Tour2”, 1, 15, 29, 1),

t3= (“Tour2”, 2, null, {5,7,15}, 29),

t4= (“Tour3”, 3, 1, null, 5),

t5= (“Tour4”, 4, 5, {15,29}, null)

t6= (“Tour5”, -2, 5, {15,29}, 3)

donde t1 es un caso de prueba para ID de tour válido con Disponibilidad(Tour)=S, t2 es un caso de prueba para ID de tour repetido, t3, t4 y t5 son casos de prueba de tour con Disponibilidad(Tour)=N y t6 es un caso de prueba para ID de tour inválido. Además, para t1 CARD(conjunto\_Tours) =0 y Conjunto\_PI= {1,3,5,7,29,15}

**Valor esperado**:

Para T = (t1):

(confirmar (“Tour1”,1, 1, {3,29,7}, 5) :: CARD(Tours) = 1, Disponibilidad(Tour) = S.

Se imprime por consola CARD(conjunto\_Tours), Disponibilidad(Tour) y el mensaje: “Se ha creado el Tour exitosamente”)

Para T = (t2)

(confirmar (“Tour2”, 1, 15, 29, 1) :: CARD(Tours) = 1.

Se imprime por consola CARD(conjunto\_Tours) y el mensaje: “Identificador ya existe”)

Para T = (t3):

(confirmar (“Tour2”, 2, null, {5,7,15}, 29) :: CARD(conjunto\_Tours) = 2, Disponibilidad(Tour) = N.

Se imprime por consola CARD(conjunto\_Tours), Disponibilidad(Tour) y el mensaje: “Se ha creado el Tour exitosamente”)

Para T = (t4):

(confirmar (“Tour3”, 3, 1, null, 5) :: CARD(conjunto\_Tours) = 3, Disponibilidad(Tour) = N.

Se imprime por consola CARD(conjunto\_Tours), Disponibilidad(Tour) y el mensaje: “Se ha creado el Tour exitosamente”)

Para T = (t5):

(confirmar (“Tour4”, 4, 5, {15,29}, null) :: CARD(conjunto\_Tours) = 4, Disponibilidad(Tour) = N.

Se imprime por consola CARD(conjunto\_Tours), Disponibilidad(Tour) y el mensaje: “Se ha creado el Tour exitosamente”)

Para T = (t6):

(confirmar (“Tour5”, -2, 5, {15,29}, 3) :: CARD(conjunto\_Tours) = 4.

Se imprime por consola CARD(conjunto\_Tours) y el mensaje: “ID inválido”)

**Resultado**:

Serán agregados en la siguiente entrega, donde se implementará el código y se ejecutaran las pruebas para obtener los resultados que se anexaran aquí

**Id Prueba**: 2

**Tipo de Prueba**: Unitaria

**Descripción**: Se elimina el Tour seleccionado del conjunto Tours siempre y cuando CARD(conjunto\_Tours) > 0

**Clase**: Ctrl\_AdminTour

**Método**: confirmarElim (int id) **Tipo de retorno**: void

**Pre-condición**: CARD(conjunto\_Tours) > 0

**Post-condición**: CARD(conjunto\_Tours)=CARD(conjunto\_Tours)’-1, CARD(conjunto\_Tours)≥0

**Casos de prueba**: Sea T ⊆D y T ={(t1), (t2)};

t1= (1)

t2= (1)

donde t1 es un caso de prueba de eliminar un tour válido y t2 es un caso de prueba de eliminar un tour inválido. Además, para t1 CARD(Tours) > 0

**Valor esperado**:

Para T = (t1):

(confirmarElim (1) :: CARD(conjunto\_Tours) = 3

Se imprime por consola CARD(Tours), el conjunto de Tours y el mensaje: “Se ha eliminado el Tour exitosamente”)

Para T = (t2):

(confirmarElim (1) :: CARD(conjunto\_Tours) = 3

Se imprime por consola CARD(Tours) y el mensaje: “El Tour seleccionado no existe”). Si se utiliza la interfaz este caso de prueba no va a ocurrir

**Resultado**:

Serán agregados en la siguiente entrega, donde se implementará el código y se ejecutaran las pruebas para obtener los resultados que se anexaran aquí

**Id Prueba**: 3

**Tipo de Prueba**: Unitaria

**Descripción**: Se selecciona uno de los tours contenidos en el conjunto Tours para mostrar sus características, tales como: ID, Nombre, Punto de interés inicial, intermedios y final, siempre y cuando CARD(conjunto\_Tours) > 0

**Clase**: Ctrl\_AdminTour

**Método**: seleccionarTour(int id) **Tipo de retorno**: ArrayList

**Pre-condición**: CARD(conjunto\_Tours) ≥ 0

**Post-condición**: CARD(conjunto\_Tours) = CARD(conjunto\_Tours)

**Casos de prueba**: Sea T ⊆D y T ={(t1), (t2)};

t1= (1)

t2= (-12)

donde t1 es un caso de prueba de consultar un tour válido y t2 es un caso de prueba de consultar un tour inválido. Además, para t1 CARD(conjunto\_Tours) > 0

**Valor esperado**:

Para T = (t1):

(seleccionarTour (1):: CARD(conjunto\_Tours) = 3

Se imprime por consola ID, Nombre y los puntos de interés asociados al tour)

Para T = (t2):

(seleccionarTour (123) :: CARD(conjunto\_Tours) = 3

Se imprime por consola el mensaje: “El Tour seleccionado no existe”). Si se utiliza la interfaz este caso de prueba no va a ocurrir

**Resultado**:

Serán agregados en la siguiente entrega, donde se implementará el código y se ejecutaran las pruebas para obtener los resultados que se anexaran aquí

**Id Prueba**: 4

**Tipo de Prueba**: Unitaria

**Descripción**: Se asocian los puntos de interés seleccionados al tour previamente escogido

**Clase**: Ctrl\_AdminTour

**Método**: agregarPi (bool CM, int id, int inicial, int intermedios [], int final) **Tipo de retorno**: boolean

**Pre-condición**: -.

**Post-condición**: CARD(conjunto\_Tours)>0

**Casos de prueba**: Sea T ⊆D y T ={(t1), (t2), (t3), (t4), (t5), (t6)};

t1= (true,1,-12, null, 5)

t2= (true,3, 1, {3, -4, 7}, 29)

t3= (true,2, 15, {3, 7}, -9)

t4= (true,1, 501, 3, 7)

t5= (true,4, 1, {3,5}, 7)

donde t1, t2 y t3 son casos de prueba de un ID de punto de interés inválido, t4 es un caso de prueba de un ID de un punto de interés inexistente, t5 es un caso de prueba de un punto de interés válido. Además, para t5 CARD(conjunto\_PI) > 0 y Conjunto\_PI= {1,3,5,7,29,15}

**Valor esperado**:

Para T = (t1):

(agregarPi (1,1,-12, null, 5) :: CARD(conjunto\_Tours) = 3

Se imprime por consola el mensaje: “ID de punto de interés inválido”)

Para T = (t2):

(agregarPi (1,3, 1, {3, -4, 7}, 29) :: CARD(conjunto\_Tours) = 3

Se imprime por consola el mensaje: “ID de punto de interés inválido”)

Para T = (t3):

(agregarPi (1,2, 15, {3, 7}, -9) :: CARD(conjunto\_Tours) = 3

Se imprime por consola el mensaje: “ID de punto de interés inválido”)

Para T = (t4):

(agregarPi (1,1, 501, 3, 7) :: CARD(conjunto\_Tours) = 3

Se imprime por consola el mensaje: “El punto de interés no existe”)

Para T = (t5):

(agregarPi (1,4, 1, {3,5}, 7) :: CARD(conjunto\_Tours) = 3

Se imprime por consola el mensaje: “Los puntos de interés fueron agregados exitosamente”)

**Resultado**:

Serán agregados en la siguiente entrega, donde se implementará el código y se ejecutaran las pruebas para obtener los resultados que se anexaran aquí

**Id Prueba**: 5

**Tipo de Prueba**: Unitaria

**Descripción**: Se eliminan los puntos de interés seleccionados del tour previamente escogido

**Clase**: Ctrl\_AdminTour

**Método**: confirmarElimPi(int id, int inicial, int intermedios [ ], int final) **Tipo de retorno**: void

**Pre-condición**: CARD(conjunto\_Tours) > 0

**Post-condición**: CARD(conjunto\_Tours) > 0

**Casos de prueba**: Sea T ⊆D y T ={(t1), (t2), (t3), (t4), (t5), (t6)};

t1= (1,-12, null, 5)

t2= (3, 1, {3, -4, 7}, 29)

t3= (2, 15, {3, 7}, -9)

t4= (1, 501, 3, 7)

t5= (4, 1, {3,5}, 7)

t6= (5, null, null, null)

donde t1, t2 y t3 son casos de prueba de un ID de punto de interés inválido, t4 es un caso de prueba de un ID de un punto de interés inexistente, t5 es un caso de prueba de un punto de interés válido, t6 es un caso de prueba donde no se seleccionen puntos de interés previos. Además, para t5 y t6 CARD(conjunto\_PI) > 0 y Conjunto\_PI= {1,3,5,7,29,15}

**Valor esperado**:

Para T = (t1):

(confirmarElimPi (1,-12, null, 5) :: CARD(conjunto\_Tours) = 3

Se imprime por consola el mensaje: “ID de punto de interés inválido”)

Para T = (t2):

(confirmarElimPi (3, 1, {3, -4, 7}, 29) :: CARD(conjunto\_Tours) = 3

Se imprime por consola el mensaje: “ID de punto de interés inválido”)

Para T = (t3):

(confirmarElimPi (2, 15, {3, 7}, -9) :: CARD(conjunto\_Tours) = 3

Se imprime por consola el mensaje: “ID de punto de interés inválido”)

Para T = (t4):

(confirmarElimPi (1, 501, 3, 7) :: CARD(conjunto\_Tours) = 3

Se imprime por consola el mensaje: “El punto de interés no existe”)

Para T = (t5):

(confirmarElimPi (4, 1, {3,5}, 7) :: CARD(conjunto\_Tours) = 3

Se imprime por consola el mensaje: “Los puntos de interés fueron eliminados exitosamente”)

Para T = (t6):

(confirmarElimPi (5, null, null, null) :: CARD(conjunto\_Tours) = 3

Se imprime por consola el mensaje: “No han sido seleccionados los puntos de interés”)

**Resultado**:

Serán agregados en la siguiente entrega, donde se implementará el código y se ejecutaran las pruebas para obtener los resultados que se anexaran aquí

**Id Prueba**: 6

**Tipo de Prueba**: Unitaria

**Descripción**: Se obtienen todos los tours disponibles para recorre

**Clase**: Ctrl\_Visitante

**Método**: getToursDisponibles()                                                              **Tipo de retorno**: ArrayList

**Pre-condición**: CARD(Conjunto\_Tours) > 0

**Post-condición**: CARD(Conjunto\_Tours) > 0

**Casos de prueba**: Sea T ⊆D y T ={(t1)};

t1= ()

donde t1 es un caso de prueba que obtiene los tours disponibles para recorrer

**Valor esperado**:

Para T = (t1):

(getToursDisponibles() :: CARD(conjunto\_Tours) = 3, CARD(conjunto\_ToursDisp) = 2

Se imprime por consola CARD(conjunto\_Tours) y CARD(conjunto\_ToursDisp)

**Resultado**:

Serán agregados en la siguiente entrega, donde se implementará el código y se ejecutaran las pruebas para obtener los resultados que se anexaran aquí

**Id Prueba**: 7

**Tipo de Prueba**: Unitaria

**Descripción**: Se obtiene el siguiente punto de interés del tour para recorrer

**Clase**: Ctrl\_Visitante

**Método**: avanzarASigPi(int id, int ID\_PI) **Tipo de retorno**: PuntoInterés

**Pre-condición**: CARD(conjunto\_Tours) > 0

**Post-condición**: CARD(conjunto\_Tours) > 0

**Casos de prueba**: Sea T ⊆D y T ={(t1),(t2),(t3)};

t1= (1,1)

t2= (1,3)

t3= (1,6)

donde t1 es un caso de prueba que obtiene el siguiente punto de interés el cual tiene Disponibilidad(pi) = N por lo que debe obtener el próximo punto de interés con Disponibilidad(pi) = S, t2 es un caso de prueba que obtiene el siguiente punto de interés el cual tiene Disponibilidad(pi) = S y t3 es un caso de prueba de punto de interés inexistente.

**Valor esperado**:

Para T = (t1):

(lista(tour1,1) :: CARD(conjunto\_Tours) = 3

Se imprime por consola el mensaje: “El punto de interés no está disponible, avanzar al siguiente punto de interés”. Se avanza al próximo punto de interés del tour hasta hallar uno disponible y se imprime por consola el punto de interés para recorrer)

Para T = (t2):

(lista (tour2,3) :: CARD(conjunto\_Tours) = 3

Se imprime por consola el punto de interés para recorrer)

Para T = (t3):

(lista (tour3,5) :: CARD(conjunto\_Tours) = 3

Se imprime por consola el mensaje: “el punto de interés no existe”)

**Resultado**:

Serán agregados en la siguiente entrega, donde se implementará el código y se ejecutaran las pruebas para obtener los resultados que se anexaran aquí

**Id Prueba**: 8

**Tipo de Prueba**: Unitaria

**Descripción**: Se avanza a la siguiente obra disponible del punto de interés asociado al tour que se esté recorriendo, siempre y cuando el punto de interés posea más obras.

**Clase**: Ctrl\_Visitante

**Método**: avanzarAObra(PuntoInteres Pactual, int id\_obra) **Tipo de retorno**: Obra

**Pre-condición**: CARD(conjunto\_Tours) > 0

**Post-condición**: CARD(conjunto\_Tours) > 0

**Casos de prueba**: Sea T ⊆D y T ={(t1), (t2)};

t1= (Facultad de Odontología, “A-A-5”)

t2= (Rectorado, “OA-MUR-23”)

donde t1 es un caso de prueba para avanzar a la próxima obra válido y t2 es un caso de prueba de última obra del punto de interés.

**Valor esperado**:

Para T = (t1):

(avanzarAObra(Facultad de Odontologia, “A-A-5”):: CARD(conjunto\_Tours) = 3

Se avanza a la siguiente obra del punto de interés asociado al tour)

Para T = (t2):

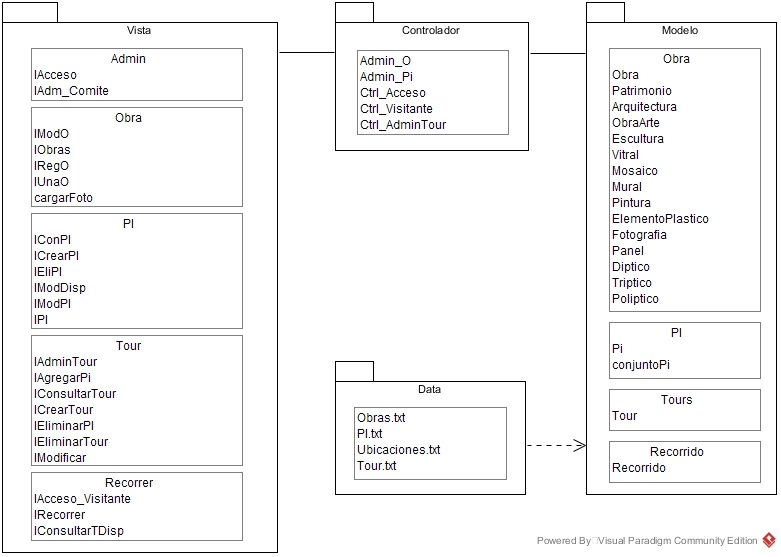
(avanzarAObra(“Rectorado, “OA-MUR-23”):: CARD(conjunto\_Tours) = 3

Se imprime por consola el mensaje: “El punto de interés no tiene más obras, avance al siguiente punto de interés”. Y se llama al método avanzarASigPi)

**Resultado**:

Serán agregados en la siguiente entrega, donde se implementará el código y se ejecutaran las pruebas para obtener los resultados que se anexaran aquí

## Diagrama de paquete



Se organizará el proyecto en paquetes siguiendo el criterio MVC (Modelo - vista - Controlador).

El paquete “Modelo” se subdividirá en 4 paquetes: Obra, PI, Tours y Recorrido, en los cuales se organizarán las clases relativas a las obras, puntos de interés, los tours y el recorrido respectivamente.

El paquete “Vista” se subdividirá en 5 paquetes: Admin, Obra, PI, Tour y Recorrer. En los paquetes “Obra”, “PI” y “Tour” se organizarán las clases relativas a las obras, puntos de interés y tours respectivamente. En el paquete “Admin” se encontrarán las interfaces relativas a la administración del sistema y en el paquete recorrido estarán las encargadas de suministrar al visitante las características de los tours disponibles, sus puntos de interés y sus obras.

Además, se encontrará el paquete “Data” el cual se contendrá los archivos encargados de la persistencia de datos.