Contenido

[Iteración 1: 3](#_Toc468896600)

[ABM Vendedor 3](#_Toc468896601)

[Vista y lógica del Login 3](#_Toc468896602)

[Entidad vendedor 3](#_Toc468896603)

[Vista alta, modificar y baja Vendedor 4](#_Toc468896604)

[Vista principal para la administración 4](#_Toc468896605)

[Lógica alta, modificación y baja vendedor 5](#_Toc468896606)

[Persistidor vendedor 6](#_Toc468896607)

[ABM Propietario 6](#_Toc468896608)

[Entidad propietario 6](#_Toc468896609)

[Vista alta, modificar y baja Propietario 7](#_Toc468896610)

[Lógica alta, modificación y baja propietario 7](#_Toc468896611)

[Persistidor propietario 8](#_Toc468896612)

[ABM Inmueble 9](#_Toc468896613)

[Entidad inmueble 9](#_Toc468896614)

[Clases de datos 10](#_Toc468896615)

[Vista alta, modificar y baja inmueble 10](#_Toc468896616)

[Lógica alta, modificación y baja inmueble 11](#_Toc468896617)

[Persistidor inmueble 12](#_Toc468896618)

[ABM Cliente 13](#_Toc468896619)

[Entidad cliente 13](#_Toc468896620)

[Vista alta, modificar y baja cliente 13](#_Toc468896621)

[Lógica alta, modificación y baja cliente 14](#_Toc468896622)

[Persistidor cliente 15](#_Toc468896623)

[Iteración 2: 17](#_Toc468896624)

[Consulta inmueble 17](#_Toc468896625)

[Vista de consulta inmueble 17](#_Toc468896626)

[Lógica y persistidor de consulta inmueble 17](#_Toc468896627)

[Catálogo de inmuebles 18](#_Toc468896628)

[Vista Alta Catálogo Inmueble 18](#_Toc468896629)

[Lógica alta catálogo de inmueble y generar PDF 19](#_Toc468896630)

[Generar reserva 19](#_Toc468896631)

[Vista alta, baja y listar reserva 19](#_Toc468896632)

[Lógica alta, baja reserva y generar PDF 20](#_Toc468896633)

[Lógica envío de mail 20](#_Toc468896634)

[Cambios ABM Cliente 21](#_Toc468896635)

[Persistidor y entidad reserva 21](#_Toc468896636)

[Ventas 22](#_Toc468896637)

[Vista alta y listar venta 22](#_Toc468896638)

[Lógica alta venta y generar PDF 23](#_Toc468896639)

[Lógica imprimir venta 23](#_Toc468896640)

[Persistidor y entidad venta 23](#_Toc468896641)

[Agregar EstadoInmueble a inmueble 24](#_Toc468896642)

# Iteración 1:

## ABM Vendedor

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Vista y lógica del Login | |  | Programador | 3 | |
|  |  |  |  |  |  |  |
| Fecha |  |  |  | Task Points | 2 |  |
|  |  |  |  |  |  |  |
| Número de Historia | | 1 |  | Número de TaskCard | | 1 |
|  |  |  |  |  |  |  |
| Descripción:  La opción Registrar inicia la actividad de Alta Vendedor.  Para ingresar al sistema se requieren los siguientes datos:   * tipoDocumento (TipoDocumento) * numeroDocumento (String:30) * Contraseña (String:100)   Se deben realizar todas las validaciones antes de permitir el ingreso.  La UI debe ser similar al spike realizado en la fase de exploración de la planificación | | | | | | |
|  |  |  |  |  |  |  |
| Notas | | | | | | |
|
|
| Fecha | Realizado | | A realizar | | Comentarios | |
| 17/10/2016 | Definición de pruebas vista | | Terminar vista  Definición de pruebas lógica  Terminar lógica autenticar | |  | |
| 24/10/2016 | Terminado función ingresar de la vista | | Terminar función entrar a registrar de la vista  Definición de pruebas lógica autenticar  Terminar lógica autenticar | | Terminar función entrar a registrar de la vista:  Se debe terminar la ventana de registrar vendedor primero. | |
| 26/10/2016 | Definición de pruebas lógica autenticar | | Terminar función entrar a registrar de la vista  Terminar lógica autenticar | | Deberán ser refinadas al terminar de implementar la lógica autenticar. | |
| 03/11/2016 | Terminar función entrar a registrar de la vista. Terminar lógica autenticar | | Nada | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Entidad vendedor | |  | Programador | 1 | |
|  |  |  |  |  |  |  |
| Fecha | 07/10/16 |  |  | Task Points | 1 |  |
|  |  |  |  |  |  |  |
| Número de Historia | | 1 |  | Número de TaskCard | | 2 |
|  |  |  |  |  |  |  |
| Descripción  Implementar la clase Vendedor. La clase debe tener los siguientes atributos: nombre(string), apellido(string), tipoDocumento(TipoDocumento), numeroDocumento(string:30), id(Integer), password(string:100), salt(String), Root(Boolean), estado(Estado) y ventas(ArrayList<Venta>).  Para cada atributo deben hacerse los métodos set y get. | | | | | | |
|
|
|  |  |  |  |  |  |  |
| Notas | | | | | | |
|
|
| Fecha | Realizado | | A realizar | | Comentarios | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Vista alta, modificar y baja Vendedor | |  | Programador | 2 | |
|  |  |  |  |  |  |  |
| Fecha |  |  |  | Task Points | 3 |  |
|  |  |  |  |  |  |  |
| Número de Historia | | 1 |  | Número de TaskCard | | 3 |
|  |  |  |  |  |  |  |
| Descripción:  Se deberá presentar una vista donde se visualice todos los vendedores cargados en el sistema y se permita dar de alta, modificar o eliminar un vendedor.  Para dar de alta un vendedor, los campos a ingresar por el usuario son los siguientes:   * Nombre (string:30) * Apellido (string:30) * Tipo de documento (TipoDocumento) * Numero de documento (String) * Contraseña (String) * Repetir contraseña (String)   Todos los datos son obligatorios. Si ocurre un error en la validación de uno o más campos, la interfaz deberá mostrar el error y explicar brevemente que ha sucedido.  La interfaz para modificar un vendedor es la misma que la interfaz de alta, y además se visualiza un checkBox para permitir al usuario cambiar o no la contraseña. Se pueden modificar todos los campos. Se deben mostrar los datos del vendedor en los campos correspondientes.  La UI para la baja de un vendedor es un cartel avisando que el vendedor va a ser eliminado, y presentará las opciones para aceptar o no la eliminación.  Las UI deben ser similares a los spikes realizados en la fase de exploración de la planificación. | | | | | | |
|  |  |  |  |  |  |  |
| Notas | | | | | | |
|
|
| Fecha | Realizado | | A realizar | | Comentarios | |
| 28/10/16 | Vista alta vendedor | | Vista modificar, baja y administración | |  | |
| 31/10/16 | Vista modificar y baja vendedor | | Vista administración y transiciones entre vistas | |  | |
| 02/11/16 | Vista administración y transiciones entre vistas | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Vista principal para la administración | |  | Programador | 3 | |
|  |  |  |  |  |  |  |
| Fecha |  |  |  | Task Points | 2 |  |
|  |  |  |  |  |  |  |
| Número de Historia | | 1 |  | Número de TaskCard | | 4 |
|  |  |  |  |  |  |  |
| Descripción:  Se deben mostrar las acciones que tiene permitido un vendedor después de ingresar al sistema, estas son:   * Inmueble: alta, modificar, eliminar, consulta, generar reserva, vender. * Propietario: alta, modificar, eliminar. * Cliente: Alta, modificar, eliminar, ver catalogo de inmuebles * Vendedor: modificar, eliminar. * Publicar.   Si una acción no está permitida, se debe ocultar el botón.  Si ocurre un error de validación al realizar una acción se deberá mostrar un mensaje de error indicando que es lo que ha sucedido.  La UI debe ser similar al spike realizado en la fase de exploración de la planificación. | | | | | | |
|  |  |  |  |  |  |  |
| Notas | | | | | | |
|
|
| Fecha | Realizado | | A realizar | | Comentarios | |
| 03/11/2016 | Todo | | Nada | | No tiene pruebas porque solo muestra otras pantallas | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Lógica alta, modificación y baja vendedor | |  | Programador | 1 | |
|  |  |  |  |  |  |  |
| Fecha |  |  |  | Task Points | 4 |  |
|  |  |  |  |  |  |  |
| Número de Historia | | 1 |  | Número de TaskCard | | 5 |
|  |  |  |  |  |  |  |
| Descripción:  Se recibe de la vista un objeto vendedor.  Para el alta se deberá validar que el tipo y número de documento no se repita con un vendedor ya registrado y el número de documento corresponda con el tipo de documento. El nombre y apellido se deberá validar que sean solo letras y tamaño máximo 30 caracteres. Si todo es correcto deberá dar de alta el vendedor en la base de datos. Si algo no es correcto devuelve una excepción.  Para la modificación se permitirá cambiar cualquier campo y se deberán validar los campos de la misma forma. Si todo es correcto se deberá modificar el vendedor en la base de datos.  Para la baja, no se deben tener consideraciones especiales y se debe realizar la baja del vendedor en la base de datos. | | | | | | |
|  |  |  |  |  |  |  |
| Notas | | | | | | |
|
|
| Fecha | Realizado | | A realizar | | Comentarios | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Persistidor vendedor | |  | Programador | 1 | |
|  |  |  |  |  |  |  |
| Fecha |  |  |  | Task Points | 3 |  |
|  |  |  |  |  |  |  |
| Número de Historia 1 | |  |  | Número de TaskCard | | 6 |
|  |  |  |  |  |  |  |
| Descripción:  Se debe hacer una interfaz con métodos que permitan:  - Guardar un vendedor (si resulta en error lanza excepción SaveUpdateException)  - Modificar un vendedor (si resulta en error lanza excepción SaveUpdateException)  Todas las excepciones mencionadas extienden a PersistenceException.  Se debe hacer una clase que implemente dicha interfaz mediante hibernate y completar las entidades con las anotaciones necesarias. | | | | | | |
|
|
|  |  |  |  |  |  |  |
| Notas | | | | | | |
|
|
| Fecha | Realizado | | A realizar | | Comentarios | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |

## ABM Propietario

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Entidad propietario | |  | Programador | 2 | |
|  |  |  |  |  |  |  |
| Fecha | 07/10/16 |  |  | Task Points | 1 |  |
|  |  |  |  |  |  |  |
| Número de Historia | | 2 |  | Número de TaskCard | | 7 |
|  |  |  |  |  |  |  |
| Descripción  Implementar la clase Propietario. La clase debe tener los siguientes atributos: nombre(string:100), apellido(string:100), tipoDocumento(TipoDocumento), numeroDocumento(string:30), id(Integer), dirección(Direccion), teléfono(string:30), email(string:30), estado(Estado) e inmuebles(ArrayList<Inmueble>).  Para cada atributo deben hacerse los métodos set y get. | | | | | | |
|
|
|  |  |  |  |  |  |  |
| Notas | | | | | | |
|
|
| Fecha | Realizado | | A realizar | | Comentarios | |
| 20/10/16 | atributos, setters y getters, equals | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Vista alta, modificar y baja Propietario | |  | Programador | 1 | |
|  |  |  |  |  |  |  |
| Fecha |  |  |  | Task Points | 3 |  |
|  |  |  |  |  |  |  |
| Número de Historia | | 2 |  | Número de TaskCard | | 8 |
|  |  |  |  |  |  |  |
| Descripción:  Los campos a ingresar por el usuario son los siguientes:   * Nombre (string:30) * Apellido (string:30) * Tipo de documento (ComboBox<TipoDocumento>) * Número de documento (String) * Dirección:   + Calle (string:30)   + Número (String:30)   + Piso (String:30)   + Departamento (string:10)   + Localidad (string:30)   + Provincia (string:30) * Teléfono (String) * Email (string:30)   Los siguientes datos son obligatorios: nombre, apellido, tipo, número de documento, calle, número, localidad, provincia, teléfono.  Si ocurre un error en la validación de uno o más campos, la interfaz deberá mostrar el error y explicar brevemente que ha sucedido.  La interfaz para modificar un propietario es la misma que la interfaz de alta, se pueden modificar todos los campos. Se deben mostrar los datos del propietario en los campos correspondientes.  La UI para la baja de un propietario es un cartel avisando que el propietario va a ser eliminado, y presentará las opciones para aceptar o no la eliminación.  Las UI deben ser similares a los spikes realizados en la fase de exploración de la planificación. | | | | | | |
|  |  |  |  |  |  |  |
| Notas | | | | | | |
|
|
| Fecha | Realizado | | A realizar | | Comentarios | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Lógica alta, modificación y baja propietario | |  | Programador | 2 | |
|  |  |  |  |  |  |  |
| Fecha |  |  |  | Task Points | 3 |  |
|  |  |  |  |  |  |  |
| Número de Historia | | 2 |  | Número de TaskCard | | 9 |
|  |  |  |  |  |  |  |
| Descripción:  Para el alta se recibe de la vista un objeto propietario y se debe validar que:   * Nombre sea un string de letras con un máximo 30 caracteres. * Apellido sea un string de letras con un máximo 30 caracteres. * Tipo de documento sea uno de los tipos definidos. * Número de documento sea numérico y corresponda con el tipo de documento. * Dirección:   + Calle sea un string de como máximo 30 caracteres alfanuméricos.   + Número sea String de como máximo 10 caracteres.   + Piso sea String de como máximo 10 caracteres.   + Departamento sea un string de como máximo 10 caracteres.   + Localidad sea un string de como máximo 30 caracteres.   + Provincia sea un string de como máximo 30 caracteres. * Teléfono sea un String de como máximo 10 caracteres. * Email sea un string con formato de correo.   Los siguientes datos son obligatorios: nombre, apellido, tipo, número de documento, calle, número, localidad, provincia, teléfono.  Si todo es correcto se da de alta al propietario en la base de datos, si no se devuelve una excepción  Para la modificación se permitirá cambiar cualquier campo, se deberán realizar las mismas validaciones y si todo es correcto se deberá modificar el propietario en la base de datos. Si algo no es correcto se deberá devolver una excepción.  Para la baja, no se deben tener consideraciones especiales y se debe realizar una baja lógica del propietario.  Se debe implementar un método que permita obtener todos los propietarios de la base de datos.  Debe implementarse métodos para la validación de los campos, pero éstos deben ser privados. | | | | | | |
|  |  |  |  |  |  |  |
| Notas | | | | | | |
|
|
| Fecha | Realizado | | A realizar | | Comentarios | |
| 21/10/16 | Logica de Alta y modificacion | | Baja de un propietario, validar datos | | Se realizaron los casos de prueba antes de codificar | |
| 22/10/16 | Validar datos | | Baja de un propietario | |  | |
| 02/11/16 | Baja de un propietario | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Persistidor propietario | |  | Programador | 2 | |
|  |  |  |  |  |  |  |
| Fecha |  |  |  | Task Points | 3 |  |
|  |  |  |  |  |  |  |
| Número de Historia | | 2 |  | Número de TaskCard | | 10 |
|  |  |  |  |  |  |  |
| Descripción  Se debe hacer una interfaz con métodos que permitan:  - Guardar un propietario (si resulta en error lanza excepción SaveUpdateException)  - Modificar un propietario (si resulta en error lanza excepción SaveUpdateException)  - Obtener un propietario (si resulta en error lanza excepción ConsultaException)  - Obtener todos los propietarios (si resulta en error lanza excepción ConsultaException)  Todas las excepciones mencionadas extienden a PersistenceException.  Se debe hacer una clase que implemente dicha interfaz mediante hibernate y completar las entidades con las anotaciones necesarias. | | | | | | |
|
|
|  |  |  |  |  |  |  |
| Notas | | | | | | |
|
|
| Fecha | Realizado | | A realizar | | Comentarios | |
| 31/10/16 | Guardar y modificar un propietario | | Obtener un propietario y listar todos los propietarios | |  | |
| 02/11/16 | Obtener un propietario y listar todos los propietarios | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |

## ABM Inmueble

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Entidad inmueble | |  | Programador | 3 | |
|  |  |  |  |  |  |  |
| Fecha | 07/10/16 |  |  | Task Points | 1 |  |
|  |  |  |  |  |  |  |
| Número de Historia | | 3 |  | Número de TaskCard | | 11 |
|  |  |  |  |  |  |  |
| Descripción  Implementar la clase Inmueble. La clase debe tener los siguientes atributos: id(Integer), fechaCarga(Date), propietario(Propietario), tipo(TipoInmueble), precio(Double), orientacion(Orientacion) , frente(Double) , fondo(Double) , superficie(Double), datosEdificio(DatosEdificio), fotos(ArrayList<Imagen>), Observaciones (String:300), direccion(Direccion), estado(Estado).  Implementar la clase HistorialInmueble. La clase debe tener los siguientes atributos: id(long), fechaYHoraCambio(Date), fechaCarga(Date), propietario(Propietario), tipo(TipoInmueble), precio(Double), orientacion(Orientacion) , frente(Double) , fondo(Double) , superficie(Double), datosEdificio(DatosEdificio), fotos(ArrayList<Imagen>), Observaciones (String:300), direccion(Direccion), estado(Estado).  Implementar la clase DatosEdificio. La clase debe tener los siguientes atributos: id(Integer), propiedadHorizontal(bool), superficie(double), antigüedad(Integer), dormitorios(Integer), baños(Integer), garaje(bool), patio(bool), piscina(bool), aguaCorriente(bool), cloacas(bool), gasNatural(bool), aguaCaliente(bool), teléfono(bool), lavadero(bool), pavimento(bool).  Para cada atributo deben hacerse los métodos set y get. | | | | | | |
|
|
|  |  |  |  |  |  |  |
| Notas | | | | | | |
|
|
| Fecha | Realizado | | A realizar | | Comentarios | |
| 17/10/16 | Terminada Entidad Inmueble | | Terminar Entidad HistorialInmueble | | No tiene pruebas porque son solo getters y setters. | |
| 22/10/16 | Terminada Entidad HistorialInmueble | | Nada | | No tiene pruebas porque son solo getters y setters. | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Clases de datos | |  | Programador | 2 | |
|  |  |  |  |  |  |  |
| Fecha | 07/10/16 |  |  | Task Points | 1 |  |
|  |  |  |  |  |  |  |
| Número de Historia | | 3 |  | Número de TaskCard | | 12 |
|  |  |  |  |  |  |  |
| Descripción  Implementar las siguientes clases de datos:  La clase Direccion que tiene los atributos calle(Calle), numero(string:30), barrio(Barrio), piso(string:30), departamento(string:30), localidad(Localidad) y otros(string:100).  La clase Barrio que tiene los atributos id(Integer), nombre(string:50) y localidad(Localidad).  La clase Calle que tiene los atributos id(Integer), nombre(string:50) y localidad(Localidad).  La clase Localidad que tiene los atributos nombre(string:50), id(Integer) y provincia(Provincia).  La clase Provincia con los atributos nombre(string:50), id(Integer) y pais(Pais).  La clase País, con atributos nombre(string:50) y id(Integer).  La clase TipoDocumento tiene los atributos id(Integer) y tipo(enumerable: DNI, LC, LE, CÉDULA\_EXTRANJERA, PASAPORTE).  La clase TipoInmueble tiene los atributos id(Integer) y nombre(enumerable: L, C, D, T, Q, G).  La clase Orientacion tiene los atributos id(Integer) y orientacion(enumerable: NORTE, SUR, ESTE, OESTE, NORESTE, NOROESTE, SURESTE, SUROESTE).  Para cada atributo deben hacerse los métodos set y get. | | | | | | |
|
|
|  |  |  |  |  |  |  |
| Notas  Estas no son TODAS las clases necesarias. Pueden y se van a tener que hacer más. Indicar las clases agregadas y que no están listadas en esta Task Card. | | | | | | |
|
|
| Fecha | Realizado | | A realizar | | Comentarios | |
| 20/10/16 | Atributos, getters y getters, equals | | Anotaciones | | Agregadas clase: Imagen, Archivo, Estado. | |
| 21/10/16 | Anotaciones | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Vista alta, modificar y baja inmueble | | | |  | | | Programador | | 3 | | |
|  |  | | |  |  | | |  | |  | |  |
| Fecha |  | | |  |  | | | Task Points | | 3 | |  |
|  |  | | |  |  | | |  | |  | |  |
| Número de Historia | | | | 3 |  | | | Número de TaskCard | | | | 13 |
|  |  | | |  |  | | |  | |  | |  |
| Descripción:  Los campos a ingresar por el usuario son los siguientes:   * Propietario (Obligatorio) * País: por defecto Argentina (String:50) (obligatorio) * Provincia: por defecto Santa Fe (String:50) (obligatorio) * Localidad (String:50) (obligatorio) * Barrio (String:50) (obligatorio) * Calle/número (String:50) (obligatorio) * Piso/Departamento/otros (String:30) * Tipo de inmueble: [L/local-oficina, C/casa, D/departamento, T/terreno, Q/quinta, G/galpón] (obligatorio); * Precio de venta (Decimal) (obligatorio) * Orientación [norte, sur, este, oeste, noreste, noroeste, sureste, suroeste] * Medidas:   + Frente en metros (Decimal)   + Fondo en metros (Decimal)   + Superficie en m2 (Decimal) * Datos del edificio:   + Propiedad horizontal (Booleano)   + Superficie en m2 (Decimal)   + Antigüedad en años (Entero)   + Dormitorios (Entero)   + Baños (Entero)   + Garaje/cochera (Booleano)   + Patio (Booleano)   + Piscina (Booleano)   + Agua corriente (Booleano)   + Cloacas (Booleano)   + Gas natural (Booleano)   + Agua caliente (Booleano)   + Teléfono (Booleano)   + Lavadero (Booleano)   + Pavimento (Booleano) * Fotos (máximo 5 fotos) * Observaciones (String:500)   Si ocurre un error en la validación de uno o más campos, la interfaz deberá mostrar el error y explicar brevemente que ha sucedido.  La interfaz para modificar un inmueble es la misma que la interfaz de alta, se pueden modificar todos los campos. Se deben mostrar los datos del inmueble en los campos correspondientes.  La UI para la baja de un inmueble es un cartel avisando que el inmueble va a ser eliminado, y presentará las opciones para aceptar o no la eliminación. | | | | | | | | | | | | |
|  | |  |  | | |  |  | |  | |  | |
| Notas | | | | | | | | | | | | |
|
|
| Fecha | | Realizado | | | | A realizar | | | Comentarios | | | |
| 03/11/2016 | | Empezados test de los controladores | | | | Terminar test de los controladores  Programar controladores | | | No podemos continuar porque no están definidos los resultados de la lógica de ABM inmueble | | | |
|  | |  | | | |  | | |  | | | |
|  | |  | | | |  | | |  | | | |
|  | |  | | | |  | | |  | | | |
|  | |  | | | |  | | |  | | | |
|  | |  | | | |  | | |  | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Lógica alta, modificación y baja inmueble | |  | Programador | 2 | |
|  |  |  |  |  |  |  |
| Fecha |  |  |  | Task Points | 4 |  |
|  |  |  |  |  |  |  |
| Número de Historia | | 3 |  | Número de TaskCard | | 14 |
|  |  |  |  |  |  |  |
| Descripción:  Se recibe de la vista un objeto inmueble.  Para el alta se debe validar que:   * código del inmueble no se repita con otro inmueble * fecha de carga sea una fecha bien formada. * propietario sea un propietario previamente cargado (obligatorio). * Provincia: sea un string de como máximo 30 caracteres (obligatorio). * Localidad: sea un string de como máximo 30 caracteres (obligatorio). * Calle sea un string de como máximo 30 caracteres alfanuméricos. * Número sea numérico. * Piso sea numérico. * Departamento sea un string de como máximo 10 caracteres. * Barrio sea un string de como máximo 30 caracteres. * Tipo de inmueble: sea del tipo: L/local-oficina, C/casa, D/departamento, T/terreno, Q/quinta, G/galpón (obligatorio); * Precio de venta sea numérico (obligatorio); * Orientación sea un string (norte, sur, este, oeste, noreste, noroeste, sureste, suroeste); * Medidas: Frente (numérico), fondo (numérico), superficie (numérico); * Datos del edificio: propiedad horizontal (booleano), superficie (numérico), antigüedad (numérico), dormitorios (numérico), baños (numérico), garaje/cochera (booleano), patio (booleano), piscina (booleano), agua corriente (booleano), cloacas (booleano), gas natural (booleano), agua caliente (booleano), teléfono (numérico), lavadero (booleano), pavimento (booleano); * Foto: máximo 5 fotos; * Observaciones, string de 500 caracteres, como máximo; * Cada inmueble debe tener un estado. Luego del alta el estado pasa a ser “Alta”.   Si todo es correcto se da de alta al inmueble en la base de datos y se le asigna el estado alta, si no se devuelve una excepción.  Para la modificación se pueden modificar todos los campos y se deberá validar de la misma manera, si todo es correcto se guarda un historial de cambio y se modifica el inmueble en la base de datos. De lo contrario se devuelve una excepción.  Para la baja, no se deben tener consideraciones especiales y se debe realizar la baja del propietario en la base de datos | | | | | | |
|  |  |  |  |  |  |  |
| Notas | | | | | | |
|
| Fecha | Realizado | | A realizar | | Comentarios | |
| 03/11/2016 | Pruebas ABMC lógica inmueble. | | Implementación ABMC inmueble. | |  | |
| 04/11/2016 | Implementación ABMC inmueble. | | Nada | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Persistidor inmueble | |  | Programador | 2 | |
|  |  |  |  |  |  |  |
| Fecha |  |  |  | Task Points | 3 |  |
|  |  |  |  |  |  |  |
| Número de Historia | | 3 |  | Número de TaskCard | | 15 |
|  |  |  |  |  |  |  |
| Descripción  Se debe hacer una interfaz con métodos que permitan:  - Guardar un inmueble (si resulta en error lanza excepción SaveUpdateException)  - Modificar un inmueble (si resulta en error lanza excepción SaveUpdateException)  - Obtener un inmueble (si resulta en error lanza excepción ConsultaException)  - Obtener todos los inmuebles (si resulta en error lanza excepción ConsultaException)  Todas las excepciones mencionadas extienden a PersistenceException.  Se debe hacer una clase que implemente dicha interfaz mediante hibernate y completar las entidades con las anotaciones necesarias. | | | | | | |
|
|
|  |  |  |  |  |  |  |
| Notas | | | | | | |
|
|
| Fecha | Realizado | | A realizar | | Comentarios | |
| 01/11/16 | Guardar, modificar, obtener un inmueble y obtener inmuebles | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |

## ABM Cliente

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Entidad cliente | |  | Programador | 3 | |
|  |  |  |  |  |  |  |
| Fecha | 07/10/16 |  |  | Task Points | 1 |  |
|  |  |  |  |  |  |  |
| Número de Historia | | 6 |  | Número de TaskCard | | 16 |
|  |  |  |  |  |  |  |
| Descripción  Implementar la clase Cliente. La clase debe tener los siguientes atributos: id(Integer), nombre(string:100), apellido(string:100), tipoDocumento(TipoDocumento), numeroDocumento(string:30), telefono(string:30), estado(Estado).  Implementar la clase InmuebleBuscado. La clase debe tener los siguientes atributos: cliente(Cliente), tipos(ArrayList<TipoInmueble>), precioMin(double), precioMax(double).  Para cada atributo deben hacerse los métodos set y get. | | | | | | |
|
|
|  |  |  |  |  |  |  |
| Notas  El atributo buscado es el inmueble que el cliente buscó. | | | | | | |
|
|
| Fecha | Realizado | | A realizar | | Comentarios | |
| 17/10/2016 | Terminada | | Nada | | No se hacen pruebas porque son solo setters y getters | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Vista alta, modificar y baja cliente | |  | Programador | 1 | |
|  |  |  |  |  |  |  |
| Fecha |  |  |  | Task Points | 3 |  |
|  |  |  |  |  |  |  |
| Número de Historia | | 6 |  | Número de TaskCard | | 17 |
|  |  |  |  |  |  |  |
| Descripción:  Los campos a ingresar por el usuario son los siguientes:   * Nombre (String:30) (obligatorio) * Apellido (String:30) (obligatorio) * Tipo de documento [DNI, LC, LE, Pasaporte, Cédula Extranjera] (obligatorio) * DNI (String:30) (obligatorio) * Teléfono (String:16) * Datos del inmueble buscado:   + Tipo de inmueble [L/local-oficina, C/casa, D/departamento, T/terreno, Q/quinta, G/galpón] (obligatorio)   + Localidad (String:30)   + Barrios (Deben ser de la misma localidad seleccionada o estar vacío si no se seleccionó ninguna) (Lista de String:30)   + Características:     - Propiedad horizontal (Booleano)     - Superficie en m2 (Decimal)     - Antigüedad en años (Entero)     - Dormitorios mínimos (Entero)     - Baños mínimos (Entero)     - Garaje/cochera (Booleano)     - Patio (Booleano)     - Piscina (Booleano)     - Agua corriente (Booleano)     - Cloacas (Booleano)     - Gas natural (Booleano)     - Agua caliente (Booleano)     - Teléfono (Booleano)     - Lavadero (Booleano)     - Pavimento (Booleano) * Monto disponible   Todos los datos son obligatorios. Si ocurre un error en la validación de uno o más campos, la interfaz deberá mostrar el error y explicar brevemente que ha sucedido.  La interfaz para modificar un cliente es la misma que la interfaz de alta, se pueden modificar todos los campos. Se deben mostrar los datos del cliente en los campos correspondientes.  La UI para la baja de un cliente es un cartel avisando que el cliente va a ser eliminado, y presentará las opciones para aceptar o no la eliminación.  Las UI deben ser similares a los spikes realizados en la fase de exploración de la planificación. | | | | | | |
|  |  |  |  |  |  |  |
| Notas | | | | | | |
|
|
| Fecha | Realizado | | A realizar | | Comentarios | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Lógica alta, modificación y baja cliente | |  | Programador | 1 | |
|  |  |  |  |  |  |  |
| Fecha |  |  |  | Task Points | 4 |  |
|  |  |  |  |  |  |  |
| Número de Historia | | 6 |  | Número de TaskCard | | 18 |
|  |  |  |  |  |  |  |
| Descripción:  Se recibe de la vista un objeto cliente.  Se deberá validar que:   * Nombre sea un string de letras con un máximo 30 caracteres. * Apellido sea un string de letras con un máximo 30 caracteres. * Tipo de documento sea uno de los tipos definidos. * Número de documento sea String y corresponda con el tipo de documento, debe ser único. * Teléfono sea String. * Datos del inmueble buscado: tipo de inmueble, localidad, barrios, características, monto disponible.   Son obligatorios nombre, apellido y DNI.  Si todo es correcto se da de alta el cliente en la base de dato y si no se devuelve una excepción.  Se puede modificar todos los datos y se deben validar de la misma manera. Si todo es correcto se modifica al cliente en la base de datos y si no, se devuelve una excepción  Para la baja, no se deben tener consideraciones especiales y se debe realizar la baja del propietario en la base de datos | | | | | | |
|  |  |  |  |  |  |  |
| Notas | | | | | | |
|
| Fecha | Realizado | | A realizar | | Comentarios | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Persistidor cliente | |  | Programador | 3 | |
|  |  |  |  |  |  |  |
| Fecha |  |  |  | Task Points | 3 |  |
|  |  |  |  |  |  |  |
| Número de Historia | | 6 |  | Número de TaskCard | | 19 |
|  |  |  |  |  |  |  |
| Descripción  Se debe hacer una interfaz con métodos que permitan:  - Guardar un cliente (si resulta en error lanza excepción SaveUpdateException)  - Modificar un cliente (si resulta en error lanza excepción SaveUpdateException)  Todas las excepciones mencionadas extienden a PersistenceException.  Se debe hacer una clase que implemente dicha interfaz mediante. | | | | | | |
|
|
|  |  |  |  |  |  |  |
| Notas | | | | | | |
|
|
| Fecha | Realizado | | A realizar | | Comentarios | |
| 02/11/2016 | Terminadas pruebas persistidor | | Programar persistidor | |  | |
|  | Terminado programar persistidor | | Nada | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |

# Iteración 2:

## Consulta inmueble

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Vista de consulta inmueble | |  | Programador | 1 | |
|  |  |  |  |  |  |  |
| Fecha |  |  |  | Task Points |  | 2 |
|  |  |  |  |  |  |  |
| Número de Historia | | 4 |  | Número de TaskCard | | 20 |
|  |  |  |  |  |  |  |
| Descripción:  Se deberá presentar una vista donde permita consultar un inmueble ingresando país((ComboBox <País>), provincia ((ComboBox<Provincia>), localidad ((ComboBox<Localidad>), barrio ((ComboBox<Barrio >), tipo ((ComboBox<TipoInmueble>), cantidad de dormitorios (Entero > 0) y precio máximo (Decimal), precio mínimo (Decimal), estado del inmuble (ComboBox<EstadoInmueble>).  Si el rango de precios está al revés, debe invertirse y luego proseguir la consulta.  Si ocurre un error en la validación de uno o más campos, la interfaz deberá mostrar el error y explicar brevemente que ha sucedido.  Si todo es correcto se debe pasar a la lógica un objeto filtro con los campos completados por el usuario. Se debe mostrar cualquier error recibido. Si todo es correcto se deben presentar los inmuebles obtenidos como resultado de la consulta en forma de lista mostrando ubicación (país provincia, localidad, barrio), dimensión (frente, fondo, superficie) y precio.  Luego de la consulta se permitirá seleccionar algún resultado y luego elegir alguna de las siguientes funciones:  Baja Inmueble  Modificación Inmueble  Catálogo de inmueble (Genera un catálogo con los inmuebles seleccionados, uno o más)  Generar Reserva  Venta  Si se selecciona alguna de esas funciones, se debe invocar a la acción correspondiente.  Si se selecciona más de un inmueble, solo se permite la opción “Catálogo de inmueble”. | | | | | | |
|  |  |  |  |  |  |  |
| Notas: Los campos con ComboBox permitirán ingresar texto. | | | | | | |
|
|
| Fecha | Realizado | | A realizar | | Comentarios | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Lógica y persistidor de consulta inmueble | |  | Programador | 3 | |
|  |  |  |  |  |  |  |
| Fecha |  |  |  | Task Points |  | 3 |
|  |  |  |  |  |  |  |
| Número de Historia | | 4 |  | Número de TaskCard | | 21 |
|  |  |  |  |  |  |  |
| Descripción:  Se recibe de la vista un objeto filtro con los datos necesarios para realizar una consulta a la base de datos (País, Provincia, localidad, barrio, TipoInmueble, cantidad de dormitorios y rango de precios).  Se realiza la consulta en la base de datos con el filtro recibido. Los campos buscados deben contener el string ingresado en el filtro. Si ocurre algún error durante la consulta se debe devolver una excepción ConsultaException que extienda de PersistenceException.  Si no ha ocurrido ningún error se debe devolver un arreglo con los inmuebles que arrojó como resultado la consulta.  La realización de la consulta en la base de datos se debe implementar mediante hibernate y se deben completar las entidades con las anotaciones necesarias. | | | | | | |
|  |  |  |  |  |  |  |
| Notas | | | | | | |
|
|
| Fecha | Realizado | | A realizar | | Comentarios | |
| 3/12/2016 | Lógica y persistidor consulta inmueble | | Filtro consulta inmuble y test querys filtro consulta inmueble. | |  | |
| 4/12/2016 | Test querys filtro consulta inmueble | | Filtro consulta inmueble | | En los test se prueba sólo que las querys sean válidas para hibernate. | |
| 5/12/2016 | Filtro consulta inmueble | | Nada | |  | |
|  |  | |  | |  | |

## Catálogo de inmuebles

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Vista Alta Catálogo Inmueble | |  | Programador | 3 | |
|  |  |  |  |  |  |  |
| Fecha |  |  |  | Task Points |  | 3 |
|  |  |  |  |  |  |  |
| Número de Historia | | 5 |  | Número de TaskCard | | 22 |
|  |  |  |  |  |  |  |
| Descripción:  La carga de un catálogo es invocada desde la administración de clientes; al seleccionar uno, se habilita la opción de generar un catálogo para el mismo. También puede ser invocado al seleccionar varios inmuebles y seleccionar la opción de generar un catálogo para los mismos.  Se deberá presentar una vista donde permita crear un nuevo catálogo seleccionando los inmuebles que se incluirán en el mismo. Para realizar esta operación, se presenta un listado (inicialmente vacío, si se llamó desde cliente) con los inmuebles que serán parte del catálogo, junto con las opciones de “agregar nuevo inmueble”, “eliminar del catálogo” y “cambiar foto”.  Cuando se presiona “Agregar nuevo inmueble”, se presenta la vista correspondiente a consultar inmuebles, sólo con la opción de “agregar”. Se permitirá agregar más de un inmueble por consulta.  En la vista de administración del catálogo, se visualizarán los inmuebles a incluir en el catálogo junto con la foto que va a ser incluida en el mismo (por defecto la foto es seleccionada aleatoriamente entre las asociadas al inmueble). La foto puede ser cambiada. Cuando se selecciona esta opción, se presenta una vista que muestre las fotos asociadas al inmueble, permitiendo seleccionar solo una foto. Si no hay ninguna foto en el inmueble, el campo de foto no se mostrará.  Cuando la lista de inmuebles no está vacía, se habilita la opción de “Generar catálogo”. Esta operación debe invocar a la generación del catálogo de la lógica. Si el catálogo es generado exitosamente, se visualiza automáticamente con el visor de archivos PDF por defecto.  Si ocurre un error en la generación del catálogo se debe visualizar con un cartel describiendo lo que sucedió. | | | | | | |
| Notas | | | | | | |
|
|
| Fecha | Realizado | | A realizar | | Comentarios | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Lógica alta catálogo de inmueble y generar PDF | |  | Programador | 2 | |
|  |  |  |  |  |  |  |
| Fecha |  |  |  | Task Points |  | 3 |
|  |  |  |  |  |  |  |
| Número de Historia | | 5 |  | Número de TaskCard | | 23 |
|  |  |  |  |  |  |  |
| Descripción:  Se recibe de la vista un objeto CatalogoVista, que consta de un cliente, una lista de inmuebles y un Map de inmueble con foto.  Para el alta se deberá validar que los inmuebles no se repitan y que haya un cliente asociado al catálogo. También que los siguientes datos de los inmuebles deben existir: código inmueble, tipo de inmueble, localidad, dirección, barrio, precio.  Si todo es correcto se deberá generar un archivo PDF con los datos del catálogo: cliente, foto del inmueble, datos del inmueble (código inmueble, tipo de inmueble, país, provincia, localidad, dirección, barrio, precio, cantidad de dormitorios, baños, garaje, patio, superficie terreno, superficie edificada). Si los siguientes datos no existen, no deben mostrarse: cantidad de dormitorios, baños, garaje, patio, superficie terreno, superficie edificada.  Si algo no es correcto se devuelve un error que explique la situación. | | | | | | |
|  |  |  |  |  |  |  |
| Notas  La generación del PDF se debe hacer en una clase aparte encargada de esto. | | | | | | |
|
|
| Fecha | Realizado | | A realizar | | Comentarios | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |

## Generar reserva

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Vista alta, baja y listar reserva | |  | Programador | 2 | |
|  |  |  |  |  |  |  |
| Fecha |  |  |  | Task Points |  | 3 |
|  |  |  |  |  |  |  |
| Número de Historia | | 7 |  | Número de TaskCard | | 24 |
|  |  |  |  |  |  |  |
| Descripción:  En la vista de alta de reserva, los campos a ingresar por el usuario son los siguientes:   * Cliente (ComboBox<Cliente>) * Inmueble (Inmueble) * Importe (Decimal, 2 dígitos) * Tiempo de vigencia:   + Fecha inicio de la reserva (Date)   + Fecha fin de la reserva (Date)   Si el rango de fechas está al revés, debe invertirse y luego proseguir al alta.  Para seleccionar un inmueble, debe usarse la vista de listar inmueble, ya que permite filtrar.  Todos los campos son obligatorios.  Si ocurre un error en la validación de uno o más campos, la interfaz deberá mostrar el error y explicar brevemente que ha sucedido.  La vista de listar reserva puede accederse al seleccionar un inmueble o un cliente. Esta vista lista todas las reservas del cliente o inmueble seleccionado y de esta se puede proceder a ver el PDF de una reserva seleccionada, dar de baja la reserva seleccionada o a crear una nueva reserva para el inmueble o el cliente seleccionado. Al hacer esto último, se carga el campo correspondiente y se lo hace no editable. | | | | | | |
|  |  |  |  |  |  |  |
| Notas | | | | | | |
|
|
| Fecha | Realizado | | A realizar | | Comentarios | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Lógica alta, baja reserva y generar PDF | |  | Programador | 3 | |
|  |  |  |  |  |  |  |
| Fecha |  |  |  | Task Points |  | 3 |
|  |  |  |  |  |  |  |
| Número de Historia | | 7 |  | Número de TaskCard | | 25 |
|  |  |  |  |  |  |  |
| Descripción:  Se recibe de la vista un objeto ReservaVista, que consta de un cliente, un inmueble, un importe y un rango de vigencia.  Para el alta se deberá validar que haya un cliente y un inmueble asociados al catálogo. También que el importe sea un número decimal válido mayor que 0 y que el rango de vigencia tenga una fecha inicial mayor a hoy, y que tenga una fecha de inicio anterior a la de finalización.  Si todo es correcto se deberá generar un archivo PDF con los datos de la reserva. Luego se genera una Reserva con dicho PDF, el cliente, el inmueble y el rango de fechas pasado y se dará de alta en la base de datos. También se deberá enviar un correo al mail del cliente con el archivo PDF como adjunto en otro hilo.  Si algo no es correcto se devuelve un error que explique la situación.  Para la baja se debe validar que los campos sean los mismos que en la base de datos, se debe cambiar el estado a BAJA y, por último, se debe persistir. Si algo no es correcto se devuelve un error que explique la situación. | | | | | | |
|  |  |  |  |  |  |  |
| Notas  La generación del PDF se debe hacer en una clase aparte encargada de esto. | | | | | | |
|
|
| Fecha | Realizado | | A realizar | | Comentarios | |
| 6/12/2016 | Tests | | Lógica alta y baja reserva y generar PDF de reserva. | | No se harán test de generar PDF reserva porque para esto se utilizará un Framework externo. | |
| 7/12/2016 | Lógica alta y baja reserva. | | Generar PDF de reserva. | |  | |
| 8/12/2016 | Generar PDF de reserva. | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Lógica envío de mail | |  | Programador | 2 | |
|  |  |  |  |  |  |  |
| Fecha |  |  |  | Task Points |  | 1 |
|  |  |  |  |  |  |  |
| Número de Historia | | 7 |  | Número de TaskCard | | 26 |
|  |  |  |  |  |  |  |
| Descripción:  Se recibe un objeto mail con un String de correo, un String de asunto y una lista de objetos Archivo para adjuntar.  La clase debe armar con estos datos un correo y lo debe enviar a través del servidor de correo y la cuenta del cliente. ([olimpoagilinmobiliaria2016@gmail.com](mailto:olimpoagilinmobiliaria2016@gmail.com)) | | | | | | |
|  |  |  |  |  |  |  |
| Notas | | | | | | |
|
|
| Fecha | Realizado | | A realizar | | Comentarios | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Cambios ABM Cliente | |  | Programador | 3 | |
|  |  |  |  |  |  |  |
| Fecha |  |  |  | Task Points |  | 1 |
|  |  |  |  |  |  |  |
| Número de Historia | | 7 |  | Número de TaskCard | | 27 |
|  |  |  |  |  |  |  |
| Descripción:  Se debe agregar un campo de correo al cliente. Para esto hay que modificar tanto la vista de crear como la de modificar cliente. También hay que validar este correo en la lógica y modificar la entidad Cliente. | | | | | | |
|  |  |  |  |  |  |  |
| Notas  Recordar actualizar las pruebas. | | | | | | |
|
|
| Fecha | Realizado | | A realizar | | Comentarios | |
| 2/12/2016 | Todo, incluso las pruebas | | Nada | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Persistidor y entidad reserva | |  | Programador | 2 | |
|  |  |  |  |  |  |  |
| Fecha |  |  |  | Task Points |  | 1 |
|  |  |  |  |  |  |  |
| Número de Historia | | 7 |  | Número de TaskCard | | 28 |
|  |  |  |  |  |  |  |
| Descripción  Se debe hacer una interfaz con métodos que permitan:  - Guardar una reserva (si resulta en error lanza excepción SaveUpdateException)  - Modificar una reserva (si resulta en error lanza excepción SaveUpdateException)  - Obtener todas las reservas (si resulta en error lanza excepción ConsultaException)  Todas las excepciones mencionadas extienden a PersistenceException.  Se debe hacer una clase que implemente dicha interfaz mediante hibernate y completar las entidades con las anotaciones necesarias.  Implementar la clase Reserva. La clase debe tener los siguientes atributos: id (Integer), cliente (Cliente), inmueble (Inmueble), archivoPDF (PDF), fechaInicio (Date), fechaFin (Date), estado (Estado).  Para cada atributo deben hacerse los métodos set y get. | | | | | | |
|
|
|  |  |  |  |  |  |  |
| Notas | | | | | | |
|
|
| Fecha | Realizado | | A realizar | | Comentarios | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |

## Ventas

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Vista alta y listar venta | |  | Programador | | 1 | |
|  |  |  |  |  | |  |  |
| Fecha |  |  |  | Task Points | |  | 3 |
|  |  |  |  |  | |  |  |
| Número de Historia | |  |  | Número de TaskCard | | | 29 |
|  |  |  |  |  |  | |  |
| Descripción:  Un vendedor seleccionará, desde el listado de inmuebles, un inmueble que desea vender.  Se mostrará una pantalla con el propietario y datos relevantes del inmueble.  En la pantalla se pedirá que se ingrese el monto de venta y el nuevo propietario, que debe ser un cliente.  Con un botón de cancelar venta y otro de realizar venta, en este último se pedirá la contraseña al vendedor para confirmar la venta. Al confirmar la venta, esta se pasa a la lógica para que genere un PDF y guarde la venta.  Si hay algún error debe mostrarse. Sino, debe abrirse un cuadro diálogo de impresión del PDF generado por la lógica. Luego, si se acepta la impresión, se lo debe mandar a la lógica para que lo imprima.  Para la vista de listar ventas, se debe tener en cuenta de donde es llamado, ya que puede llamarse desde la selección de un vendedor (al listar vendedores), desde la selección de un cliente (al listar clientes), desde la selección de un propietario (al listar propietarios). Se debe mostrar el nombre y apellido de a quién pertenecen las ventas junto con su rol.  Al listar las ventas, se debe mostrar la fecha de la venta, cliente (nombre y apellido), propietario (nombre y apellido), inmueble (dirección), vendedor (nombre y apellido) – quitando la columna del rol que lo llamó. Se debe poder seleccionar una y ver el PDF generado al concretar la venta. | | | | | | | |
|  |  |  |  |  |  | |  |
| Notas | | | | | | | |
|
|
| Fecha | Realizado | | A realizar | | Comentarios | | |
|  |  | |  | |  | | |
|  |  | |  | |  | | |
|  |  | |  | |  | | |
|  |  | |  | |  | | |
|  |  | |  | |  | | |
|  |  | |  | |  | | |
|  |  | |  | |  | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Lógica alta venta y generar PDF | |  | Programador | | 1 | |
|  |  |  |  |  | |  |  |
| Fecha |  |  |  | Task Points | |  | 3 |
|  |  |  |  |  | |  |  |
| Número de Historia | |  |  | Número de TaskCard | | | 30 |
|  |  |  |  |  |  | |  |
| Descripción:  Se recibe un objeto venta con el monto de venta, un vendedor, un cliente, un propietario y un inmueble.  Se le asignará una fecha de venta, se validará el monto ingresado y que el cliente que compra el inmueble sea distinto del propietario. También hay que validar que si el inmueble tiene una reserva en el día de la fecha de venta, la venta no se realizará, a menos que el cliente de la reserva sea el mismo del de la compra.  Si todo es correcto se deberá generar un archivo PDF con los siguientes datos de la venta: fecha, monto, propietario anterior, propietario actual, y los datos relevantes del inmueble.  El inmueble pasa al estado VENDIDO. | | | | | | | |
|  |  |  |  |  |  | |  |
| Notas  La generación del PDF se debe hacer en una clase aparte encargada de esto. | | | | | | | |
|
|
| Fecha | Realizado | | A realizar | | Comentarios | | |
|  |  | |  | |  | | |
|  |  | |  | |  | | |
|  |  | |  | |  | | |
|  |  | |  | |  | | |
|  |  | |  | |  | | |
|  |  | |  | |  | | |
|  |  | |  | |  | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Lógica imprimir venta | |  | Programador | 1 | |
|  |  |  |  |  |  |  |
| Fecha |  |  |  | Task Points |  | 1 |
|  |  |  |  |  |  |  |
| Número de Historia | | 8 |  | Número de TaskCard | | 31 |
|  |  |  |  |  |  |  |
| Descripción:  Se recibe un objeto PDF y se realiza una impresión del mismo. | | | | | | |
|  |  |  |  |  |  |  |
| Notas | | | | | | |
|
|
| Fecha | Realizado | | A realizar | | Comentarios | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Persistidor y entidad venta | |  | Programador | 2 | |
|  |  |  |  |  |  |  |
| Fecha |  |  |  | Task Points |  | 2 |
|  |  |  |  |  |  |  |
| Número de Historia | | 8 |  | Número de TaskCard | | 32 |
|  |  |  |  |  |  |  |
| Descripción  Se debe hacer una interfaz con métodos que permitan:  - Guardar una venta (si resulta en error lanza excepción SaveUpdateException)  - Modificar una venta (si resulta en error lanza excepción SaveUpdateException)  - Obtener todas las ventas (si resulta en error lanza excepción ConsultaException)  - Obtener todas las ventas de un vendedor (si resulta en error lanza excepción ConsultaException)  - Obtener todas las ventas de un propietario (si resulta en error lanza excepción ConsultaException)  - Obtener todas las ventas de un cliente (si resulta en error lanza excepción ConsultaException)  - Obtener la venta de un inmueble (si resulta en error lanza excepción ConsultaException)  Todas las excepciones mencionadas extienden a PersistenceException.  Se debe hacer una clase que implemente dicha interfaz mediante hibernate y completar las entidades con las anotaciones necesarias.  Implementar la clase Venta. La clase debe tener los siguientes atributos: id (Integer), cliente (Cliente), inmueble (Inmueble), propietario (Propietario), vendedor (Vendedor), archivoPDF (PDF), importe (Double), medioDePago (String), fecha (Date).  Para cada atributo deben hacerse los métodos set y get. | | | | | | |
|
|
|  |  |  |  |  |  |  |
| Notas | | | | | | |
|
|
| Fecha | Realizado | | A realizar | | Comentarios | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Agregar EstadoInmueble a inmueble | |  | Programador | 1 | |
|  |  |  |  |  |  |  |
| Fecha |  |  |  | Task Points |  | 1 |
|  |  |  |  |  |  |  |
| Número de Historia | | 7 |  | Número de TaskCard | | 33 |
|  |  |  |  |  |  |  |
| Descripción:  Se debe agregar un campo de EstadoInmueble al inmueble. Para esto hay que modificar la lógica de creación del inmueble y modificar la entidad Inmueble.  También hay que agregar una función para listar los EstadoInmueble en el gestor de datos y en persistidor de datos.  Hay que crear una entidad EstadoInmueble y un Enum EstadoInmuebleStr (VENDIDO y NO\_VENDIDO). También hay que agregar estos Strings al sql inicial. | | | | | | |
|  |  |  |  |  |  |  |
| Notas  Recordar actualizar las pruebas. | | | | | | |
|
|
| Fecha | Realizado | | A realizar | | Comentarios | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |