Contenido

[Iteración 1: 2](#_Toc463881336)

[ABM Vendedor 2](#_Toc463881337)

[Vista y lógica del Login 2](#_Toc463881338)

[Entidad vendedor 2](#_Toc463881339)

[Vista alta, modificar y baja Vendedor 3](#_Toc463881340)

[Vista principal para la administración 3](#_Toc463881341)

[Lógica alta, modificación y baja vendedor 4](#_Toc463881342)

[Persistidor vendedor 4](#_Toc463881343)

[ABM Propietario 5](#_Toc463881344)

[Entidad propietario 5](#_Toc463881345)

[Vista alta, modificar y baja Propietario 5](#_Toc463881346)

[Lógica alta, modificación y baja propietario 6](#_Toc463881347)

[Persistidor propietario 7](#_Toc463881348)

[ABM Inmueble 7](#_Toc463881349)

[Entidad inmueble 7](#_Toc463881350)

[Clases de datos 8](#_Toc463881351)

[Vista alta, modificar y baja inmueble 9](#_Toc463881352)

[Persistidor inmueble 11](#_Toc463881353)

[ABM Cliente 11](#_Toc463881354)

[Entidad cliente 11](#_Toc463881355)

[Vista alta, modificar y baja cliente 12](#_Toc463881356)

[Lógica alta, modificación y baja cliente 13](#_Toc463881357)

[Persistidor cliente 13](#_Toc463881358)

# 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) * Contraseña (String)   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. | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **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), id(Integer), password(string), salt(String), 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:  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, 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 | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **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), apellido(string), tipoDocumento(TipoDocumento), numeroDocumento(string), id(Integer), dirección(Direccion), teléfono(string), email(string), inmuebles(ArrayList<Inmueble>).  Para cada atributo deben hacerse los métodos set y get. | | | | | | |
|
|
|  |  |  |  |  |  |  |
| Notas | | | | | | |
|
|
| Fecha | Realizado | | A realizar | | Comentarios | |
| 20/10/16 2hs | 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 la baja del propietario en la base de datos | | | | | | |
|  |  |  |  |  |  |  |
| Notas | | | | | | |
|
|
| Fecha | Realizado | | A realizar | | Comentarios | |
| 21/10/16  2hs | Logica de Alta y modificacion | | Baja de un propietario | | Se realizaron los casos de prueba antes de codificar | |
| 22/10/16  1hs | 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)  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 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), 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), direccion(Direccion), estado(Estado).  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(string), numero(string), barrio(string), piso(string), departamento(string) y localidad(Localidad).  La clase Localidad que tiene los atributos nombre(string), id(Integer) y provincia(Provincia).  La clase Provincia con los atributos nombre(string), id(Integer) y pais(Pais).  La clase Pais, con atributos nombre(string) y id(Integer).  La clase TipoDocumento tiene los atributos id(Integer) y nombre(string).  La clase TipoInmueble tiene los atributos id(Integer) y nombre(string).  La clase Orientacion tiene los atributos id(Integer) y orientacion(string).  La clase MedidasInmueble tiene los atributos frente(double), fondo(double), superficie(double).  La clase DatosEdificio tiene los 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).  La clase InmuebleBuscado tiene los atributos cliente(Cliente), tipos(ArrayList<TipoInmueble>), precioMin(double), precioMax(double).  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. | | | | | | |
|
|
| Fecha | Realizado | | A realizar | | Comentarios | |
| 20/10/16 | Atributos, getters y getters, equals | | Anotaciones | |  | |
| 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) * Provincia: por defecto Santa Fe (String:30) (obligatorio) * Localidad (String:30) (obligatorio) * Barrio (String:30) (obligatorio) * Calle/número (String:30) (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) * 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 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 | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **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)  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 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), apellido(string), tipoDocumento(TipoDocumento), numeroDocumento(string), telefono(string), estado(Estado).  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 | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |