**UTN - Facultad Regional Santa Fe**

**MÉTODOS ÁGILES**

**PARA EL DESARROLLO DE SOFTWARE**

Trabajo Práctico Programación Extrema

**Alumnos:**

Berti, Fernando bertilxi@gmail.com

Campodonico, Daniel d.campodonico@hotmail.com

Gioria, Emiliano emigioria@hotmail.com

Moretti, Lucas moretti.lucas@hotmail.com

Rebechi, Esteban estebanrebechi\_5@hotmail.com

Rico, Andrés andres.rico94@gmail.com

**Docentes:**

* Ing. Bracalenti, Claudio.
* Ing. Ledesma, Rodrigo.

**Año 2016**

Índice

[Descripción del sistema 4](#_Toc469586367)

[Equipo de desarrollo 4](#_Toc469586368)

[Planificación del Release 4](#_Toc469586369)

[Estimación de Esfuerzo, Riesgo y Prioridad 12](#_Toc469586370)

[Spikes 13](#_Toc469586371)

[Definición de Velocidad y Alcance 13](#_Toc469586372)

[Metáfora 13](#_Toc469586373)

[Fecha de Inicio del desarrollo 13](#_Toc469586374)

[Planificación Iteración 1 14](#_Toc469586375)

[Tareas 14](#_Toc469586376)

[ABM Vendedor 14](#_Toc469586377)

[Vista y lógica del Login 14](#_Toc469586378)

[Entidad vendedor 14](#_Toc469586379)

[Vista alta, modificar y baja Vendedor 15](#_Toc469586380)

[Vista principal para la administración 16](#_Toc469586381)

[Lógica alta, modificación y baja vendedor 16](#_Toc469586382)

[Persistidor vendedor 17](#_Toc469586383)

[ABM Propietario 18](#_Toc469586384)

[Entidad propietario 18](#_Toc469586385)

[Vista alta, modificar y baja Propietario 18](#_Toc469586386)

[Lógica alta, modificación y baja propietario 19](#_Toc469586387)

[Persistidor propietario 20](#_Toc469586388)

[ABM Inmueble 21](#_Toc469586389)

[Entidad inmueble 21](#_Toc469586390)

[Clases de datos 22](#_Toc469586391)

[Vista alta, modificar y baja inmueble 23](#_Toc469586392)

[Lógica alta, modificación y baja inmueble 24](#_Toc469586393)

[Persistidor inmueble 25](#_Toc469586394)

[ABM Cliente 26](#_Toc469586395)

[Entidad cliente 26](#_Toc469586396)

[Vista alta, modificar y baja cliente 27](#_Toc469586397)

[Lógica alta, modificación y baja cliente 28](#_Toc469586398)

[Persistidor cliente 29](#_Toc469586399)

[Estimación de Esfuerzo 29](#_Toc469586400)

[Conclusiones parciales de lo estimado 29](#_Toc469586401)

[Planificación Iteración 2 30](#_Toc469586402)

[Tareas 30](#_Toc469586403)

[Consulta inmueble 30](#_Toc469586404)

[Vista de consulta inmueble 30](#_Toc469586405)

[Lógica y persistidor de consulta inmueble 31](#_Toc469586406)

[Catálogo de inmuebles 32](#_Toc469586407)

[Vista Alta Catálogo Inmueble 32](#_Toc469586408)

[Lógica alta catálogo de inmueble y generar PDF 32](#_Toc469586409)

[Generar reserva 33](#_Toc469586410)

[Vista alta, baja y listar reserva 33](#_Toc469586411)

[Lógica alta, baja reserva y generar PDF 34](#_Toc469586412)

[Lógica envío de mail 34](#_Toc469586413)

[Cambios ABM Cliente 35](#_Toc469586414)

[Persistidor y entidad reserva 35](#_Toc469586415)

[Ventas 36](#_Toc469586416)

[Vista alta y listar venta 36](#_Toc469586417)

[Lógica alta venta y generar PDF 37](#_Toc469586418)

[Lógica imprimir venta 38](#_Toc469586419)

[Persistidor y entidad venta 38](#_Toc469586420)

[Agregar EstadoInmueble a inmueble 39](#_Toc469586421)

[Estimación de Esfuerzo 40](#_Toc469586422)

[Conclusiones parciales de lo estimado 40](#_Toc469586423)

[Código Fuente 41](#_Toc469586424)

[Pruebas de Unidad 41](#_Toc469586425)

[Refactorización 41](#_Toc469586426)

[Estándares de programación 41](#_Toc469586427)

[Conclusiones 41](#_Toc469586428)

# Descripción del sistema

La inmobiliaria FRSF nos ha solicitado desarrollar un sistema de información que permita gestionar operaciones de compra-venta de inmuebles, desde que un propietario se presenta con un inmueble para la venta hasta que se concreta la misma.

# Equipo de desarrollo

* Fernando Berti: Programador.
* Daniel Campodonico: Programador.
* Emiliano Gioria: Programador.
* Lucas Moretti: Programador.
* Esteban Rebechi: Programador.
* Andrés Rico: Programador.

Para la realización de este trabajo práctico con la metodología ágil XP, el cliente del sistema es el Ing. Rodrigo Ledesma.

La programación de a pares se realizó para las dos iteraciones de la siguiente manera:

* Pareja 1: Berti/Rebechi.
* Pareja 2: Campodonico/Moretti.
* Pareja 3: Gioria/Rico.

# Planificación del Release

Historias de usuario

1. ***ABM Vendedor - Baja***

El vendedor es el usuario del sistema que realiza todas las operaciones sobre el mismo. Para su acceso se requiere los datos personales y una clave de acceso, la cual se debe solicitar antes de acceder a las funcionalidades del sistema.

1. ***ABM propietario - Alta***

Se debe permitir cargar, modificar y eliminar propietarios. Sus datos son nombre y apellido, tipo documento, número documento, calle, número, localidad, provincia, teléfono, email.

1. ***ABM inmueble - Alta***

El sistema debe permitir cargar y modificar inmuebles asociados a un propietario (se debe elegir el propietario previamente cargado). Cuando se carga un inmueble, el sistema debe otorgar código de inmueble, guardar la fecha de carga y asignar estado alta al inmueble. El código de inmueble es un valor secuencial independiente para cada inmueble.

Los datos a cargar para un inmueble son:

* provincia: por defecto Santa Fe;
* localidad: para la provincia de Santa Fe, se deben incluir las siguientes localidades en un listado: Santa Fe, Santo Tomé, Sauce Viejo, Rincón, Colastiné Norte, Colastiné Sur. En caso de que la localidad no sea ninguna de las anteriores, se debe poder ingresar una;
* calle/número: si la localidad elegida tiene asociada calles, se debe elegir la misma de un listado. Si no tiene asociada calles, se permite ingresar la calle/número o una ubicación;
* piso/departamento;
* barrio;
* tipo de inmueble: se debe poder elegir de un listado. Los tipos de inmuebles son: L/local-oficina, C/casa, D/departamento, T/terreno, Q/quinta, G/galpón;
* precio de venta (obligatorio) ;
* orientación (norte, sur, este, oeste, noreste, noroeste, sureste, suroeste), frente (metros), fondo (metros), superficie (m2). Datos del edificio: propiedad horizontal (si/no), superficie (m2), antigüedad, dormitorios, baños, garaje/cochera, patio, piscina, agua corriente, cloacas, gas natural, agua caliente, teléfono, lavadero, pavimento;
* Foto y observaciones;

1. ***Consulta Inmueble - Alta***

El sistema debe permitir consultar inmueble por provincia, localidad, barrio, tipo, cantidad de dormitorios y precio (rango).

1. ***Catálogo de Inmuebles - Media***

El sistema debe permitir generar un catálogo con inmuebles para un cliente (con posibilidad de elegir los inmuebles). Debe incluir para cada inmueble una foto, código inmueble, tipo de inmueble, localidad, dirección, barrio, cantidad de dormitorios, baños, garaje, patio, superficie de terreno, superficie edificada, precio. En cada página debe figurar la fecha de emisión.

1. ***ABM cliente - Media***

El sistema debe permitir cargar y modificar datos de clientes: nombre, apellido, teléfono y los datos del inmueble buscado: tipo de inmueble, localidad, barrios, características, monto disponible.

1. ***Generar Reserva - Media***

El cliente puede reservar un inmueble elegido generándose un documento reserva del inmueble a partir de los datos del cliente y del inmueble. Se debe ingresar el importe de la reserva y el tiempo de vigencia de la misma. Se debe marcar que el inmueble está reservado y por quién en caso de que otro cliente desee reservarlo. El documento debe quedar asociado al Cliente y debe enviarse a su casilla de correo.

1. ***Venta - Media***

Al generarse la venta se genera un documento similar al de reserva indicando los datos del inmueble vendido y el cliente. Este documento debe imprimirse para la firma por parte del cliente. Así también debe figurar el monto de la venta y se debe marcar como vendido a dicho inmueble. El documento de venta debe asociarse al cliente y al inmueble.

1. ***Publicar - Baja***

La inmobiliaria posee un sitio web donde publica los inmuebles de colocando los datos de los inmuebles más relevantes, las fotos y un video de tour del inmueble dependiendo de la calidad del mismo. En dicho sitio deben figurar los datos de contacto de la inmobiliaria.

Historias Refinadas

|  |  |
| --- | --- |
| **Historia de Usuario** | |
|
| Numero: 1 | Nombre: ABM Vendedor |
| Usuario: | |
| Modificación Historia: | Iteración Asignada: |
| Prioridad en Negocio: Baja | Puntos Estimados: 2 |
|
| Riesgo en Desarrollo: Baja |
|
| Descripción: | |
| Se deberán ingresar los siguientes datos personales para realizar el alta: (todos los datos son obligatorios)   * + Nombre (String:30),   + Apellido (String:30),   + Tipo de Documento (DNI, LC, LE, Pasaporte, Cédula Extranjera),   + Número de documento (String:30) y   + Contraseña (String:100).   La contraseña debe estar compuesta por caracteres alfanuméricos. Se podrán modificar todos los datos. El DNI no podrá repetirse.  Cada vendedor tendrá una clave de acceso al sistema, la cual, luego de ingresar al sistema, solo permitirá acceder a las siguientes funcionalidades:   * ABM Inmueble. * ABM Propietario. * Consulta inmueble. * Catálogo de inmueble. * ABM Cliente. * Generar Reserva. * Venta. * Publicar.   Además, se debe crear un vendedor con el rol de administrador el cual tendrá acceso a la funcionalidad de administrar otros vendedores. | |
|
|
|
| Observaciones:  La contraseña se deberá hashear con un salt distinto para cada usuario, el cual se deberá almacenar en la entidad correspondiente a Vendedor.  El vendedor debe tener un atributo Estado para manejar la baja lógica. | |
|
| **Historia de Usuario** | |
|
| Numero: 2 | Nombre: ABM propietario |
| Usuario: | |
| Modificación Historia: | Iteración Asignada: |
| Prioridad en Negocio: Alta | Puntos Estimados: 2 |
|
| Riesgo en Desarrollo: Baja |
|
| Descripción: | |
| Se deberán ingresar los siguientes datos personales para realizar el alta de un propietario:   * Nombre (String:30), * Apellido (String:30), * Tipo de Documento (DNI, LC, LE, Pasaporte, Cédula Extranjera), * Número de documento (String:30), * Dirección (País, Provincia, Localidad, Barrio, Calle, Número, Piso, Departamento, Otros), * Teléfono (String:30) y * Email (String:30).   Los siguientes datos son obligatorios: nombre, apellido, tipo, número de documento, calle, número, país, provincia, localidad, barrio, teléfono.  Todos los datos pueden ser modificados.  Todos los datos pueden repetirse, excepto el tipo y número de documento.  No se llevará historial de cambios del mismo.  El propietario no podrá ser dado de baja en caso de tener alguna propiedad asociada.  La baja se realiza de manera lógica. | |
|
|
|
| Observaciones:  Una dirección está compuesta por:   * Localidad (String:50), * Barrio (String:50), * Calle (String:50), * Número (String:30), * Piso (String:30), * Departamento (String:30), * Otros (String:100).   Una Localidad tiene una Provincia (String:50).  Una Provincia tiene un País (String:50). | |
|
| **Historia de Usuario** | |
|
| Numero: 3 | Nombre: ABM inmueble |
| Usuario: | |
| Modificación Historia: | Iteración Asignada: |
| Prioridad en Negocio: Alta | Puntos Estimados: 2 |
|
| Riesgo en Desarrollo: Media |
|
| Descripción: | |
| Se deberán ingresar los siguientes datos para el alta de un inmueble:   * Código del inmueble (código generado por el sistema), * Fecha de carga (Date, fecha actual, no se puede modificar), * Propietario (previamente cargado, obligatorio), * País (String:50), * Provincia (String:50): por defecto Santa Fe (obligatorio), * Localidad (String:50): para la provincia de Santa Fe, se deben incluir las siguientes localidades en un listado: Santa Fe, Santo Tomé, Sauce Viejo, Rincón, Colastiné Norte, Colastiné Sur. En caso de que la localidad no sea ninguna de las anteriores, se debe poder ingresar una (obligatorio), * Calle (String:50) * Número (String:30, numérico): si la localidad elegida tiene asociada calles, se debe elegir la misma de un listado. Si no tiene asociada calles, se permite ingresar la calle y número o una ubicación mediante el campo Otros (obligatorio), * Piso (String:30), * Departamento (String:30), * Barrio (String:50, obligatorio), * Tipo de inmueble (listado: L/local-oficina, C/casa, D/departamento, T/terreno, Q/quinta, G/galpón) (obligatorio), * Precio de venta (Double, obligatorio), * Orientación (norte, sur, este, oeste, noreste, noroeste, sureste, suroeste); * Medidas (Double):   + Frente (en metros),   + Fondo (en metros),   + Superficie (en m2). * Datos del edificio:   + Propiedad horizontal (Boolean),   + Superficie (Double, en m2),   + Antigüedad (Integer, en años),   + Dormitorios (Integer),   + Baños (Integer),   + Garaje/cochera (Boolean),   + Patio (Boolean),   + Piscina (Boolean),   + Agua corriente (Boolean),   + Cloacas (Boolean),   + Gas natural (Boolean),   + Agua caliente (Boolean),   + Teléfono (Boolean),   + Lavadero (Boolean),   + Pavimento (Boolean). * Fotos (sin restricción de tamaño), * Observaciones (String:300).   Cada inmueble debe tener un estado (alta, baja). Luego del alta el estado pasa a ser “Alta”.  Todos los datos se pueden repetir, excepto el código.  Se pueden modificar todos los cambios.  La baja se realiza de forma lógica seteando el estado en “Baja”.  Se debe llevar un historial de todos los cambios. | |
|
|
|
| Observaciones: | |
|
| **Historia de Usuario** | |
|
| Numero: 4 | Nombre: Consulta Inmueble |
| Usuario: | |
| Modificación Historia: | Iteración Asignada: |
| Prioridad en Negocio: Alta | Puntos Estimados: 1 |
|
| Riesgo en Desarrollo: Baja |
|
| Descripción: | |
| El sistema debe permitir consultar por inmuebles que cumplan determinados criterios de búsqueda que los ingresa el usuario:   * País (String:50), * Provincia (String:50), * Localidad (String:50), * Barrio (String:50), * Tipo de inmueble (TipoInmueble), * Cantidad de dormitorios (Integer), * Precio (seleccionando un rango) y * Estado (vendido, disponible).   Se listarán los inmuebles de la base de datos que cumplan con los criterios de búsqueda y se los lista en una tabla mostrando el tipo, la ubicación y el propietario del mismo.  Se permitirá seleccionar un inmueble y luego elegir alguna de las siguientes funciones:   * Baja Inmueble. * Modificación Inmueble. * Catálogo de inmueble (Genera un catálogo con los inmuebles seleccionados). * Generar Reserva. * Venta.   Si se selecciona más de un inmueble se deben deshabilitar todos los botones que permitan las acciones anteriores excepto el de generar catálogo. | |
|
|
|
| Observaciones:  El criterio de búsqueda para los campos que son del tipo String debe ser “contiene”. | |
|
| **Historia de Usuario** | |
|
| Numero: 5 | Nombre: Catálogo de Inmuebles |
| Usuario: | |
| Modificación Historia: | Iteración Asignada: |
| Prioridad en Negocio: Media | Puntos Estimados: 2 |
|
| Riesgo en Desarrollo: Alta |
|
| Descripción: | |
| Se debe permitir generar un catálogo de inmuebles para un cliente. Para cada inmueble dentro del catálogo se debe incluir una foto, código inmueble, tipo de inmueble, localidad, dirección, barrio, cantidad de dormitorios, baños, garaje, patio, superficie terreno, superficie edificada, precio.  En cada página debe figurar la fecha de emisión del catálogo.  La foto que irá en el catálogo se podrá elegir entre las fotos que ya están cargadas en el sistema para cada inmueble.  El diseño del catálogo deberá contemplar un encabezado donde figure la fecha de emisión del mismo, el logo del sistema y la identificación de la inmobiliaria. Un pie de página en donde figure la página actual y la cantidad de páginas en total.  Las imágenes de los inmuebles en el catálogo deben tener un tamaño considerable; 3 inmuebles por página.  Luego de generar el catálogo se debe poder mostrar al usuario y permitirle imprimirlo y guardarlo. | |
|
|
|
| Observaciones:  En el caso de los departamentos la superficie del terreno y la cubierta es la misma.  El catálogo es generado como un archivo PDF. | |
|
| **Historia de Usuario** | |
|
| Numero: 6 | Nombre: ABM cliente |
| Usuario: | |
| Modificación Historia: | Iteración Asignada: |
| Prioridad en Negocio: Media | Puntos Estimados: 2 |
|
| Riesgo en Desarrollo: Baja |
|
| Descripción: | |
| Para realizar el alta de un nuevo cliente se deberá permitir ingresar los siguientes datos:   * Nombre (String:30), * Apellido (String:30), * Número de documento (String:30), * Teléfono (String:30) y * Los datos del inmueble buscado:   + Tipo de inmueble [L/local-oficina, C/casa, D/departamento, T/terreno, Q/quinta, G/galpón] (obligatorio),   + Localidad (String:50),   + Barrios (Lista de String:50),   + 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).   Son obligatorios nombre, apellido, tipo y número de documento.  Se puede modificar y repetir todos los datos menos el tipo y número de documento.  No se llevará el historial de cambios del cliente. | |
|
|
|
| Observaciones: | |
|
| **Historia de Usuario** | |
|
| Numero: 7 | Nombre: Generar Reserva |
| Usuario: | |
| Modificación Historia: | Iteración Asignada: |
| Prioridad en Negocio: Media | Puntos Estimados: 1 |
|
| Riesgo en Desarrollo: Baja |
|
| Descripción: | |
| El cliente debe poder generar una reserva para un inmueble elegido; la reserva debe tener los siguientes datos:   * Cliente (Cliente), * Inmueble (Inmueble), * Fecha Actual (Date), * Fecha de vigencia (Date), * Importe (Double).   Para generar la reserva el usuario (vendedor) debe ingresar el importe de la reserva y el tiempo de vigencia de la misma.  El inmueble se debe marcar como reservado.  Se debe generar un documento asociado a la reserva con los datos del cliente, el inmueble, el propietario y de la reserva (fecha de vigencia e importe).  El documento se debe asociar al cliente y ser enviado a su dirección de correo.  A su vez, se guardará en la base de datos y se le preguntará al usuario si desea imprimirlo.  También se debe poder listar las reservas seleccionando un cliente o un inmueble.  Se debe permitir dar de baja una reserva. | |
|
|
|
| Observaciones:  La baja de una reserva se debe implementar de manera lógica. | |
|
| **Historia de Usuario** | |
|
| Numero: 8 | Nombre: Venta |
| Usuario: | |
| Modificación Historia: | Iteración Asignada: |
| Prioridad en Negocio: Media | Puntos Estimados: 2 |
|
| Riesgo en Desarrollo: Baja |
|
| Descripción: | |
| El usuario debe poder generar una venta seleccionando un inmueble e ingresando los siguientes datos:   * Cliente (ComboBox<Cliente>) (comprador), * Importe (Double), * Medio de pago (String).   Una vez confirmada la venta, se genera un documento de venta donde figuran los datos del cliente (comprador), del inmueble, del propietario (vendedor del inmueble) y datos de la venta (importe, medio de pago, fecha).  El documento se debe poder imprimir y también almacenarse digitalmente.  El inmueble debe poder ser marcado como vendido.  El documento generado se debe asociar al cliente y al inmueble.  Una venta no puede ser modificada ni eliminada. | |
|
|
|
| Observaciones: | |
|
| **Historia de Usuario** | |
|
| Numero: 9 | Nombre: Publicar |
| Usuario: | |
| Modificación Historia: | Iteración Asignada: |
| Prioridad en Negocio: Baja | Puntos Estimados: 3 |
|
| Riesgo en Desarrollo: Alta |
|
| Descripción: | |
| Se debe poder publicar un inmueble en el sitio web de la empresa con los siguientes datos:   * código inmueble (Integer), * tipo de inmueble (TipoInmueble), * Localidad (String:50), * Dirección, * Barrio (String:50), * Cantidad de dormitorios (Integer), * Baños (Integer), * Garaje (Integer), * Patio (Boolean), * Superficie terreno (Double), * Superficie edificada (Double), * Precio (Double), * Fotos del inmueble y * Video del tour dentro del inmueble.   Se mostrarán los datos disponibles del inmueble.  Nunca se mostrará el propietario del inmueble.  Se debe mostrar los siguientes datos de contacto de la inmobiliaria:   * Nombre, * Dirección, * Teléfono, * Fax, * Correo electrónico. | |
|
|
|
| Observaciones:  Al crear una publicación se debe eliminar la anterior. | |
|

# Estimación de Esfuerzo, Riesgo y Prioridad

Para el equipo de desarrollo, un Story Point es equivalente a una semana, de 5 días, de 3 horas de trabajo cada día.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Historia** | | **Story Points** | **Prioridad en Negocio** | **Riesgo en Desarrollo** |
| **Numero** | **Nombre** |
| 1 | ABM Vendedor | 2 | Baja | Baja |
| 2 | ABM propietario | 2 | Alta | Baja |
| 3 | ABM inmueble | 2 | Alta | Media |
| 4 | Consulta Inmueble | 1 | Alta | Baja |
| 5 | Catálogo de Inmuebles | 2 | Media | Alta |
| 6 | ABM cliente | 2 | Media | Baja |
| 7 | Generar Reserva | 1 | Media | Baja |
| 8 | Venta | 2 | Media | Baja |
| 9 | Publicar | 3 | Baja | Alta |

|  |  |  |  |
| --- | --- | --- | --- |
| **Riesgo** | **Prioridad** | | |
| **Alta** | **Media** | **Baja** |
| **Alto** |  | 5 | 9 |
|
|
| **Medio** | 3 |  |  |
|
|
| **Bajo** | 2, 4 | 6, 7, 8 | 1 |
|
|

(comentar la interacci�n con el cliente)

# Spikes

SPIKES

# Definición de Velocidad y Alcance

* Total de historias de usuario = 17 Story Point.

Para el primer release del sistema definimos con el cliente que no se va a implementar la historia de usuario 9.

* Alcance: 1, 2, 3, 4, 5, 6, 7 y 8.
* Total de historias de usuario para este release = 14 Story Point.
* Tiempo de iteración promedio = 3 semanas.

Estimamos que en 3 semanas una pareja de desarrollo puede realizar 3 Story Points. Esto nos da una velocidad de 9 Story Points por iteración, pero la definimos en 8 para tener un margen de maniobra en el caso de existir problemas en el desarrollo.

* Velocidad de desarrollo = 8 Story Point/iteración (para una iteración de 3 semanas).
* Cantidad de iteraciones = 14 [story point] / 8 [story point/iteración] = 2 iteraciones.

# Metáfora

(comentar que se eligi� y porqu�)

# Fecha de Inicio del desarrollo

Fecha de inicio: martes 18 de octubre de 2016.

# Planificación Iteración 1

Fecha de inicio: martes 18 de octubre de 2016.

Fecha de finalización: martes 8 de noviembre de 2016.

## Tareas

### ABM Vendedor

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Vista y lógica del Login | |  | Programador | 3 | |
|  |  |  |  |  |  |  |
| Fecha | 08/10/16 |  |  | Task Points | 2 |  |
|  |  |  |  |  |  |  |
| Número de Historia | | 1 |  | Número de TaskCard | | 1 |
|  |  |  |  |  |  |  |
| Descripción:  Se presenta una vista con las opciones para el inicio de sesión o para registrarse como un nuevo usuario del sistema, es decir, un vendedor.  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)   Se deben realizar todas las validaciones antes de permitir el ingreso. De existir un error se presenta un cartel indicando lo sucedido.  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 | Terminada 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 | |  | |

* Estimación 2 Task Points.
* Tiempo real 4 Task Points.
* Desvío 2 Task Points de subestimación.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **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:30), * apellido(string:30), * tipoDocumento(TipoDocumento), * numeroDocumento(string:30), * id(Integer), * password(string:100), * salt(String), * Root(Boolean) y * estado(Estado).   Para cada atributo deben hacerse los métodos set y get. | | | | | | |
|
|
|  |  |  |  |  |  |  |
| Notas:  El atributo Root es true para el vendedor con privilegios de administrador en el sistema.  El atributo Estado debe representar el estado de la entidad (alta, baja). | | | | | | |
|
|
| Fecha | Realizado | | A realizar | | Comentarios | |
| 18/10/2016 | Definición completa de la entidad y sus anotaciones | | nada | |  | |

* Estimación 1 Task Points.
* Tiempo real 1 Task Points.
* Desvío 0 Task Points de desvío.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Vista alta, modificar y baja Vendedor | |  | Programador | 2 | |
|  |  |  |  |  |  |  |
| Fecha | 09/10/16 |  |  | 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:30), * Contraseña (String) y * 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 | |  | |  | |

* Estimación 3 Task Points.
* Tiempo real 3 Task Points.
* Desvío 0 Task Points de desvío.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Vista principal para la administración | |  | Programador | 3 | |
|  |  |  |  |  |  |  |
| Fecha | 08/10/16 |  |  | 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:   * Ver mis datos. * Inmueble: alta, modificar, eliminar, consulta, generar reserva, vender. * Propietario: alta, modificar, eliminar. * Cliente: Alta, modificar, eliminar, ver catálogo de inmuebles * Vendedor: alta, modificar, eliminar. (sólo si es administrador)   Si una acción no está permitida, se debe ocultar el botón.  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 | |

* Estimación 2 Task Points.
* Tiempo real 1 Task Points.
* Desvío 1 Task Points de sobreestimación.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Lógica alta, modificación y baja vendedor | |  | Programador | 1 | |
|  |  |  |  |  |  |  |
| Fecha | 07/10/16 |  |  | 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 se lanza 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 lógica del vendedor. | | | | | | |
|  |  |  |  |  |  |  |
| Notas: | | | | | | |
|
|
| Fecha | Realizado | | A realizar | | Comentarios | |
| 19/10 | Test y lógica de alta vendedor | | Test y lógica de modificar y de baja | |  | |
| 20/10 | Test y lógica de modificar | | Lógica de baja | |  | |
| 21/10 | Agregadas verificaciones de datos y casos de prueba | | Lógica de baja | |  | |
| 22/10 | Lógica de baja | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |

* Estimación 4 Task Points.
* Tiempo real 4 Task Points.
* Desvío 0 Task Points de desvío.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Persistidor vendedor | |  | Programador | 1 | |
|  |  |  |  |  |  |  |
| Fecha | 07/10/16 |  |  | 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). * Obtener un vendedor por medio de su tipo y número de documento (si resulta en error lanza excepción SaveUpdateException). * Obtener todos los vendedores (si resulta en error lanza excepción SaveUpdateException).   Todas las excepciones mencionadas extienden de PersistenceException.  Se debe hacer una clase que implemente dicha interfaz mediante hibernate y completar las entidades con las anotaciones necesarias. | | | | | | |
|
|
|  |  |  |  |  |  |  |
| Notas  El método para obtener todos los vendedores debe buscar aquellos vendedores con el estado “Alta”. | | | | | | |
|
|
| Fecha | Realizado | | A realizar | | Comentarios | |
| 23/10 | Definido persistidor | | Manejo de excepciones | |  | |
| 24/10 | Manejo de excepciones | |  | | Completado en menos tiempo de lo estimado por baja complejidad | |

* Estimación 3 Task Points.
* Tiempo real 2 Task Points.
* Desvío 1 Task Points de sobreestimación.

### 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:   * id(Integer), * nombre(string:30), * apellido(string:30), * tipoDocumento(TipoDocumento), * numeroDocumento(string:30), * 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:  El atributo Estado debe representar el estado de la entidad (alta, baja). | | | | | | |
|
|
| Fecha | Realizado | | A realizar | | Comentarios | |
| 20/10/16 | atributos, setters y getters, equals, anotaciones JPA y NamedQueries | |  | |  | |

* Estimación 1 Task Points.
* Tiempo real 1 Task Points.
* Desvío 0 Task Points de desvío.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Vista alta, modificar y baja Propietario | |  | Programador | 1 | |
|  |  |  |  |  |  |  |
| Fecha | 07/10/16 |  |  | 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:30), * Dirección:   + Calle (string:50),   + Número (String:30),   + Piso (String:30),   + Departamento (string:30),   + Otros (String:100)   + País (string:50),   + Provincia (string:50) y   + Localidad (string:50). * Teléfono (String:30) y * Email (string:30).   Los siguientes datos son obligatorios: nombre, apellido, tipo, número de documento, calle, número, país, provincia, localidad, barrio, 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 y 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 | |
| 25/10 | Definido fxml y controlador alta | | Fxml y Controlador modificar y baja | |  | |
| 26/10 | Definido fxml y Controlador modificar | | Fml y controlador de administrar propietarios | |  | |
| 27/10 | Terminados fxml y controlador administrar con funcionalidad baja y transiciones a pantallas alta y modificar | |  | |  | |

* Estimación 3 Task Points.
* Tiempo real 3 Task Points.
* Desvío 0 Task Points de desvío.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Lógica alta, modificación y baja propietario | |  | Programador | 2 | |
|  |  |  |  |  |  |  |
| Fecha | 09/10/16 |  |  | 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.   + País sea un string de como máximo 30 caracteres.   + Provincia sea un string de como máximo 30 caracteres.   + Localidad sea un string de como máximo 30 caracteres.   + Barrio sea un string de como máximo 50 caracteres. * Teléfono sea un String de como máximo 30 caracteres. * Email sea un string de hasta 30 caracteres con formato de correo.   Los siguientes datos son obligatorios: nombre, apellido, tipo, número de documento, calle, número, localidad, provincia, teléfono, barrio.  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 | |  | |  | |

* Estimación 3 Task Points.
* Tiempo real 3 Task Points.
* Desvío 0 Task Points de desvío.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Persistidor propietario | |  | Programador | 2 | |
|  |  |  |  |  |  |  |
| Fecha | 09/10/16 |  |  | 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 de PersistenceException.  Se debe hacer una clase que implemente dicha interfaz mediante hibernate y completar las entidades con las anotaciones necesarias. | | | | | | |
|
|
|  |  |  |  |  |  |  |
| Notas:  El método para obtener todos los propietarios debe buscar aquellos propietarios con el estado “Alta”. | | | | | | |
|
|
| 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 | |  | |  | |

* Estimación 3 Task Points.
* Tiempo real 2 Task Points.
* Desvío 1 Task Points de sobreestimación.

### 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(Boolean), * superficie(double), * antigüedad(Integer), * dormitorios(Integer), * baños(Integer), * garaje(Boolean), * patio(Boolean), * piscina(Boolean), * aguaCorriente(Boolean), * cloacas(Boolean), * gasNatural(Boolean), * aguaCaliente(Boolean), * teléfono(Boolean), * lavadero(Boolean), * pavimento(Boolean).   Para cada atributo deben hacerse los métodos set y get. | | | | | | |
|
|
|  |  |  |  |  |  |  |
| Notas  El atributo Estado es un enumerable con cuatro estados posibles, alta, baja, vendido, no vendido. | | | | | | |
|
|
| 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. | |

* Estimación 1 Task Points.
* Tiempo real 2 Task Points.
* Desvío 1 Task Point de subestimación.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **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 Dirección que tiene los atributos:   * calle(Calle), * número(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:   * id(Integer), * nombre(string:50) y * provincia(Provincia).   La clase Provincia con los atributos:   * id(Integer), * nombre(string:50) y * pais(Pais).   La clase País con atributos:   * id(Integer) y * nombre(string:50).   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 * orientación (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 en los comentarios 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 | |  | |  | |

* Estimación 1 Task Points.
* Tiempo real 2 Task Points.
* Desvío 1 Task Points de subestimación.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Vista alta, modificar y baja inmueble | | | |  | | | Programador | | 3 | | |
|  |  | | |  |  | | |  | |  | |  |
| Fecha | 07/10/16 | | |  |  | | | 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 (String:50), * número (String:50, numérico) (obligatorio), * Piso (String:30), * Departamento (String:30), * otros (String:100), * Tipo de inmueble (L/local-oficina, C/casa, D/departamento, T/terreno, Q/quinta, G/galpón) (obligatorio), * Precio de venta (Double, 2 cifras significativas) (obligatorio), * Orientación (norte, sur, este, oeste, noreste, noroeste, sureste, suroeste), * Medidas:   + Frente en metros (Double),   + Fondo en metros (Double),   + Superficie en m2 (Double). * Datos del edificio:   + Propiedad horizontal (Booleano),   + Superficie en m2 (Double),   + Antigüedad en años (Integer),   + Dormitorios (Integer),   + Baños (Integer),   + Garaje/cochera (Boolean),   + Patio (Boolean),   + Piscina (Boolean),   + Agua corriente (Boolean),   + Cloacas (Boolean),   + Gas natural (Boolean),   + Agua caliente (Boolean),   + Teléfono (Boolean),   + Lavadero (Boolean),   + Pavimento (Boolean). * Fotos (sin restricción de tamaño), * Observaciones (String:300).   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, excepto el código. 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:  La baja se realiza de manera lógica seteando el estado en Baja. | | | | | | | | | | | | |
|
|
| 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 | | | |
|  | |  | | | |  | | |  | | | |
|  | |  | | | |  | | |  | | | |

* Estimación 3 Task Points.
* Tiempo real Task Points.
* Desvío Task Points de subestimación.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Lógica alta, modificación y baja inmueble | |  | Programador | 2 | |
|  |  |  |  |  |  |  |
| Fecha | 09/10/16 |  |  | 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 (independientemente del estado) * fecha de carga sea una fecha bien formada. * propietario sea un propietario previamente cargado (obligatorio). * País: sea un string de como máximo 50 caracteres (obligatorio). * Provincia: sea un string de como máximo 50 caracteres (obligatorio). * Localidad: sea un string de como máximo 50 caracteres (obligatorio). * Calle sea un string de como máximo 50 caracteres alfanuméricos. * Número sea numérico. * Piso sea numérico. * Departamento sea un string de como máximo 30 caracteres. * Barrio sea un string de como máximo 50 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); * Observaciones, string de 300 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 ocurrieron errores de validación se lanza una excepción.  Para la modificación se pueden modificar todos los campos (excepto el código) 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 lanza una excepción.  Para la baja no se deben tener consideraciones especiales y se debe realizar la baja lógica del inmueble seteando el estado en “Baja” | | | | | | |
|  |  |  |  |  |  |  |
| Notas: | | | | | | |
|
| Fecha | Realizado | | A realizar | | Comentarios | |
| 03/11/2016 | Pruebas ABMC lógica inmueble. | | Implementación ABMC inmueble. | |  | |
| 04/11/2016 | Implementación ABMC inmueble. | | Nada | |  | |

* Estimación 4 Task Points.
* Tiempo real 2 Task Points.
* Desvío 2 Task Points de sobreestimación.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Persistidor inmueble | |  | Programador | 2 | |
|  |  |  |  |  |  |  |
| Fecha | 09/10/16 |  |  | 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 por id (si resulta en error lanza excepción ConsultaException). * Obtener todos los inmuebles (si resulta en error lanza excepción ConsultaException)   Todas las excepciones mencionadas extienden de PersistenceException.  Se debe hacer una clase que implemente dicha interfaz mediante hibernate y completar las entidades con las anotaciones necesarias. | | | | | | |
|
|
|  |  |  |  |  |  |  |
| Notas:  El método para obtener todos los inmuebles debe buscar aquellos inmuebles con el estado “Alta”. | | | | | | |
|
|
| Fecha | Realizado | | A realizar | | Comentarios | |
| 01/11/16 | Guardar, modificar, obtener un inmueble y obtener inmuebles | |  | |  | |

* Estimación 3 Task Points.
* Tiempo real 1 Task Points.
* Desvío 2 Task Points de sobreestimación.

### 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:30), * apellido (string:30), * tipoDocumento (TipoDocumento), * numeroDocumento (string:30), * teléfono (string:30), * estado (Estado), * inmuebleBuscado (InmuebleBuscado)   Implementar la clase InmuebleBuscado. La clase debe tener los siguientes atributos:   * cliente(Cliente), * Tipo de inmueble [L/local-oficina, C/casa, D/departamento, T/terreno, Q/quinta, G/galpón] (obligatorio) * Localidad (String:50) * Barrios (Lista de String:50) * 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)   Para cada atributo deben hacerse los métodos set y get. | | | | | | |
|
|
|  |  |  |  |  |  |  |
| Notas:  El atributo Estado debe representar el estado de la entidad (alta, baja). | | | | | | |
|
|
| Fecha | Realizado | | A realizar | | Comentarios | |
| 17/10/2016 | Terminada | | Nada | | No se hacen pruebas porque son solo setters y getters | |

* Estimación 1 Task Points.
* Tiempo real 1 Task Points.
* Desvío 0 Task Points de desvío.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Vista alta, modificar y baja cliente | |  | Programador | 1 | |
|  |  |  |  |  |  |  |
| Fecha | 07/10/16 |  |  | Task Points | 3 |  |
|  |  |  |  |  |  |  |
| Número de Historia | | 6 |  | Número de TaskCard | | 17 |
|  |  |  |  |  |  |  |
| Descripción:  Se presenta una vista con los campos a ingresar por el usuario para realizar el alta de un cliente:   * Nombre (String:30) (obligatorio) * Apellido (String:30) (obligatorio) * Tipo de documento [DNI, LC, LE, Pasaporte, Cédula Extranjera] (obligatorio) * Número de documento (String:30) (obligatorio) * Teléfono (String:30) * Datos del inmueble buscado:   + Tipo de inmueble [L/local-oficina, C/casa, D/departamento, T/terreno, Q/quinta, G/galpón] (obligatorio)   + Localidad (String:50)   + Barrios (Deben ser de la misma localidad seleccionada o estar vacío si no se seleccionó ninguna) (Lista de String:50)   + 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.   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 a modificar 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 | |
| 28/10 | Definido fxml y controladores alta | | Fxml y controladores Modificar, administrar, baja, inmueble buscado | |  | |
| 29/10 | Definido fxml y controlador modificar | | Fxml y controladores administrar-baja e inmueble buscado | |  | |
| 30/10 | Definido fxml y controlador administrar clientes con funcionalidad baja | | Fxml y controlador de cargar inmueble buscado | |  | |
| 31/10 | Terminado fxml y controlador de cargar inmueble buscado | |  | |  | |

* Estimación 3 Task Points.
* Tiempo real 4 Task Points.
* Desvío 1 Task Point de subestimación.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Lógica alta, modificación y baja cliente | |  | Programador | 1 | |
|  |  |  |  |  |  |  |
| Fecha | 07/10/16 |  |  | 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 de números con un máximo de 30 caracteres y corresponda con el tipo de documento. Debe ser único. * Teléfono sea String de números con un máximo 30 caracteres. * Validaciones para los datos del inmueble buscado.   Son obligatorios nombre, apellido, tipo y número de documento.  Si todo es correcto se da de alta el cliente en la base de datos. Si ocurre un error se lanza 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. Si ocurre un error se lanza una excepción.  Para la baja no se deben tener consideraciones especiales y se debe realizar la baja del propietario de manera lógica cambiando el estado a “Baja”. | | | | | | |
|  |  |  |  |  |  |  |
| Notas: | | | | | | |
|
| Fecha | Realizado | | A realizar | | Comentarios | |
| 1/11 | Test y lógica de alta | | Tests y lógica de modificar y de baja | |  | |
| 2/11 | Terminado modificar y baja | |  | |  | |

* Estimación 4 Task Points.
* Tiempo real 2 Task Points.
* Desvío 2 Task Point de sobreestimación.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Persistidor cliente | |  | Programador | 3 | |
|  |  |  |  |  |  |  |
| Fecha | 10/10/16 |  |  | 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). * Obtener un cliente por tipo y número de documento (si resulta en error lanza excepción SaveUpdateException). * Obtener todos los clientes (si resulta en error lanza excepción SaveUpdateException).   Todas las excepciones mencionadas extienden de PersistenceException.  Se debe hacer una clase que implemente dicha interfaz mediante hibernate y completar las entidades con las anotaciones necesarias. | | | | | | |
|
|
|  |  |  |  |  |  |  |
| Notas  El método para obtener todos los clientes debe buscar aquellos clientes con el estado “Alta”. | | | | | | |
|
|
| Fecha | Realizado | | A realizar | | Comentarios | |
| 02/11/2016 | Terminadas pruebas persistidor | | Programar persistidor | |  | |
| 03/11/2016 | Terminado de programar persistidor | | Nada | |  | |

* Estimación 3 Task Points.
* Tiempo real 2 Task Points.
* Desvío 1 Task Points de sobreestimación.

## Estimación de Esfuerzo

Sumatoria de desvíos =

Total de task points realizados = 39 + task(13) + task(5) =

## Conclusiones parciales de lo estimado

# Planificación Iteración 2

Fecha de inicio: miércoles 30 de noviembre de 2016.

Fecha de finalización: miércoles 14 de diciembre de 2016.

## Tareas

### Consulta inmueble

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Vista de consulta inmueble | |  | Programador | 1 | |
|  |  |  |  |  |  |  |
| Fecha | 27/11/16 |  |  | 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), * precio máximo (Double), * precio mínimo (Double) y * 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 el tipo, ubicación (país provincia, localidad, barrio, calle y número), nombre y apellido del propietario.  Luego de la consulta se permitirá seleccionar algún resultado y 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. * Ver reservas. * Ver más (muestra la pantalla de modificar inmueble con los campos correspondientes rellenados y deshabilitados).   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 “Generar catálogo de inmueble”. | | | | | | |
|  |  |  |  |  |  |  |
| Notas:  Los campos de dirección con ComboBox permitirán ingresar texto y el criterio de la búsqueda será “contiene”. | | | | | | |
|
|
| Fecha | Realizado | | A realizar | | Comentarios | |
| 30/11 | Agregada sección de búsqueda al fxml de administrarInmueble y atributos asociados en su controlador | | Lógica de control y de transiciones del controlador | | Se agregó la búsqueda a una pantalla existente | |
| 1/12 | Lógica de control y de transiciones del controlador | | Nada | |  | |

* Estimación 2 Task Points.
* Tiempo real 2 Task Points.
* Desvío 0 Task Points de desvío.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Lógica y persistidor de consulta inmueble | |  | Programador | 3 | |
|  |  |  |  |  |  |  |
| Fecha | 27/11/16 |  |  | 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, rango de precios y estado).  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:  Los inmuebles que se obtienen de la base de datos deben tener el estado “Alta”. | | | | | | |
|
|
| 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 | |  | |

* Estimación 3 Task Points.
* Tiempo real 3 Task Points.
* Desvío 0 Task Points de desvío.

### Catálogo de inmuebles

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Vista Alta Catálogo Inmueble | |  | Programador | 3 | |
|  |  |  |  |  |  |  |
| Fecha | 27/11/16 |  |  | 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 un cliente, se habilita la opción de generar un catálogo para el mismo. También puede ser invocado desde la vista administración de inmuebles, al seleccionar uno o varios inmuebles y presionar 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 la administración de clientes) 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 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 en otra vista, con las opciones para imprimir y/o guardar.  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 | |
|  |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |

* Estimación 3 Task Points.
* Tiempo real Task Points.
* Desvío Task Points de subestimación.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Lógica alta catálogo de inmueble y generar PDF | |  | Programador | 2 | |
|  |  |  |  |  |  |  |
| Fecha | 26/11/16 |  |  | 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 | |
| 12/12/16 | Lógica alta catálogo y manejo de errores | | Definir estructura del PDF y lógica para generar PDF | |  | |
| 13/12/16 | Hecho generar PDF | |  | |  | |

* Estimación 3 Task Points.
* Tiempo real 2 Task Points.
* Desvío 1 Task Point de sobreestimación.

### Generar reserva

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Vista alta, baja y listar reserva | |  | Programador | 2 | |
|  |  |  |  |  |  |  |
| Fecha | 26/11/16 |  |  | 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 (ComboBox<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 listarse en un comboBox los inmuebles de la base de datos.  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 | |
| 7/12/16 | Vista de listar reservas | | Vista alta y baja | |  | |
| 8/12/16 | Vista alta y baja reserva | |  | |  | |
| 9/12/16 | Transiciones desde listar inmueble y listar cliente | |  | |  | |

* Estimación 3 Task Points.
* Tiempo real 3 Task Points.
* Desvío 0 Task Points de desvío.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Lógica alta, baja reserva y generar PDF | |  | Programador | 3 | |
|  |  |  |  |  |  |  |
| Fecha | 28/11/16 |  |  | Task Points |  | 3 |
|  |  |  |  |  |  |  |
| Número de Historia | | 7 |  | Número de TaskCard | | 25 |
|  |  |  |  |  |  |  |
| Descripción:  Se recibe de la vista un objeto Reserva con 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 y que estos tengan los datos necesarios para generar el PDF. 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, no se deben tener consideraciones especiales y se debe realizar una baja lógica de la reserva. 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. | |  | |  | |

* Estimación 3 Task Points.
* Tiempo real 3 Task Points.
* Desvío 0 Task Points de desvío.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Lógica envío de mail | |  | Programador | 2 | |
|  |  |  |  |  |  |  |
| Fecha | 28/11/16 |  |  | 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, un mensaje y un 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 | |
| 6/12/16 | Hecha lógica envío de mail | |  | | Se ha utilizado la API de Gmail para simplificar la lógica del envío de mail.  2 horas. | |
| 7/12/16 | Corregidos errores presentados en Windows | |  | | En Windows se presentaba un error que impedía el envío de mails, mientras que en Linux funcionaba correctamente.  1 hora. | |

* Estimación 1 Task Points.
* Tiempo real 1 Task Points.
* Desvío 0 Task Points de desvío.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Cambios ABM Cliente | |  | Programador | 3 | |
|  |  |  |  |  |  |  |
| Fecha | 28/11/16 |  |  | 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 de alta y modificación de un cliente, y modificar la entidad Cliente. | | | | | | |
|  |  |  |  |  |  |  |
| Notas:  Recordar actualizar las pruebas. | | | | | | |
|
|
| Fecha | Realizado | | A realizar | | Comentarios | |
| 2/12/2016 | Todo | | Nada | | Se actualizaron los casos de prueba y se verificó que no se hayan introducido errores. | |

* Estimación 1 Task Points.
* Tiempo real 1 Task Points.
* Desvío 0 Task Points de desvío.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Persistidor y entidad reserva | |  | Programador | 2 | |
|  |  |  |  |  |  |  |
| Fecha | 28/11/16 |  |  | 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 de 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:  El atributo Estado debe representar el estado de la entidad (alta, baja).  El método para obtener todas las reservas debe buscar aquellas reservas con el estado “Alta”. | | | | | | |
|
|
| Fecha | Realizado | | A realizar | | Comentarios | |
| 30/11/16 | Entidad Reserva, agregados atributos y anotaciones JPA a las clases relacionadas, métodos guardar, modificar y obtener reservas | |  | |  | |

* Estimación 1 Task Points.
* Tiempo real 1 Task Points.
* Desvío 0 Task Points de desvío.

### Ventas

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Vista alta y listar venta | |  | Programador | | 1 | |
|  |  |  |  |  | |  |  |
| Fecha | 29/11/16 |  |  | Task Points | |  | 3 |
|  |  |  |  |  | |  |  |
| Número de Historia | | 8 |  | 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 cargado en el sistema.  Se presenta 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 una vista para visualizar el documento de venta en PDF generado por la lógica. Se permiten realizar dos acciones: imprimir y guardar.  Para la vista de listar ventas, se debe tener en cuenta de donde es llamada, 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 documento PDF generado al concretar la venta. | | | | | | | |
|  |  |  |  |  |  | |  |
| Notas: | | | | | | | |
|
|
| Fecha | Realizado | | A realizar | | Comentarios | | |
| 2/12 | Tests del controlador de alta venta, fxml de alta venta | | Controlador de alta venta, fxml y controlador de confirmar contraseña; test, fxml y controlador de listar venta | |  | | |
| 3/12 | Controlador de alta venta, fxml y controlador de confirmar contraseña | | test, fxml y controlador de listar venta | |  | | |
| 4/12 | test, fxml y controlador de listar venta | | nada | |  | | |

* Estimación 3 Task Points.
* Tiempo real 3 Task Points.
* Desvío 0 Task Points de desvío.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Lógica alta venta y generar PDF | |  | Programador | | 1 | |
|  |  |  |  |  | |  |  |
| Fecha | 28/11/16 |  |  | Task Points | |  | 3 |
|  |  |  |  |  | |  |  |
| Número de Historia | | 8 |  | 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 asigna una fecha de venta.  Se valida que el inmueble se encuentre en el estado “No vendido”.  Se valida 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.  Setear el estado del inmueble a “Vendido”. | | | | | | | |
|  |  |  |  |  |  | |  |
| Notas  La generación del PDF se debe hacer en una clase aparte encargada de esto. | | | | | | | |
|
|
| Fecha | Realizado | | A realizar | | Comentarios | | |
| 5/12 | Test de alta venta y lógica de alta venta | | Test y lógica de generar pdf de venta | |  | | |
| 6/12 | Test y lógica de generar pdf | | nada | | Nos llevó menos tiempo por experiencia y soluciones similares realizadas anteriormente que ayudaron | | |

* Estimación 3 Task Points.
* Tiempo real 2 Task Points.
* Desvío 1 Task Point de sobreestimación.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Lógica imprimir venta | |  | Programador | 1 | |
|  |  |  |  |  |  |  |
| Fecha | 28/11/16 |  |  | 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 | |
| 7/12 | Estructura (clase y método que no hace nada) para que pueda ser referenciado | | Lógica de la impresión | | No llevó un task point ya que tomó solo un par de minutos | |
| 9/12 | Lógica de la impresión | | nada | | Nos llevó más tiempo de lo estimado ya que nos encontramos con problemas para realizar este tipo de impresión | |

* Estimación 1 Task Points.
* Tiempo real 2 Task Points.
* Desvío 1 Task Points de subestimación.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Persistidor y entidad venta | |  | Programador | 2 | |
|  |  |  |  |  |  |  |
| Fecha | 29/11/16 |  |  | 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 de 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:  El atributo Estado debe representar el estado de la entidad (alta, baja).  Los métodos para obtener todas las ventas deben buscar aquellas ventas con el estado “Alta”. | | | | | | |
|
|
| Fecha | Realizado | | A realizar | | Comentarios | |
| 2/12/16 | Anotaciones JPA, clase venta, relaciones con las demás clases y método guardar venta | |  | | No fue necesario realizar los métodos obtener ventas ya que las ventas pueden ser obtenidas directamente de las clases vendedor, propietario e inmueble | |

* Estimación 2 Task Points.
* Tiempo real 1 Task Points.
* Desvío 1 Task Points de sobreestimación.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task Card** | Agregar EstadoInmueble a inmueble | |  | Programador | 1 | |
|  |  |  |  |  |  |  |
| Fecha | 26/11/16 |  |  | Task Points |  | 1 |
|  |  |  |  |  |  |  |
| Número de Historia | | 8 |  | 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.  Se debe agregar un campo de EstadoInmueble al HistorialInmueble. Para esto hay que modificar la lógica de modificación del inmueble y modificar la entidad HistorialInmueble.  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 que carga los datos necesarios para ejecutar el sistema. | | | | | | |
|  |  |  |  |  |  |  |
| Notas  Recordar actualizar las pruebas. | | | | | | |
|
|
| Fecha | Realizado | | A realizar | | Comentarios | |
| 7/12 | Todo | | nada | | 1 hora. | |

* Estimación 1 Task Points.
* Tiempo real 1/3 Task Points.
* Desvío 2/3 Task Points de sobreestimación.

## Estimación de Esfuerzo

Sumatoria de desvíos =

## Conclusiones parciales de lo estimado

# Código Fuente

(debe estar ordenado por iteración y por tarea - debe estar comentado)

# Pruebas de Unidad

(deben especificarse a que tarea corresponden)

# Refactorización

(deben especificarse a que tarea corresponden)

# Estándares de programación

# Conclusiones

(deben ser exahustivas respecto al trabajo realizado y la metodología, criticando a favor y en contra de la misma en base a la experiencia adquirida)