Laboratorio de Computación III

Programa Inmobiliario

Integrantes: Mayor Joaquín Ignacio
Milano Facundo
Tosini Ignacio

Introducción:

.

La aplicación a desarrollar consiste en un programa de gestión para una inmobiliaria, la cual cuenta con un sistema de interacción del usuario, el cual va a poder crearse una cuenta y a partir de la misma va a poder consultar y alquilar/comprar el tipo de inmueble que desea (El mismo cuenta con casas, departamentos, locales y cocheras). A su vez cuenta con un sistema de administrador, propio de los "dueños" de la inmobiliaria el cual le va a permitir gestionar y administrar los diferentes inmuebles con los que cuenta o vaya a contar, permitiendo incorporar nuevos inmuebles, dar de baja los ya existentes, editar algún dato de los inmuebles, dar de baja a usuarios y ver toda la información con respecto a los usuarios y a los inmuebles.

Se optó por el desarrollo del mismo, ya que en equipo se cuenta con conocidos/familiares que se encuentran en el área y pueden brindarnos información acerca del funcionamiento del negocio inmobiliario. Una vez obtenida toda la información del funcionamiento del mundo inmobiliario se procedió a diagramar la aplicación con sus respectivas clases y métodos.

Informe Técnico:

El desarrollo de la aplicación comenzó el diagrama de los **usuarios** que iban a utilizar el sistema, para ello se realizó una clase usuario la cual cuenta con los siguientes atributos:

- nombreYApellido : String
- contraseña : Contraseña
- dni : String
- mail : Mail
- edad : int
- historial : ArrayList<String>
- facturas : HashMap<Integer, Factura>
- estado : boolean

El atributo contraseña pertenece a la clase **Contraseña**, la cual almacena un String con la contraseña del usuario, esta clase se decidió hacer debido a que la misma cuenta con un método propio estático el cual permite validar si la contraseña establecida a la hora de registrarse cuanta con los criterios mínimos para que la contraseña sea segura (tener mínimo 8 caracteres, una mayúscula y un número).

Por otro lado el mail pertenece a la clase **Mail** el cual almacena en un String el mail del usuario, el motivo de esta clase es similar al de la contraseña, ya que el mismo cuenta con métodos especiales para validar si un mail, efectivamente lo es, comprobando

la presencia del @ y del .com, a su vez para la facilitación de la creación de mails a la hora de registrarse se estableció un Enum llamado Tipos Mail el cual cuenta con la parte final de los tipos de mails más usados (Gmail, Hotmail y Yahoo).

La clase usuario también cuenta con los siguientes métodos:

- ➢ comprobarAdmin(Usuario usuario) : boolean [Static]
- ➢ agregar(String dato) : void
- ➢ agregar(Factura factura) : void
- ➢ mostrarFacturas() : String

El método **comprobarAdmin** se encarga de comprobar a la hora de hacer el login en el sistema si el los datos del usuario pertenecen al administrador o a otro usuario, esto es para que posteriormente lo envíe el menú y las opciones correspondiente al tipo de usuario.

Los 2 métodos de **agregar** se encargan de incorporar cuando sea necesario en el momento de la venta o alquiler los datos de la transacción realizada, siendo en el **agregar(String dato)** almacena las fechas en las cuales se hizo alguna transacción, y el **agregar(Factura factura)** almacena las facturas perteneciente a ese usuario. Por último el método de **mostrarFacturas** muestra todas las facturas perteneciente a ese usuario, por si él mismo quiere realizar el registro de las mismas.

Una vez establecido las bases del usuario se comenzó con el diagrama de las clases pertenecientes a los inmuebles con los que se va a trabajar estos son:

1. Casa
2. Departamento
3. Cochera
4. Local

Las clases **Casa** y **Departamento** heredan de la clase abstracta **Vivienda,** esta va a contar con los siguientes atributos:

- ➢ disponibilidad : ArrayList<Fecha>
- ➢ estado : Estado
- ➢ direccion : String
- ➢ ambientes : short
- ➢ cantBanios : short
- ➢ metrosCuadrados : int
- ➢ amueblado : boolean
- ➢ cochera : boolean
- ➢ precio :double

La clase **Casa** a su vez incorpora 2 nuevos atributos los cuales son:

- patio : boolean
- pisos : short

Mientras que la clase **Departamento** incorpora:

- nroPiso : String
- disposicion : String

La clase **Cochera** cuenta con los siguientes atributos:

- disponibilidad : ArrayList<Fecha>
- direccion : String
- estado : Estado
- piso : short
- posicion : short
- medioDeAcceso : String
- precio : double

Y la clase **Local** cuanto con los siguientes:

- disponibilidad : ArrayList<Fecha>
- direccion : String
- estado : Estado
- ambientes : short
- vidriera : boolean
- precio : double

En las 3 clases principales pertenecientes a los inmuebles (local, cochera y vivienda), cuentan con un atributo perteneciente al **Enum Estado**, el cual determina si se encuentra “EnVenta, “EnAlquiler” o “Baja” y con una ArrayList de Fechas que se utiliza para determinar en caso de que se alquile el inmueble que fechas ya no se encuentran disponibles y si se vende cual es la fecha en la que se vendio. Dichas clases además implementan las siguientes interfaces: IMetodoDePago (método que determina cómo se va a realizar el pago del inmueble y su precio final según el tipo de inmueble), IBuscar (Método para buscar un inmueble en particular) e IBaja (método para dar de baja el inmueble).

Ya terminado con la conformación de estas clases se procedió a realizar la clase **Inmobiliaria** que va a ser la clase envoltorio de todas las clases previamente mencionadas, la cual va a contar con los siguientes atributos:

- viviendas : Abm<Vivienda>
- cocheras : Abm<Cochera>
- locales : Abm<Local>

- usuarios : HashMap<String, Usuario>
- facturas : HashMap<Integer, Factura>
- nombre : String
- direccion : String
- telefono : String
- correo : String

Los atributos de viviendas, cocheras y locales, pertenecen a la clase genérica **Abm<T>** , dicha clase se utiliza para incorporar comportamiento iguales a estos 3 tipos de atributos (siendo estos como HashSets) que van a almacenar cada uno un conjunto del tipo de un inmueble, ya que sus acciones a la hora de alquilar y vender van a ser iguales y cuenta con los siguientes atributos:

- miHashSet : HashSet<T>
- bajas : HashSet<T>

E incorporando los siguientes métodos:

- agregar(T elemento) : void
- baja(T elemento) : boolean
- ponerEnBaja(T elemento) : void
- modificar(T elemento) : boolean
- buscador(String direccion) : T
- listado(String nombreClase) : String
- toJsonGenerico() : JSONObject

Volviendo a la clase **Inmobiliaria** todos los usuarios de la misma se almacenan en un HashMap en cual tendrán como clave el mail del propio usuario, algo similar ocurre con las facturas el cual la key será la propia id de la factura.

Los métodos a utilizar por la Inmobiliaria son:

- buscarUsuario(String mail) : Usuario
- darBaja(String mail) : boolean
- listarViviendas(String eleccion) : String
- listarLocales(String eleccion) : String
- listarCocheras(String eleccion) : String
- venta(Usuario usuario, String direccion, String tipo, Fecha fecha) : void
- alquilar(Usuario usuario, String direccion, String tipo, Fecha fecha) : void
- validarDireccion(String direccion) : boolean [Static]
- buscarCasa(String direccion) : Casa
- buscarDepartamento(String direccion) : Departamento
- buscarLocal(String direccion) : Local
- buscarCochera(String direccion) : Cochera
- estadoString(Estado estado) : String [Static] //para la factura

Con relación al método **venta** este se utiliza para dar comienzo al proceso de venta una vez logueado el usuario de un inmueble el mismo comprueba si el inmueble existe y si esta en venta, este solicita el usuario logueado, la dirección del inmueble a comprar, el tipo de inmueble que desea comprar y la fecha del dia de la compra creada a partir de un LocalDate localdate = LocalDate.now en el menu anterior al método. Una vez ingresados esos datos, comprueba que los datos del inmueble son válidos con el método de la inmobiliaria **validarDireccion**, en caso de ser válidos procede consultar el método de pago y en (efectivo, débito o crédito) y en base a la elección se procede a calcular el precio final, para luego crear la factura, para cargarla en el usuario que realizó la compra, y en la inmobiliaria a modo de registro, por otro lado también da de baja la propia vivienda, para que no se vuelva a comprar, quedando guardada por si en un futuro quiere volver a comprar o alquilar.

El método de **alquilar** funciona similar al de venta en lo que a validación del inmueble y generación de factura, pero en vez de darla de baja, guarda las fechas alquiladas que irá comprobando siempre que se quiere alquilar para verificar que la fecha introducida no esté el inmueble alquilado. Para la creación de la factura fue necesario crear el método estadoString, el cual convierte el estado de “enAlquiler” en un string “alquilado” para la factura.

Una vez definidas y desarrolladas todas estas clases podemos mencionar a la clase factura que se encuentra interactuando con varios métodos y clases que se fueron desarrollando la clase **Factura** se encuentra conformada de la siguiente manera:

- id : int
- nombre : String
- dni : String
- mail : Mail
- precioFinal : double
- fecha : Fecha
- dirInmobiliaria : String
- dirInmueble : String
- estadoActual : String

Creando objetos del tipo Factura a partir de los datos de los inmuebles y usuarios al momento de un alquiler o venta.

Para implementar los diferentes menús, muestra de datos por pantalla y la utilización de los scanner para que el usuario pueda interactuar con la aplicación se creó una clase **ControladoraUsuario** la cual va a tener todos los métodos relacionados a los diferentes menús e interacciones para ingresar datos, permitiendo la buena interacción de todas las clases modificar, dar baja, poder

alquilar, vender, buscar y ver datos. A continuación se muestra una lista de todos los métodos:

- menu(Inmobiliaria inmobiliaria) : Usuario [Static]
- menuAdmin(Inmobiliaria inmobiliaria) : void [Static]
- login(Inmobiliaria inmobiliaria) : Usuario [Static]
- registrarse(Inmobiliaria inmobiliaria) : Usuario [Static]
- menuTipoMail(int eleccion) : String [Static]
- menuUsuario(Inmobiliaria inmobiliaria, Usuario usuario) : void [Static]
- crearFechaAlquiler() : LocalDate [Static]
- mostrarUsuario(Inmobiliaria inmobiliaria) : void [Static]
- darBajaUsuario(Inmobiliaria inmobiliaria) : void [Static]
- darDeBajaInmueble(Inmobiliaria inmobiliaria) : void [Static]
- modificarInmueble(Inmobiliaria inmobiliaria) : void [Static]
- modificarCasa(Inmobiliaria inmobiliaria) : void [Static]
- modificarDepartamento(Inmobiliaria inmobiliaria) : void [Static]
- modificarLocal(Inmobiliaria inmobiliaria) : void [Static]
- modificarCochera(Inmobiliaria inmobiliaria) : void [Static]
- listarInmueble(Inmobiliaria inmobiliaria) : void [Static]
- buscarInmueble(Inmobiliaria inmobiliaria) : void [Static]
- agregarInmuebles(Inmobiliaria inmobiliaria) : void [Static]
- agregarCasa(Inmobiliaria inmobiliaria) : void [Static]
- agregarDepartamento(Inmobiliaria inmobiliaria) : void [Static]
- agregarLocal(Inmobiliaria inmobiliaria) : void [Static]
- agregarCochera(Inmobiliaria inmobiliaria) : void [Static]

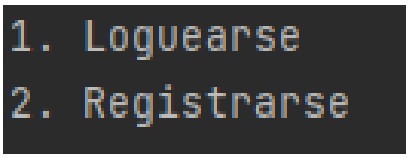
En el **menu** se encuentran las opciones de registrarse o loguearse, en caso de registrarse llamará al método **registrarse** y procederá a solicitar los datos de creación de usuario, validando en cada instancia de que los datos sean correctos. En cambio si desea loguearse el método **login** solicitara el mail y la contraseña para poder ingresar, en caso de que el mail y la contraseña coincidan con la del administrador **(Mail : admin@gmail.com Contraseña: Admin123)** se accede al método **menuAdmin** el cual tendrá opciones de agregar modificar, dar de baja inmuebles, así como ver y dar de baja usuario, cada uno con sus respectivos métodos con sus submenús solicitando los datos necesarios.

En cambio si el usuario que ingresa no es el administrador se abre otro submenú del método **menuUsuario** donde el usuario podrá ver los distintos inmuebles, alquilar, comprar o ver sus facturas. Si desea alquilar un inmueble este le solicitara la fecha que desea alquilar, y cuantos días desea estar, realizando la posterior validación de si la fecha existe, es posterior a la fecha actual y si se encuentra alquilada el inmueble en ese periodo.

Manual de Usuario:

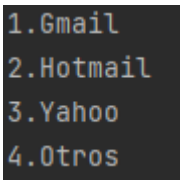
Bienvenido a la aplicación Inmobiliaria a continuación se describe el funcionamiento del mismo y los pasos a seguir:

Cuando iniciamos el programa nos va dar a elegir 2 opciones loguearse o registrarse.



Si aún usted no tiene una cuenta debe escribir 2 para registrarse.

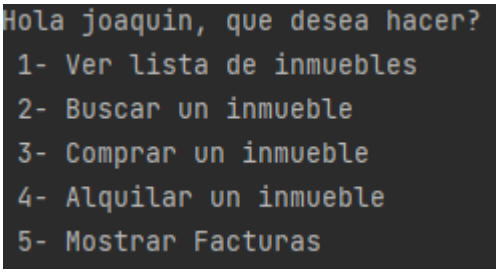
Luego le comenzará a solicitar diferentes datos personales para poder crear su cuenta entre esos datos esta, el nombre y apellido, la contraseña (la cual tiene que tener como mínimo 8 caracteres, una mayúscula y un número), su dni, su edad, el tipo de mail que utiliza:



En caso de elegir otros va a tener que escribir el mail completo, pero si elige cualquiera de las otras 3 opciones solo deberá ingresar la parte delantera del @.

Una vez registrado ya puede elegir la opción 1 para loguearse, solicitando el mail y la contraseña.

Una vez dentro tendremos a nuestra disposición 5 opciones.



Estas opciones las elegiremos ingresando el numero perteneciente a dichas opciones.

Si elegimos la 1 nos preguntará que tipo de inmueble queremos ver (casa, departamento, local o cochera), ingresamos el tipo de inmueble y procederá a mostrarnos una lista de todos los inmuebles solicitados.

La opción 2 también comenzará solicitandonos el tipo de inmueble a buscar, además de la dirección del inmueble en particular que deseamos ver.

```
Que tipo de inmueble desea buscar? (Casa/Departamento/Local/Cochera)
casa
Ingrese la direccion del inmueble:
colon 1000
```

En caso de no estar el inmueble este nos dará un aviso del mismo.

```
Ingrese la direccion del inmueble:
brwon 2337
Dirección no existente
¿Desea hacer otra acción? Si es así ingrese si
```

La opción 3 corresponde a si queremos comprar algunos de los inmuebles en venta, solicitando el tipo de inmueble a comprar, su dirección y elegir el método de pago que pensamos realizar.

```
Ingrese el tipo de inmueble que desea comprar. (Casa, Departamento, Local, Cochera)
local
Ingrese la direeción del inmueble que desea comprar
cas 123
¿Cómo desea Pagar?
1. Efectivo
2. Targeta Debito
3. Targeta Credito
```

La opción 4 nos permitirá hacer el alquiler de un inmueble disponible, solicitando los mismos datos que en la venta, pero sumando el día que se quiere ingresar al inmueble y la cantidad de días que quiere estar.

```
Ingrese el tipo de inmueble que desea alquilar. (Casa, Departamento, Local, Cochera)
casa
Ingrese la direeción del inmueble que desea alquilar
Colon 1000
Ingrese la fecha de ingreso:

Ingrese el dia:
30
Ingrese el mes:
6
Ingrese el año:
2025
Ingrese cuantos dias desea estar:
10
¿Cómo desea Pagar?
1. Efectivo
2. Targeta Debito
3. Targeta Credito
1
```

Si se alquiló con éxito nos preguntará si queremos hacer otra acción, en caso contrario nos avisará que la fecha no está disponible.

```
Esa fecha no se encuentra disponible
Fechas ocupadas: Fecha{fechaEntrada=2024-05-20, fechaSalida=2024-05-30}
```

Por último, si se elige la opción 5 mostrará todas las facturas pertenecientes a su usuario.

En caso de que se ingrese el usuario del admin al loguearse:
(Mail : admin@gmail.com Contraseña: Admin123)

Se abrirá el siguiente menú:

```
¿Qué le gustaría realizar?
1. Agregar inmueble
2. Remover inmueble
3. Modificar inmueble
4. Listar inmueble
5. Mostrar inmueble
6. Dar de baja usuario
7. Mostrar usuario
```

Si se elige la opción 1 le solicitará los datos necesario con relación al tipo de inmueble a cargar. Ejemplo:

```
Que tipo de inmueble desea agregar? (Casa/Departamento/Local/Cochera)
local
Direccion del local:
Brown 2446
Cantidad de ambientes?
2
Tiene vidrieras? (1 -Si/ 2 -No)
1
Precio del inmueble?
900000
Desea alquilar o vender? (1 -Alquiler/ 2 -Venta)
2
```

La opción 2 nos solicitará los datos pertenecientes al inmueble a eliminar “dar de baja”.

```
Que tipo de inmueble desea dar de baja? (Casa/Departamento/Local/Cochera)
casa
Ingrese la direccion del inmueble:
colon 1000
Se dio de baja con exito
```

La opción 3 nos preguntará cual es el inmueble el cual quisiéramos modificar un atributo, una vez ingresado los datos nos preguntará cual de los datos, ingresando el valor de las opciones dadas (números) queremos modificar.

```
Que inmueble desea modificar? (Casa/Departamento/Local/Cochera)
local
Ingrese la direccion del inmueble:
brown 2446
Que atributo desea modificar? (1-estado, 2-piso, 3-posicion, 4-medio de acceso,5-precio)
```

Permitiéndonos posteriormente ingresar el nuevo valor de esa característica.

La opción 4 nos permitirá hacer una lista de todos los inmuebles de una categoria.

La opción 5 nos solicitará el tipo de inmueble y su dirección para mostrar posteriormente todas sus características.

La opción 6 nos va a permitir dar de baja un usuario ya existente en caso de que no queramos que siga utilizando la aplicación. Solicitandonos el mail. Si no se marca ningún error, el usuario se da de baja con éxito.

```
Que usuario desea dar de baja? (Mail del Usuario)
joaquin@gmail.com
Desea dar de baja otro usuario?
```

Por último la opción 7 nos permite buscar y mostrar los datos de un usuario, buscándolo por el mail.

Matriz de Soluciones:

Fecha	Contribuidor	Problema	Solucion
5/6/2023	Joaquin Mayor	Comprobacion de la contraseña en la clase **Usuario**.	Creacion de la clase **Contraseña**.
10/6/2023	Ignacio Tosini	Problematica al manejar fechas en la clase **Usuario** y las clases de los diferentes inmuebles.	Creacion de la clase **Fecha**.
12/6/2023	Joaquin Mayor	Error en la clase **Inmobiliaria** al tratar de alquilar	Comprobacion de de objeto != a null.
14/6/2023	Facundo Milano	Complicaciones al tener que dar de baja un tipo de inmueble.	Se agrego un **TreeSet** adicional a la clase **Abm** para los inmuebles dados de baja.
17/6/2023	Facundo Milano	Fallo al dar de baja a los usuarios.	Correccion en la funcion **darBaja()**
18/6/2023	Ignacio Tosini	Error al cargar las facturas cuando se alquila.	Se añadieron comprobaciones adicionales en la creacion de la factura.
18/6/2023	Joaquin Mayor	Compliacion al momento de validar si las fechas son validas.	Modificacion de la clase **Fecha** en sus parametros, ahora del tipo **LocalDate**.
19/6/2023	Facundo Milano	Cierre forzoso al ingresar una fecha no existente.	Cambio en las comprobaciones de fecha.
19/6/2023	Joaquin Mayor	**JSON** se guardaba de manera correcta pero faltaban datos al leerlo.	Se corrigio el metodo **FromJson()** para la correcta lectura del mismo.
19/6/2023	Ignacio Tosini	Permitia usuarios dados de baja conectarse.	Correccion en la funcion **darBaja()**.

Diario de Trabajo:

24/5/2023 - Toma de dedicación grupal del proyecto inmobiliario, y diagramado de las clases correspondientes.

29/5/23 - Diagramado grupal de métodos y excepciones de las diferentes clases.
Creación de los diferentes documentos (Manual de usuario e Informe Técnico).

01/06/23 - Realización de las clases relacionada a inmuebles, sin métodos (casa, local, garaje y departamento, realización de la clase usuario, contraseña, y mail, con métodos estáticos de validación.

3/06/23 - Creación de la interfaz de IJson y implementación del mismo en las clases creadas de usuario, mail y contraseña.

5/06/23 - Terminación de la clase usuario, contraseña y mail. Realización del sistema de registración y logueo.

8/06/23 - Creación de la interfaz gráfica de la parte de registro y logueo, además de agregar distintas excepciones necesarias y empezar las funciones del alquiler.

10/6/2023 - Se empezó la realización del método Alquiler y se creó la clase Fecha.

11/6/2023 - Creación de las excepciones para la clase Usuario.

12/06/23 - Debate sobre si asignarle saldo al usuario, el cual se decidió que no, seguimos con las funciones del alquiler y ordenamos en carpetas las distintas clases.

14/06/23 - Terminamos la función de alquiler, cambios en la clase genérica, además de empezar el Abm.

17/06/23 - Terminamos los métodos para el usuario Admin junto con correcciones al menú del usuario..

18/06/23 - Modularización del sistema y cambios en la clase Fechas para un mejor uso.

19/06/23 - Correcciones a los métodos ToJson() y FromJson(), finalizados y listos para su uso.

20/06/23 - Últimas revisiones al proyecto y finalización de la documentación del mismo.

Fuentes Consultadas:

https://javautodidacta.es/tiempo-en-java-localdate-localtime/

https://recursosformacion.com/2022/11/java-8-el-acceso-al-tiempo-con-localdate-localtime-y-localdatetime/