**PROGRAMACION Y LABORATORIO III**

****

**INTEGRANTES:**

* Abril Galarraga
* Mateo Cuevas
* Nahuel Moron

**PROFESOR:**

* Gonzalo Benoffi

**COMISION 2**

**TEMA DEL TP:** Aplicación sobre reserva de vuelos y alojamientos

**RESUMEN DE LA APLICACION**

Nuestra aplicación consta de un Sistema de reservas de viajes y también ofrece la posibilidad de reservar alojamiento, ya sea que se desea realizar un viaje o no. Los viajes se realizan únicamente mediante vuelos, los cuales pueden ser de diferentes tipos, como primera clase, económico, ejecutivo, etc. Por su parte, los alojamientos tienen una capacidad máxima de personas, por lo que, si la cantidad de personas que buscan alojamiento supera la capacidad, no se podrá realizar dicha reserva.

**INFORME TECNICO**

**CLASES:**

* **Pais**: Es una clase que contiene todos los atributos de un país que va a ser leído desde una API. Los métodos que implementa son su constructor, los getters que devuelven a cada atributo, el método toString que devuelve toda la información del país, y 2 metodos de conversión, JAVA🡪JSON y JSON🡪JAVA.
* **ControladoraPaises**: Contiene un solo atributo y el mismo es un mapa ordenado de países, la clave de cada país será su ID, y los métodos que implementa son un constructor para inicializar el mapa, un método de mostrar la información del mapa, y otro método de conversión JSON 🡪 JAVA, en el que se va a leer país por país la API enviada por URL, y cada país leído se va a almacenar en el mapa que forma parte de la clase.
* **ConsumoAPI**: Esta clase nos permite traer la información de tipo JSON a nuestro programa mediante una URL que contiene toda la información.
* **Empresa**: Es una de las clases principales del proyecto, en ella se almacenan 3 atributos los cuales son 3 colecciones, una de vuelos y otra de alojamientos mediante un set, y otra de países mediante un mapa. La clase implementa un constructor para inicializar todas las colecciones, un método toString para vuelos y otro para alojamientos, en el que retorna toda la información. Cada colección tiene sus métodos JSON 🡪 JAVA y JAVA 🡪 JSON, en los que mediante un iterador o un bucle for en el que se recorre la colección y se va pasando la información de un objeto a la vez, mediante el llamado a métodos de otras clases como por ejemplo en la función JSONtoVuelo en donde se llama a un método que se encuentra en la clase “Vuelo”(JSONAvuelos). Estas funciones serán explicadas en sus respectivas clases más tarde. Esta clase implementa una especie de polimorfismo, ya que contiene métodos como “buscarVuelo” que se encuentran repetidos varias veces pero con distintos parámetros, esto es, el método buscarVuelo va a recibir por parámetro lo que desea buscar, o más bien como desea buscar el vuelo, en este caso, por ejemplo, se puede buscar vuelos por precio, por clase, por ID, etc. Lo mismo ocurre con el método “buscarAlojamiento”, en el que se implementa este método 3 veces para buscar por país, por precio y por capacidad.

Otros métodos que implementa la empresa son:

* + Verificar país: Le llega por parámetro un nombre de un país y verifica recorriendo la colección de países si el enviado por parámetro se encuentra dentro de la lista.
  + Buscar país: Tiene dos maneras de buscar un país, por clave o por nombre. Si es por clave, se le pedirá al usuario que ingrese el ID que desea buscar, si es por nombre se le pedirá el nombre.
  + Cargar vuelos/Alojamientos: Son métodos de carga automática de vuelos o alojamientos, la empresa carga sus vuelos o alojamientos automáticamente y no hay una función de carga en un menú para que el usuario no pueda manipular los datos que se manejan en la empresa.
  + Reservar: Se divide en dos métodos, también aplica polimorfismo.

Reservar vuelo: Permite crear una reserva a partir de un vuelo preseleccionado por el usuario, sabiendo la cantidad de pasajeros a reservar. El método retorna la reserva en caso de realizarse de manera correcta, de lo contrario, podría lanzar una excepción.

Reservar alojamiento: Permite crear una reserva a partir de un alojamiento preseleccionado anteriormente por el usuario, con conocimiento de la cantidad de huéspedes a reservar. El método retorna la reserva en caso de realizarse de manera correcta, de lo contrario, podría lanzar una excepción.

* Vuelo: Es un servicio que ofrece la empresa, implementa 2 interfaces, una de calcular precio y otra de verificar disponibilidad. Sus atributos son todas las características que tiene un vuelo, como país de origen, país de destino, cantidad máxima de asientos, pasajeros, etc. implementa tres constructores, un constructor completo, un constructor sin precio y un constructor vacío. Cabe destacar que es una clase abstracta, por lo que no se puede instanciar. Los métodos que contiene son todos los getters para devolver cada atributo, un metodo toString que devuelve toda la información del vuelo, 2 funciones de intercambio de información de JAVA 🡪 JSON y JSON 🡪 JAVA. Tambien cuenta con un metodo de establecer precio, en el que se implementa la función calcular precio de la interfaz ICalcularPrecio. Algunos otros métodos con los que cuenta son:
  + VerificarDisponibilidad: Comprueba si la cantidad de personas que desean reservar un vuelo no superan el límite de asientos que tiene un avión.
  + ModificarDisponibilidad: Si el avión se encuentra disponible para volar, esta función lo va a poner no disponible. Si no se encuentra disponible, la función lo cambiara a disponible.
  + VerificarVuelo:
  + AumentarPasajeros: Una vez que se crea la reserva del vuelo, modifica la cantidad de pasajeros incrementando su valor por la cantidad de pasajeros para los que se reservó ese vuelo.
  + DisminuirPasajeros: Cuando una reserva es cancelada, modifica la cantidad de pasajeros incrementando su valor por la cantidad de pasajeros para los que se reservó ese vuelo.
* VueloEconomico, VueloEjecutivo y VueloPrimeraClase: Extienden de la clase vuelo, son justamente los tipos de vuelo disponibles y los métodos que implementan son los de calcular precio de la interfaz ICalcularPrecio, el método toString que devuelve la información del tipo de vuelo junto a la información de la clase padre, y un método de intercambio de información JAVA 🡪 JSON.
* Reserva: Es una clase genérica que contiene varios atributos entre ellos uno genérico que va a ser convertido a un vuelo o a un alojamiento según se lo diga el programa. Implementa un método de toString y también 2 métodos de conversión de información JSON 🡪 JAVA y JAVA 🡪 JSON.
* Alojamiento: Es otro de los servicios ofrecidos por la empresa, es una clase que contiene todos los atributos comunes de un alojamiento como el lugar, capacidad, disponibilidad, etc., e implementa métodos como calcular precio, el cual es un método de la interfaz ICalcularPrecio, y también otros métodos como ver y modificar disponibilidad, un método toString y un método de intercambio de información JAVA 🡪 JSON.

**INTERFACES:**

* ICalcularPrecio: implementa un método de calcular precio que se va a desarrollar en las subclases de vuelo y alojamiento.
* IVerificarDisponibilidad: implementa un método de verificar si algo está disponible o no, el método se va a desarrollar en las subclases de vuelo y alojamiento.

**EXCEPCIONES:**

* ClavePaisInexistenteException: Se utiliza para cuando el usuario ingresa por consola una clave de un país que no se encuentra registrada en la colección de países o en el archivo de países, o puede ser incorrecta.
* DisponibilidadAgotadaException: Se utiliza para cuando el usuario ingresa por consola una cantidad de pasajeros que supera el límite permitido de asientos del avión, cuando la cantidad de personas que se quiere reservar supera la capacidad máxima de un alojamiento, cuando el vuelo no está disponible para viajar, o para cuando el alojamiento no se encuentra disponible ya que tiene otra reserva hecha.
* PaisInexistenteException: Se utiliza para cuando el usuario ingresa por consola un país que no se encuentra registrado en la colección de países o en el archivo de países.
* ContraseñaIncorrectaException: Se utiliza para cuando la persona que desea ingresar su usuario por consola ingresa una contraseña equivocada que no coincide con la contraseña guardada en el archivo de usuarios.
* UsuarioNoEncontradoException: Se utiliza cuando la persona que está intentando iniciar sesión con su usuario escribe un mail que no se encuentra registrado en el archivo o la colección de usuarios.
* UsuarioRepetidoException: Se utiliza cuando la persona que está intentando registrarse con un nuevo usuario ingresa un mail que ya se encuentra registrado en el archivo o colección de usuarios.

**MANUAL DE USUARIOS**

El funcionamiento de la aplicación comienza a partir del menú de inicio, en donde se tienen 3 opciones:

* **Iniciar sesión**: El inicio de sesión para el usuario le permitirá ingresar al programa e interactuar con las diferentes opciones del sistema, para iniciar sesión, se pide primero un mail y una contraseña, ya registrada anteriormente. Para corroborar los datos ingresados por el usuario, el sistema leerá un archivo en donde tiene cargados todos los usuarios con sus respectivas contraseñas. Esto mediante una función llamada “LeerArchivoUsuarios”, primero verifica que el mail ingresado sea correcto, de no serlo, lanzará una excepción que terminará con la ejecución del programa, informándole al usuario que el mail ingresado es incorrecto. Si el mail ingresado es correcto, procederá a verificar la contraseña ingresada. Al igual que con el mail, si la contraseña ingresada es incorrecta, lanzará una excepción, y caso contrario accederá directamente al menú principal donde se tienen todas las opciones de reservas.
* **Registrarse**: En esta opción se le permite a la persona crear un nuevo usuario con sus datos, se le pedirá que ingrese no solamente un mail y contraseña, sino también datos más específicos como el nombre, DNI, fecha de nacimiento, etc. Un dato importante es que a la hora de ingresar el mail, la aplicación va a realizar una verificación del archivo de usuarios para comprobar que el mail ingresado no se encuentre utilizado por otra persona, si se encuentra utilizado le va a informar a la persona mediante una excepción que el mail que desea ingresar ya se encuentra utilizado por alguien más. Si el mail no está en uso, la aplicación enviará todos los datos del nuevo usuario a una colección que contiene todos los usuarios, para luego guardar esa misma colección en el archivo de usuarios.
* **Salir**: Esta opción dará lugar al cierre del programa, el programa se mantendrá abierto mientras un atributo sea verdadero, la opción de “salir” cambiara el estado del atributo a falso y terminará de manera normal la ejecución del programa.

Una vez iniciada la sesión, se accede al menú principal de la aplicación, en donde encontramos opciones variadas:

* **Buscar vuelos**: La opción de buscar vuelos ingresará a un submenú en el cual tendremos distintas opciones para buscar un vuelo. El usuario deberá ingresar un origen y un destino para ver los vuelos disponibles en ese recorrido. Se podrá buscar un vuelo por:
  + Precio: Se le pedirá al usuario que ingrese un precio mínimo y un precio máximo, y se filtraran vuelos disponibles en ese rango de precio.
  + Clase: De las 3 clases disponibles (Económica, ejecutiva y primera clase), el usuario elegirá el nombre de la clase que desea ver vuelos, y los vuelos se filtrarán según la clase que elija.
  + Aerolínea: El usuario podrá elegir una aerolínea de las que se encuentren registradas en la página, y se filtraran vuelos únicamente de esa aerolínea.
  + ID: Se pedirá al usuario que ingrese un ID (o número de vuelo) para filtrar un solo vuelo.
* **Buscar alojamientos**: Al igual que en la búsqueda de vuelos, primero se filtrarán los alojamientos según la región que el usuario elija. Luego, se ingresará a otro submenú en el cual tendremos diversas opciones de búsqueda:
  + Precio: Se le pedirá al usuario que ingrese un precio máximo, y se filtraran vuelos disponibles a partir de precio.
  + Capacidad: El usuario deberá ingresar el mínimo de personas que se pueden alojar en la casa, y los resultados que aparecerán serán con una capacidad mínima de lo que el usuario ingresó.
* **Ver reservas**: Esta opción mostrará todas las reservas del usuario. Si el usuario tiene reservas, primero las mostrará, y luego le preguntará si quiere cancelar alguna. Luego de eso, el programa le volverá a preguntar al usuario, esta vez, si desea ver las reservas ya canceladas anteriormente.
* **Salir**: Funciona igual que el método salir del menú principal, a diferencia de que en este no se cerrará el programa, sino que volverá a dicho menú principal.

**MATRIZ DE SOLUCIONES Y DIARIO DE TRABAJO**

|  |  |  |  |
| --- | --- | --- | --- |
| FECHA | PERSONA | PROBLEMA | SOLUCION |
| 10/06/2023 | ABRIL | Problemas al identificar la clave del mapa de usuarios, principal idea utilizar contraseña como clave. | Poner como clave del mapa de usuarios un email, ya que el email es un elemento único que no se puede repetir. |
| 13/06/2023 | NAHUEL | Problemas en la clase reserva al querer leer un JSONObject en un atributo generico | Realizar una verificación del tipo de dato que le llega al atributo genérico y hacer una función para cada tipo de dato |
| 19/06/2023 | NAHUEL | Problemas con la API de países, la API dejo de funcionar ya que la pagina aparece caída | Buscar otra API y utilizar únicamente los atributos que iban a ser utilizados con la API anterior |
| 19/06/2023 | MATEO | Problemas para pasar del archivo de alojamientos a la colección de alojamientos | El error estaba al cargar el nombre de la clave del JSONObject |

**FUENTES DE INFORMACION CONSULTADAS**

* **Chat GPT**
* **Diseño UML**

https://lucid.app/lucidspark/3b10567c-cf81-43d5-a808-59f34336d2ee/edit?viewport\_loc=-4524%2C-860%2C10037%2C4951%2C0\_0&invitationId=inv\_7eab2a71-584b-4c49-b429-76c9c496db4b

* **API de países**

https://restcountries.com/v2/all

* **Información recolectada de la catedra del campus de UTN.**

https://campus.mdp.utn.edu.ar