

VERIFICACIÓN Y VALIDACIÓN

Link GIT: https://gitlab.etsisi.upm.es/bt0051/cap_iwsim22-09

Link Redmine: <https://fis.etsisi.upm.es/projects/iwsim22-09-cuidando-a-pancho/wiki>

Líder de requisitos: Rafael Stefanescu Mihoc

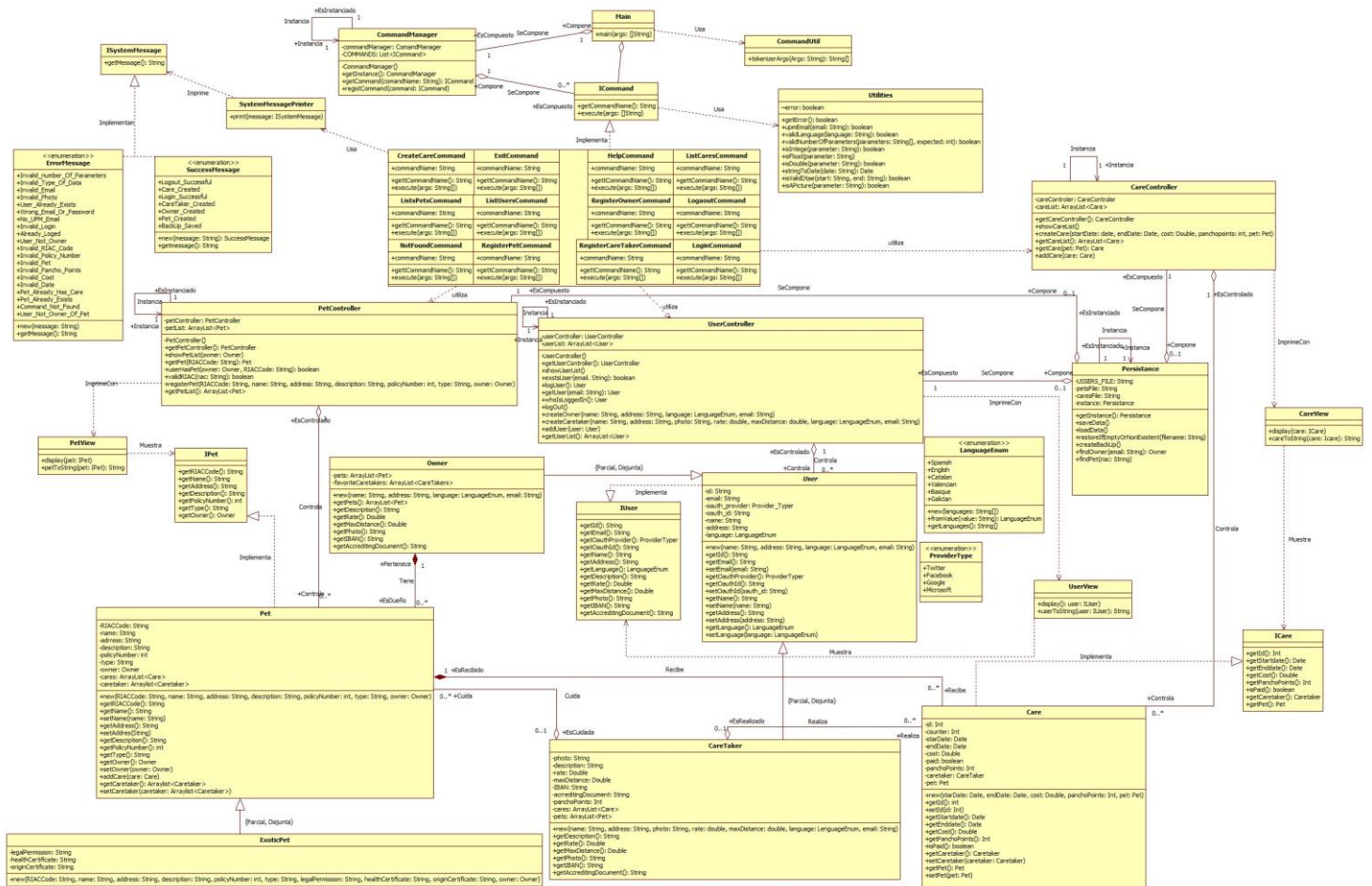
Líder de análisis: Moisés Fernández Fernández

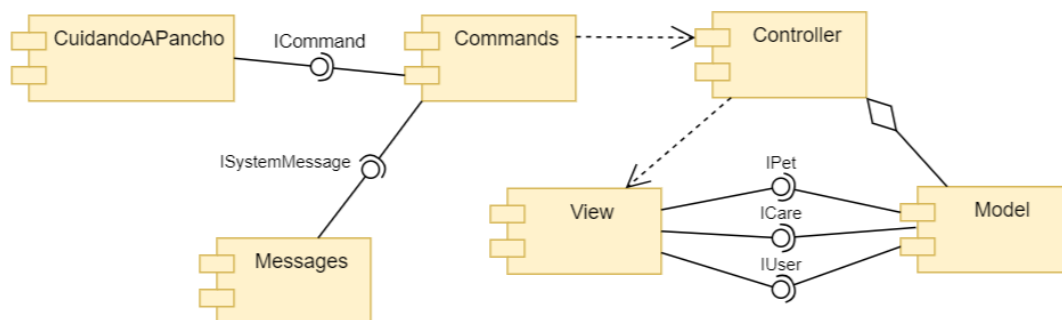
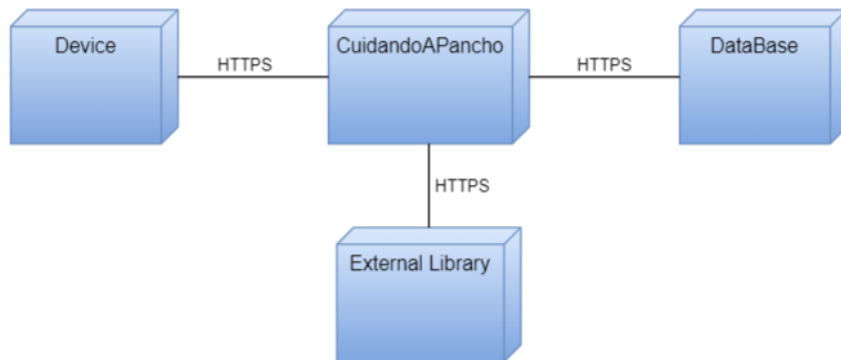
Líder de diseño: Jinsong Chen

Líder de Implementación: David Gálvez Redondo

Líder de Verificación y Validación: Eduardo Enrique Montiel Rios

Diagramas



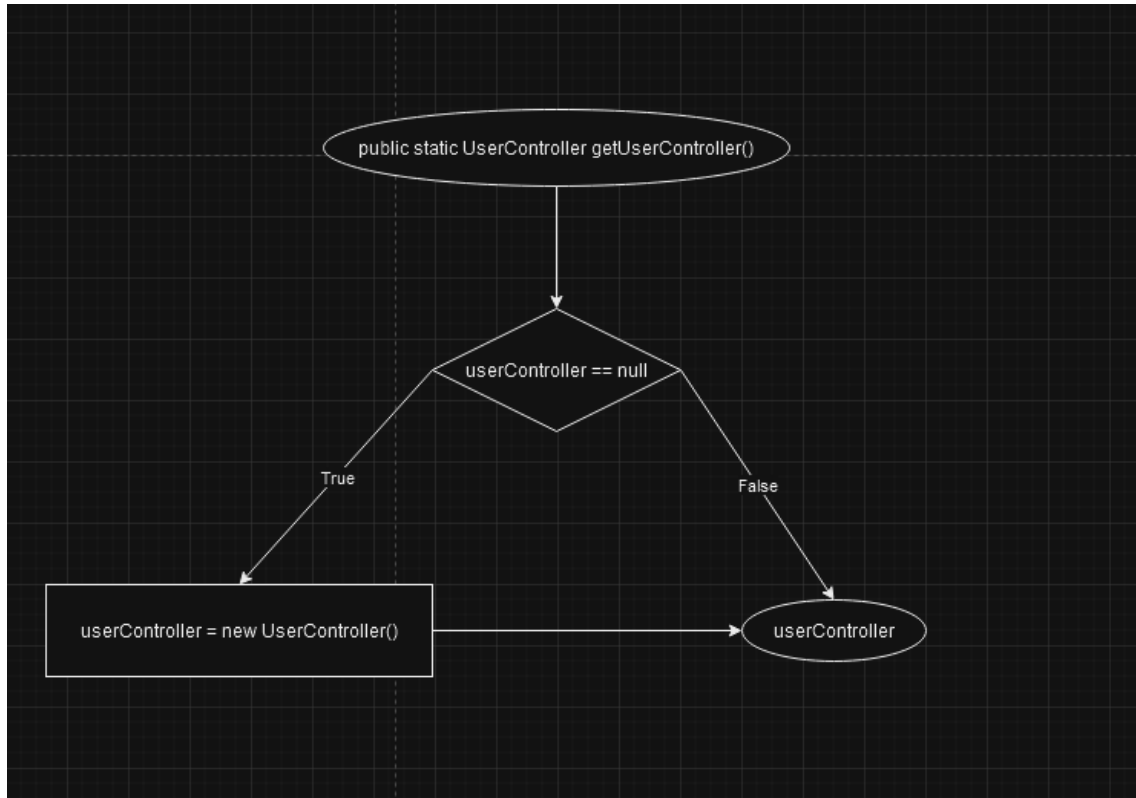


Pruebas Unitarias

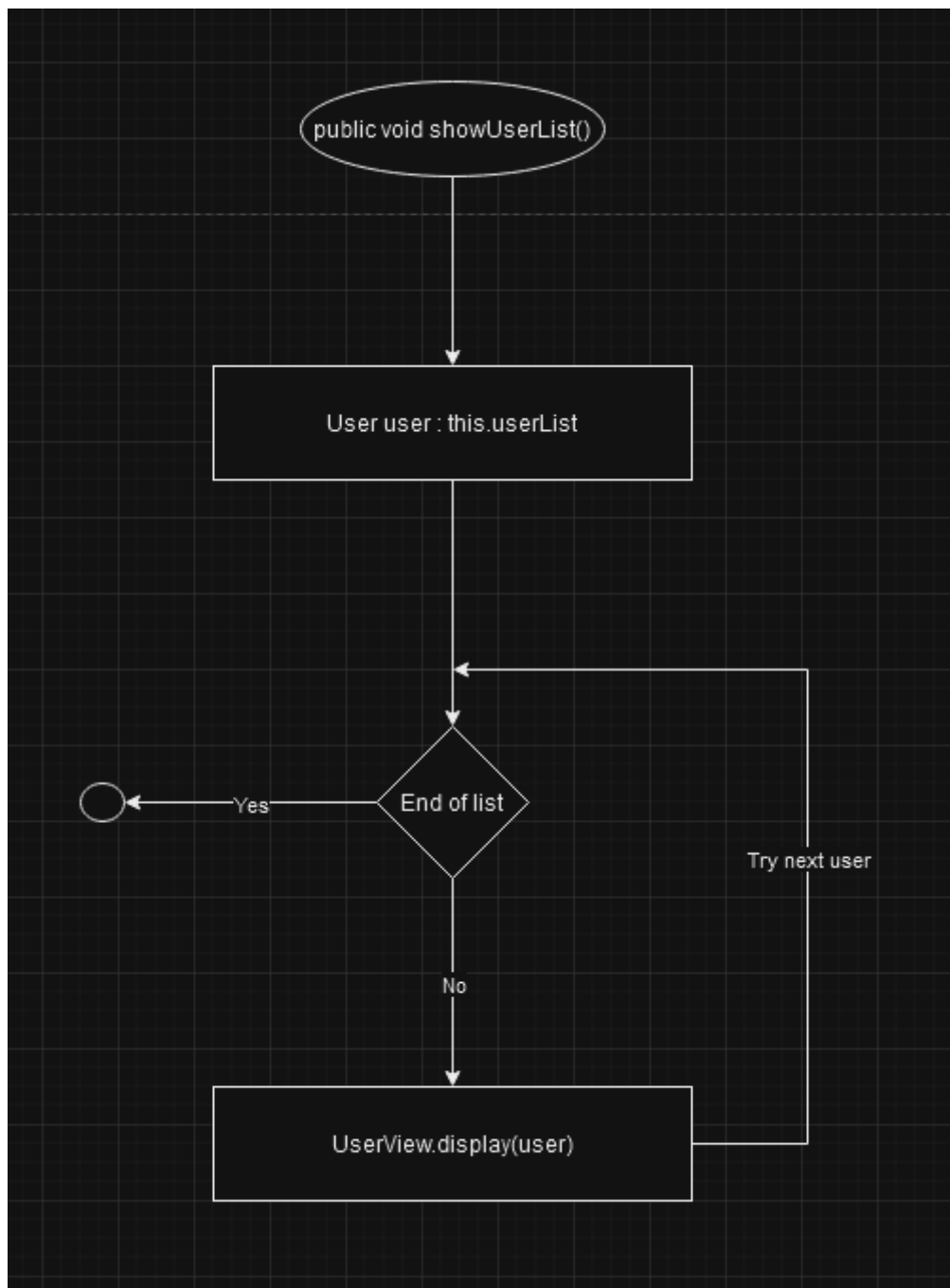
caja negra y caja blanca

Creación de nuevos clientes en el sistema

public static UserController getUserController()

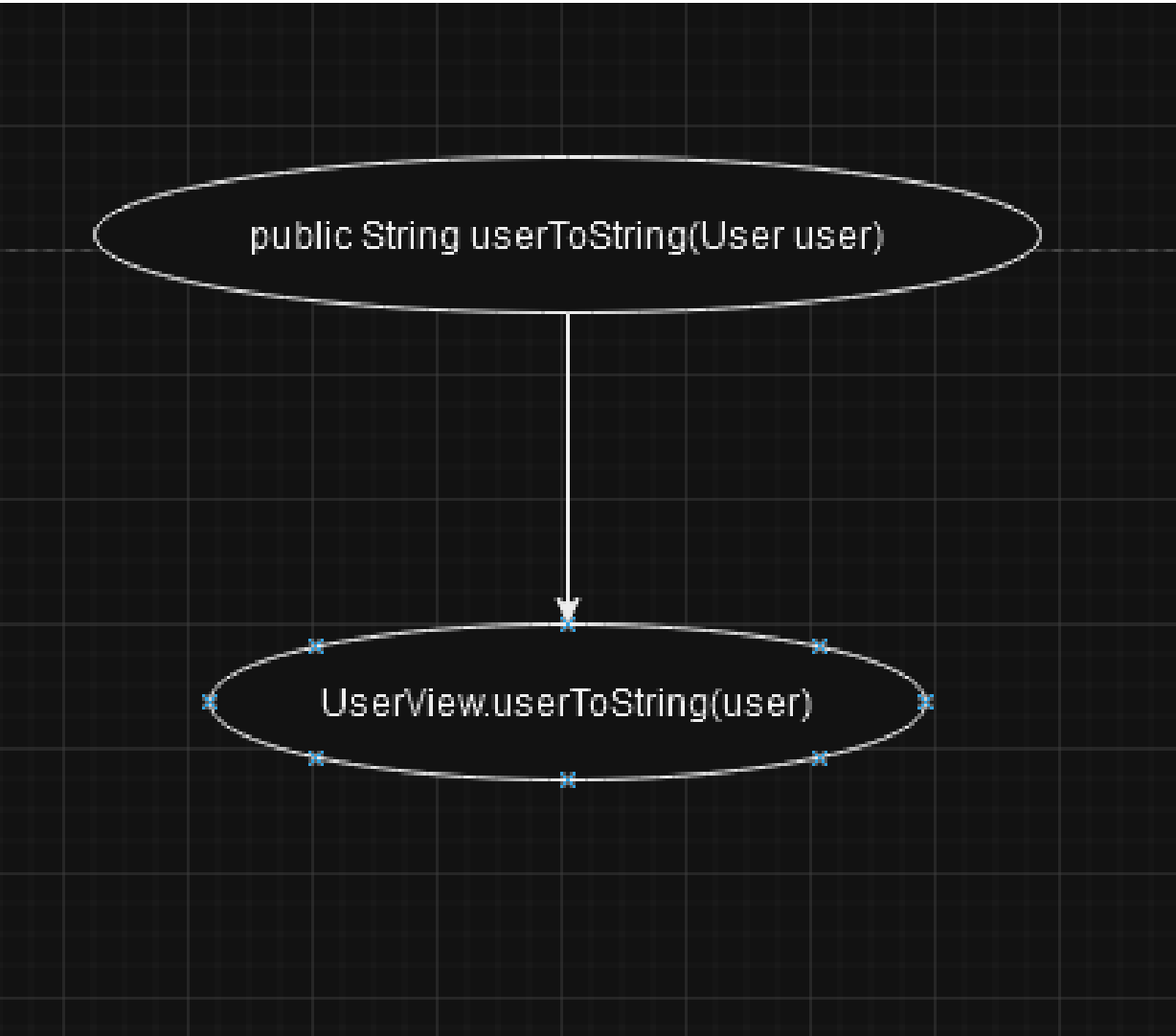


public void showUserList()



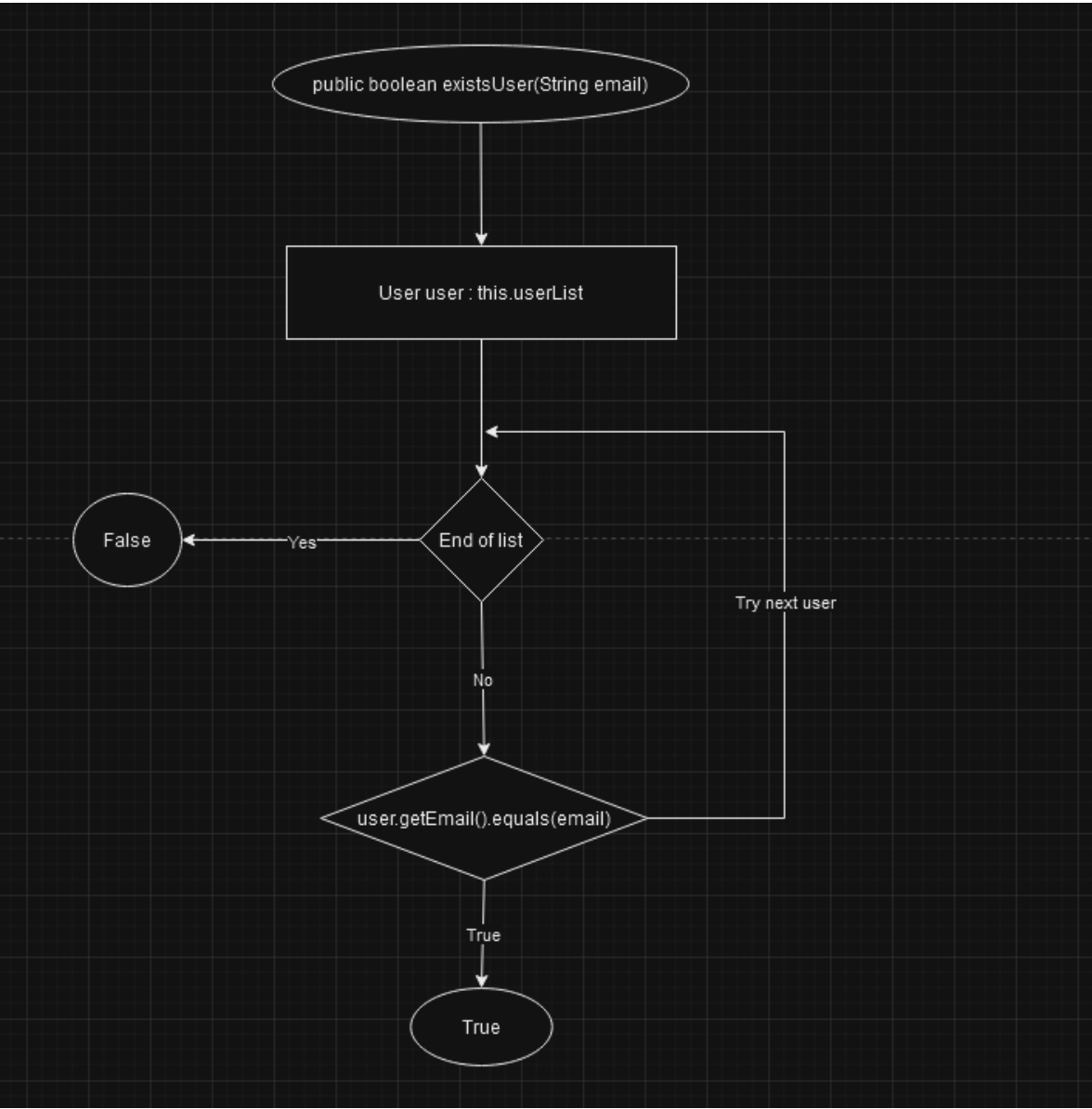
public String userToString(User user)

Tipo	Entrada	Valores válidos	Valores no válidos
Condición Booleano	User user	V1. User user	N1. Cualquier otro valor

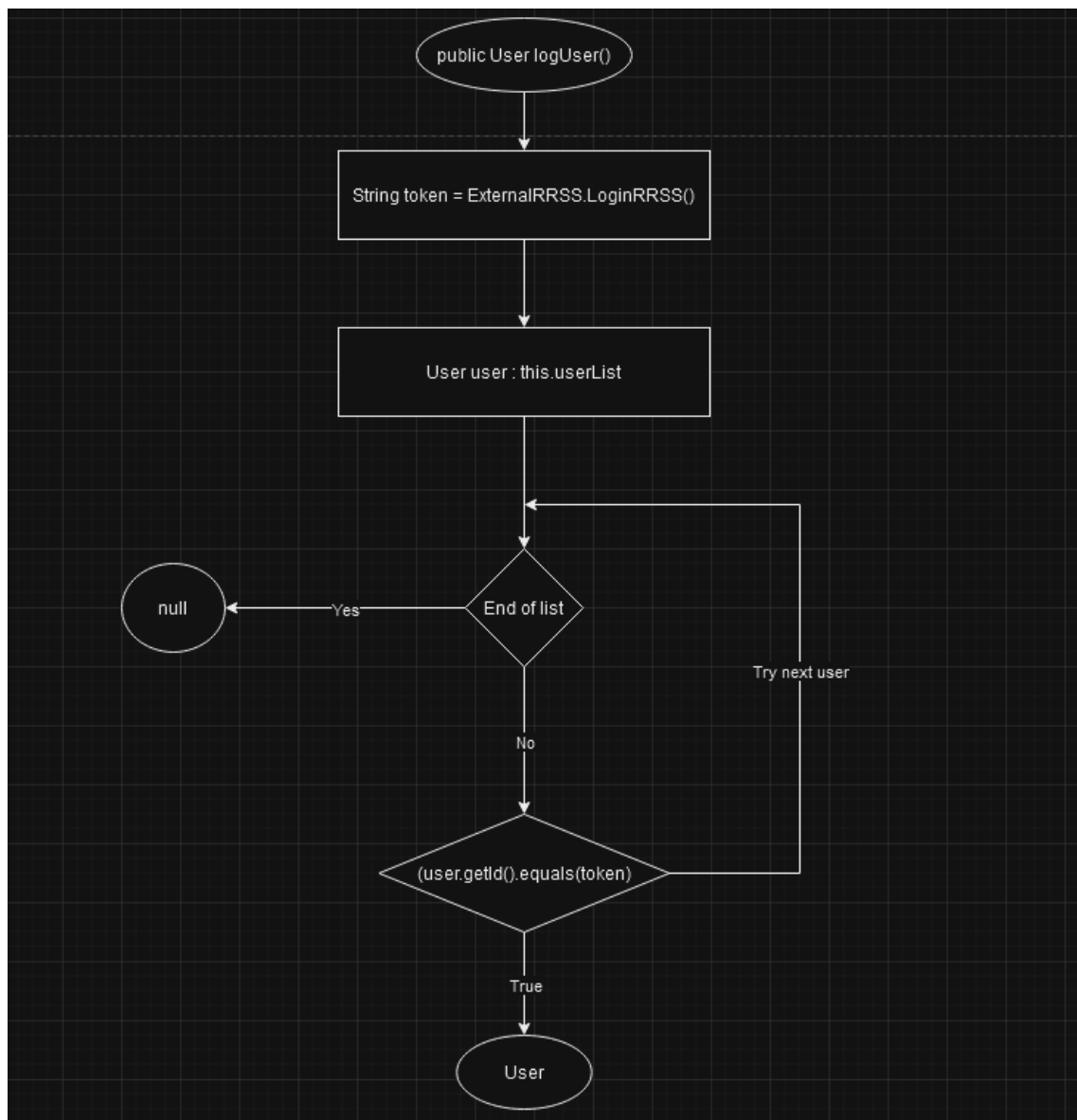


public boolean existsUser(String email)

Tipo	Entrada	Valores válidos	Valores no válidos
Condición Booleano	String email	V1. jin@upm.es	N1. jin@mm.es

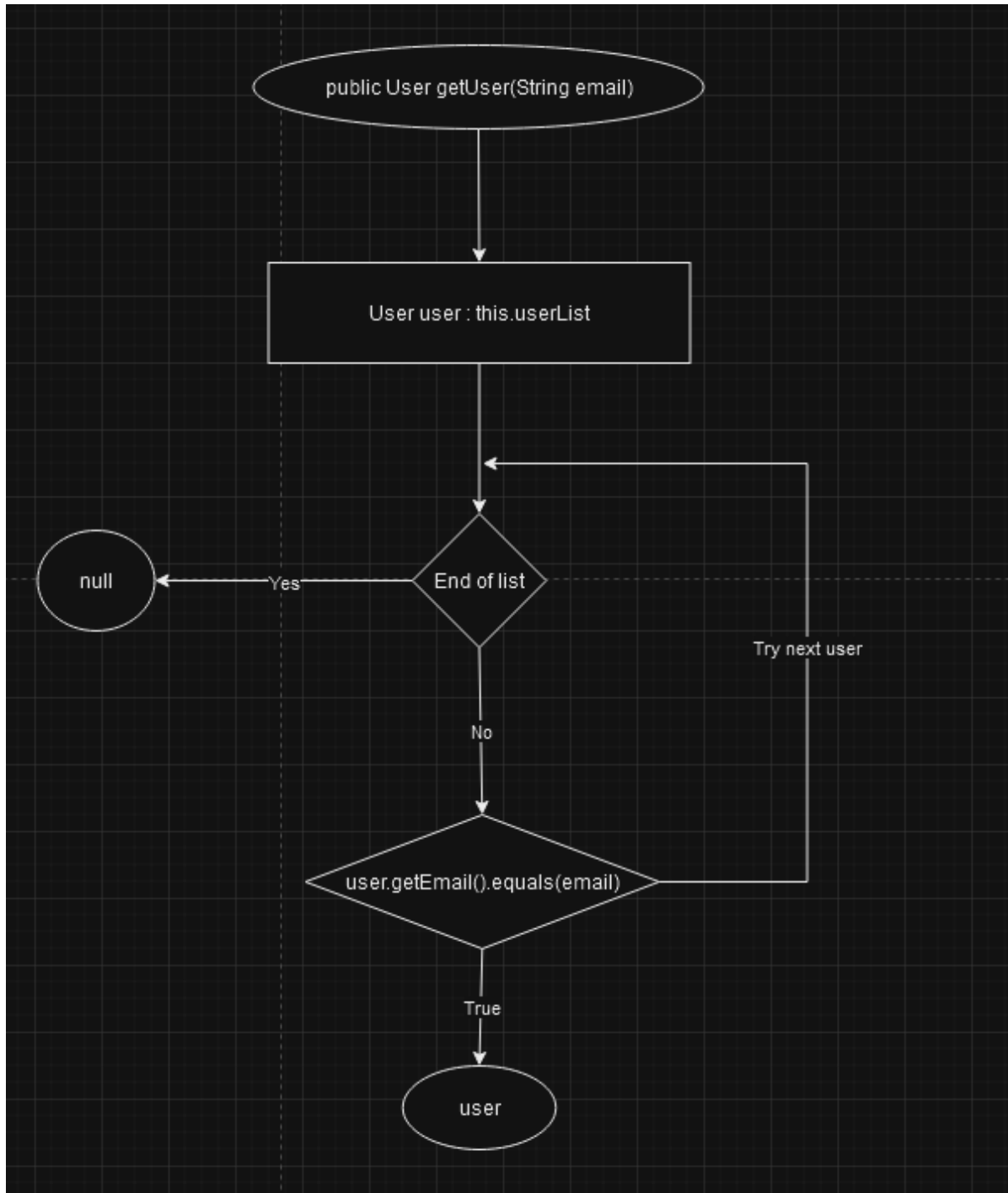


public User logUser()

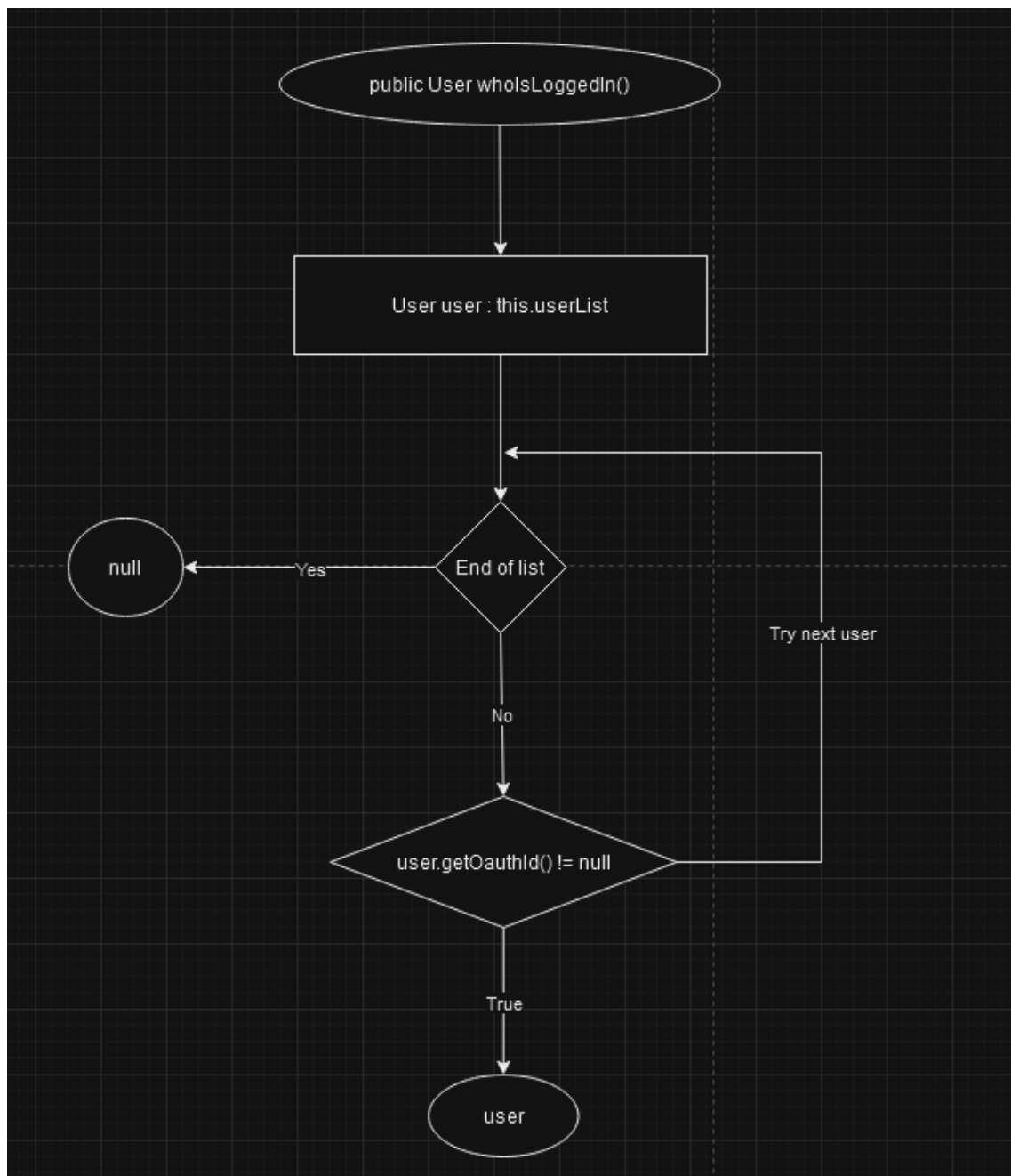


public User getUser(String email)

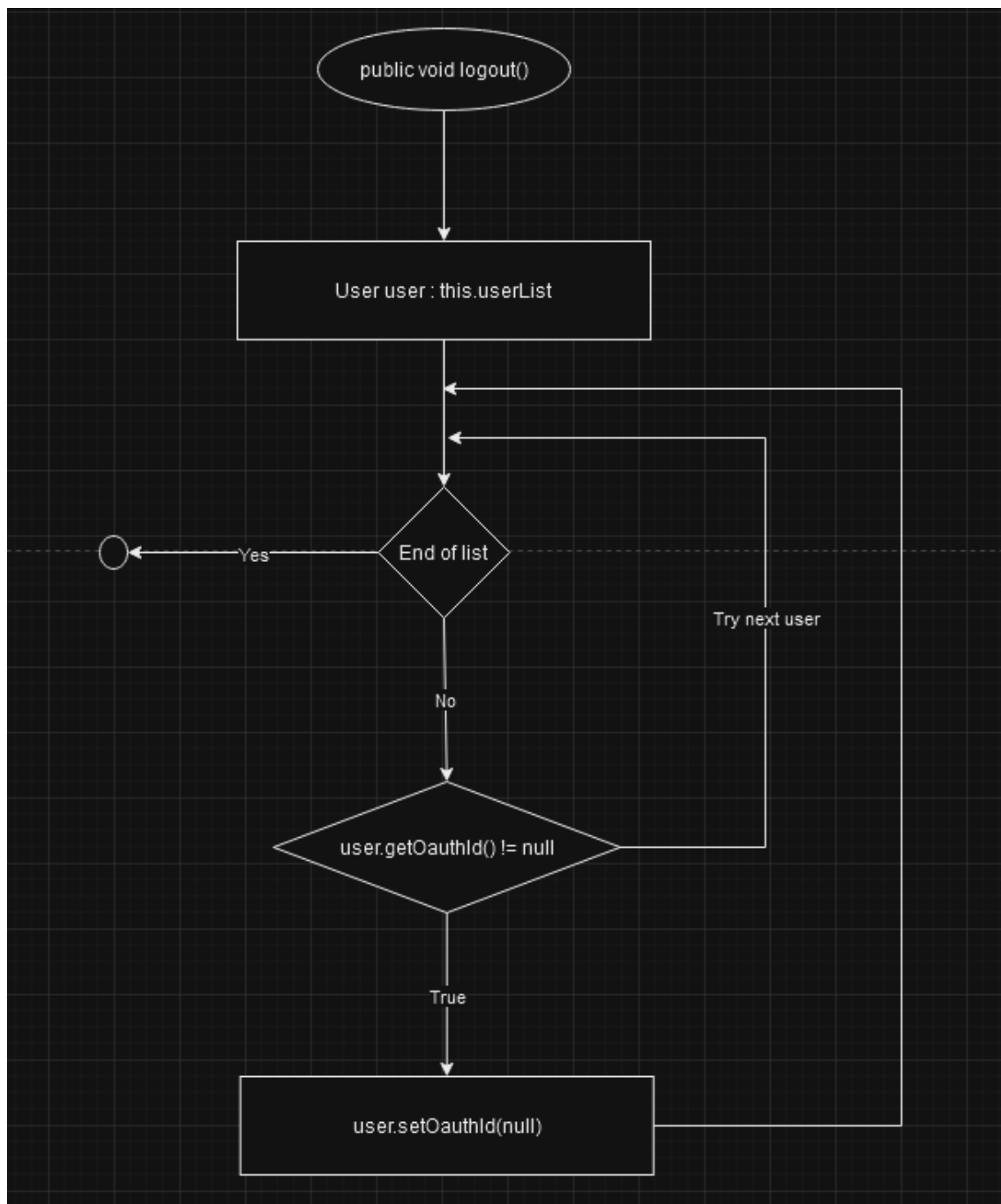
Tipo	Entrada	Valores válidos	Valores no válidos
Condición Booleano	String email	V1. rafa@upm.es	N1. rafa@mm.es



public User whoIsLoggedIn()

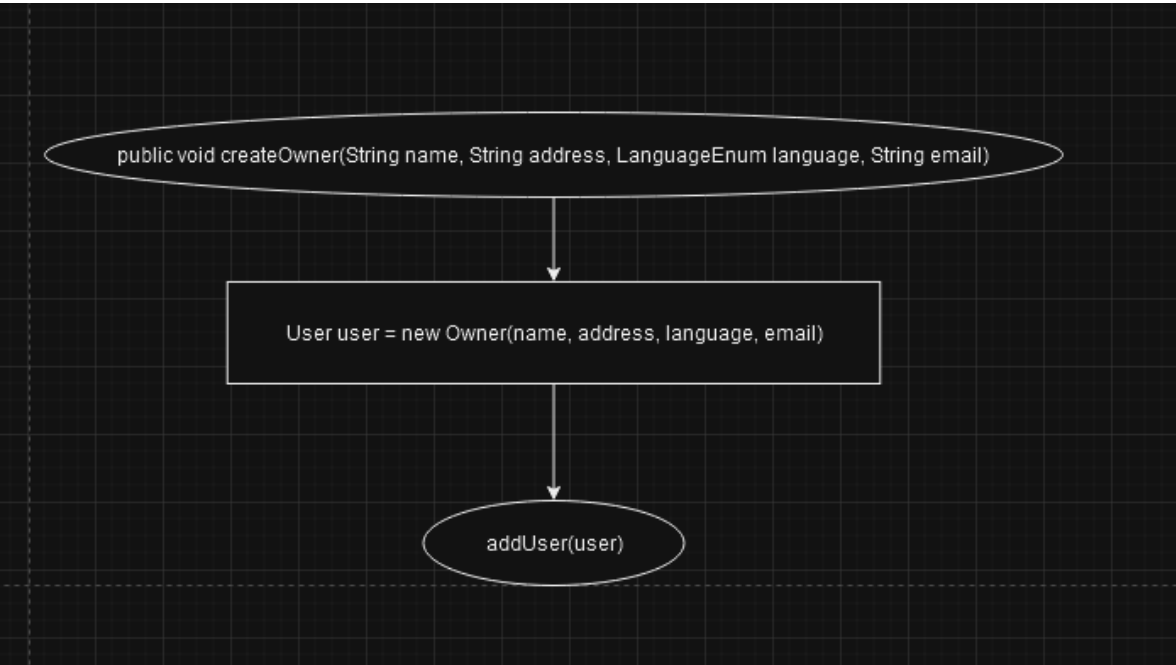


public void logout()



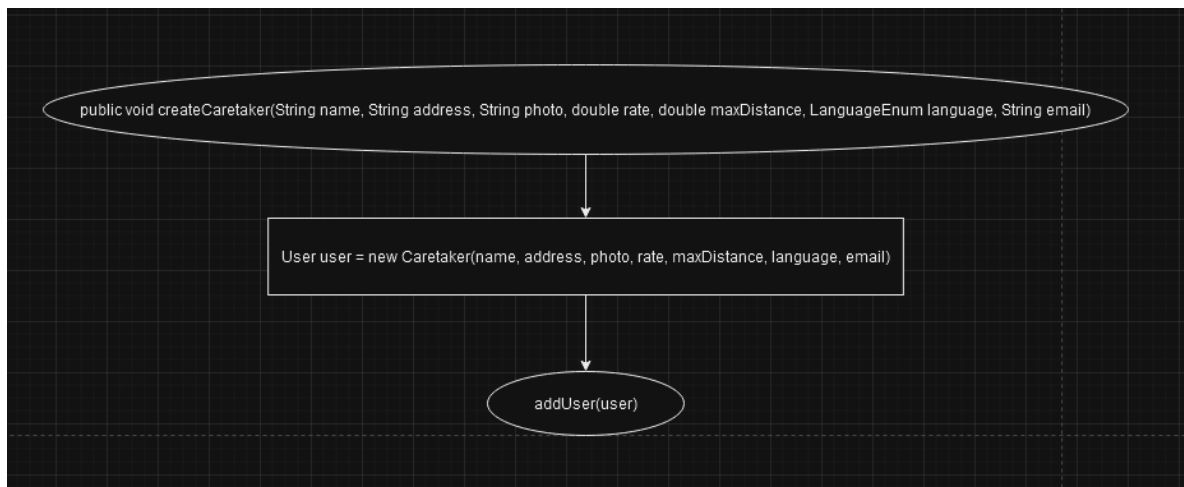
public void createOwner(String name, String address, LanguageEnum language, String email)

Tipo	Entrada	Valores válidos	Valores no válidos
Condición Booleano	String name	V1. jin	N1. 23
Condición Booleano	String address	V2. Calle pepito	N2. 53
Conjunto de valores	LanguageEnum language	V3. es V4. spanish V5. SPANISH V6. ES	N3. español
Condición Booleano	String email	V1. moi@upm.es	N1. moi@mm.es



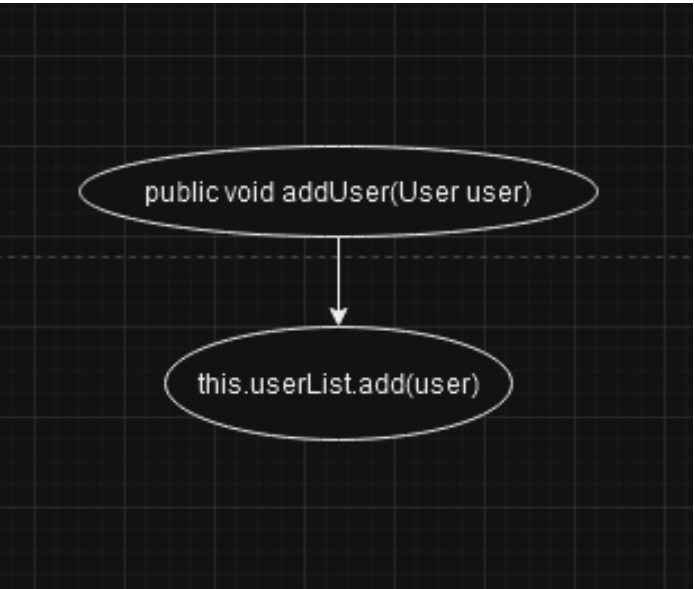
public void createCaretaker(String name, String address, String photo, double rate, double maxDistance, LanguageEnum language, String email)

Tipo	Entrada	Valores válidos	Valores no válidos
Condición Booleano	String name	V1. jin	N1. 23
Condición Booleano	String address	V2. Calle pepito	N2. 53
Condición Booleano	String photo	V3. Foto.jpg	N3. foto
Condición Booleano	Double rate	V4. 22.2	N4. 2
Condición Booleano	Double maxDistance	V5. 23.3	N5. 22
Conjunto de valores	LanguageEnum language	V6. es V7. spanish V8. SPANISH V9. ES	N6. español
Condición Booleano	String email	V1. david@upm.es	N1. david@mm.es

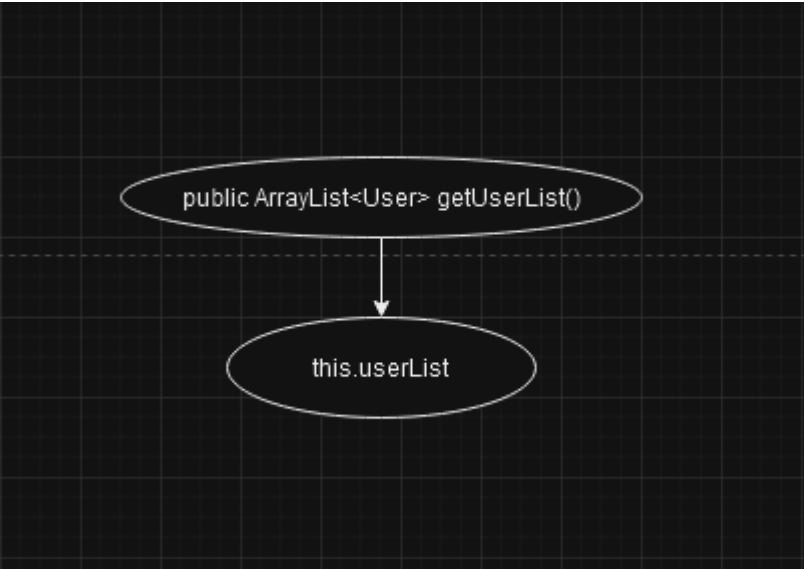


public void addUser(User user)

Tipo	Entrada	Valores válidos	Valores no válidos
Condición Booleano	User user	V1. User user	N1. Cualquier otro valor



public ArrayList<User> getUserList()



Pruebas de aceptación

Pruebas de aceptación para la creación de clientes:

```
17
18 class UserControllerTest {
19
```

La clase UserController se encarga de la creación de los usuarios, aquí están sus pruebas ejecutadas y pasadas:

```
39 @Test
40 public void testSingleton() {
41     UserController instance1 = UserController.getUserController();
42     UserController instance2 = UserController.getUserController();
43     assertEquals(instance1, instance2);
44 }
```

```

7 @Test
void showUserList() {
    Owner user = new Owner(name: "Edu", address: "Bravo Murillo", LanguageEnum.fromValue("es"), email: "edu@upm.es");
    userController.addUser(user);
}
```

```

52 // Test con un user correcto
53 @Test
54 void existsUser() {
55     Owner user = new Owner(name: "Edu", address: "Bravo Murillo", LanguageEnum.fromValue("ES"), email: "edu@upm.es");
56     userController.addUser(user);
57     assertTrue(userController.existsUser(email: "edu@upm.es"));
58 }
```

```

// Test con un user incorrecto
@Test
void existsUser2() {
    Owner user = new Owner(name: "Edu", address: "Bravo Murillo", LanguageEnum.fromValue("ES"), email: "edu@upm.es");
    userController.addUser(user);
    assertFalse(userController.existsUser(email: "edu2@upm.es"));
}
```

```

//Test correcto al meter bien el correo
@Test
void logUser() {
    Owner user = new Owner(name: "Moi", address: "Bravo Murillo 123", LanguageEnum.fromValue("es"), email: "moi@upm.es");
    userController.addUser(user);

    User loggedInUser = userController.logUser();

    assertNotNull(loggedInUser);
    assertEquals(user, loggedInUser);
}
```

```

//Test que devuelva null al introducir mal el correo
@Test  ± moisitx
void logUser2() {

    Owner user = new Owner( name: "Moi", address: "Bravo Murillo 123", LanguageEnum.fromValue("es"), email: "moi@upm.es");
    userController.addUser(user);

    User loggedInUser = userController.logUser();

    assertNull(loggedInUser);
    assertNotEquals(user, loggedInUser);

}

```

```

//Pilla el user correctamente
@Test  ± moisitx
void getUser() {

    Owner user = new Owner( name: "Edu", address: "Bravo Murillo", LanguageEnum.fromValue("ES"), email: "edu@upm.es");
    userController.addUser(user);
    User XUser = userController.getUser( email: "edu@upm.es");
    assertNotNull(XUser);
    assertEquals(user, XUser);

}

```

```

//Pilla el user incorrectamente y es igual a null
@Test  ± bs0457 <https://gitlab.etsisi.upm.es/bs0457/airupm>
void getUser2() {

    Owner user = new Owner( name: "Edu", address: "Bravo Murillo", LanguageEnum.fromValue("ES"), email: "edu@upm.es");
    userController.addUser(user);
    User XUser = userController.getUser( email: "edu2@upm.es");
    assertNull(XUser);

}

```

```

//usuario logueado correctamente
@Test  ± moisitx
void whoIsLoggedIn() {

    Owner user = new Owner( name: "Edu", address: "Bravo Murillo", LanguageEnum.fromValue("es"), email: "edu@upm.es");
    Owner user2 = new Owner( name: "jin", address: "Calle pepito", LanguageEnum.fromValue("es"), email: "jin.chen@upm.es");
    user2.setOAuthId("oauth-id");
    userController.addUser(user);
    userController.addUser(user2);
    assertEquals(user2, userController.whoIsLoggedIn());

}

```

```

//usuario no logeado
@Test  ± bs0457 <https://gitlab.etsisi.upm.es/bs0457/airupm>
void whoIsLoggedIn2() {

    Owner user = new Owner( name: "Edu", address: "Bravo Murillo", LanguageEnum.fromValue("es"), email: "edu@upm.es");
    Owner user2 = new Owner( name: "Jin", address: "Calle pepito", LanguageEnum.fromValue("es"), email: "jin.chen@upm.es");
    userController.addUser(user);
    userController.addUser(user2);

    User loggedInUser = userController.whoIsLoggedIn();
    assertNull(loggedInUser);

}

```

```

@Test  ± moisitx
void logout() {

    Owner user = new Owner( name: "Edu", address: "Bravo Murillo", LanguageEnum.fromValue("es"), email: "edu@upm.es");
    Owner user2 = new Owner( name: "jin", address: "Calle pepito", LanguageEnum.fromValue("es"), email: "jin.chen@upm.es");
    user2.setOAuthId("oauth-id");
    userController.addUser(user);
    userController.addUser(user2);
    userController.logout();
    assertNull(user2.getOAuthId());

}

```



```

27 //Creado owner correctamente
28 @Test @moisltx
29 void createOwner() {
30     userController.createOwner( name: "Rafa", address: "Bravo Murillo 22", LanguageEnum.fromValue("es"), email: "rafa@upm.es");
31     assertTrue(userController.existsUser( email: "rafa@upm.es"));
32 }

```

```

33 //Owner no creado
34 @Test @bs0457 <https://gitlab.etsisi.upm.es/bs0457/ainrpm>
35 void createOwner2() {
36     userController.createOwner( name: "Rafa", address: "Bravo Murillo 22", LanguageEnum.fromValue("es"), email: "rafa@mm.es");
37     assertFalse(userController.existsUser( email: "rafa2@mm.es"));
38 }

```

```

39 // Caretaker correcto
40 @Test @moisltx
41 void createCaretaker() {
42     userController.createCaretaker( name: "David", address: "Calle juan", photo: "2", rate: 12.2, maxDistance: 15.5, LanguageEnum.fromValue("ES"), email: "david@upm.es");
43     assertTrue(userController.existsUser( email: "david@upm.es"));
44 }

```

```

45 // Caretaker incorrecto
46 @Test @bs0457 <https://gitlab.etsisi.upm.es/bs0457/ainrpm>
47 void createCaretaker2() {
48     userController.createCaretaker( name: "David", address: "Calle juan", photo: "2", rate: 12.2, maxDistance: 15.5, LanguageEnum.fromValue("ES"), email: "david@mm.es");
49     assertFalse(userController.existsUser( email: "david2@mm.es"));
50 }

```

```

51 @Test @moisltx
52 public void testAddUser() {
53     Owner user = new Owner( name: "jin", address: "Calle pepito", LanguageEnum.fromValue("es"), email: "jin.chen@upm.es");
54     userController.addUser(user);
55     assertTrue(userController.getUserList().contains(user));
56 }

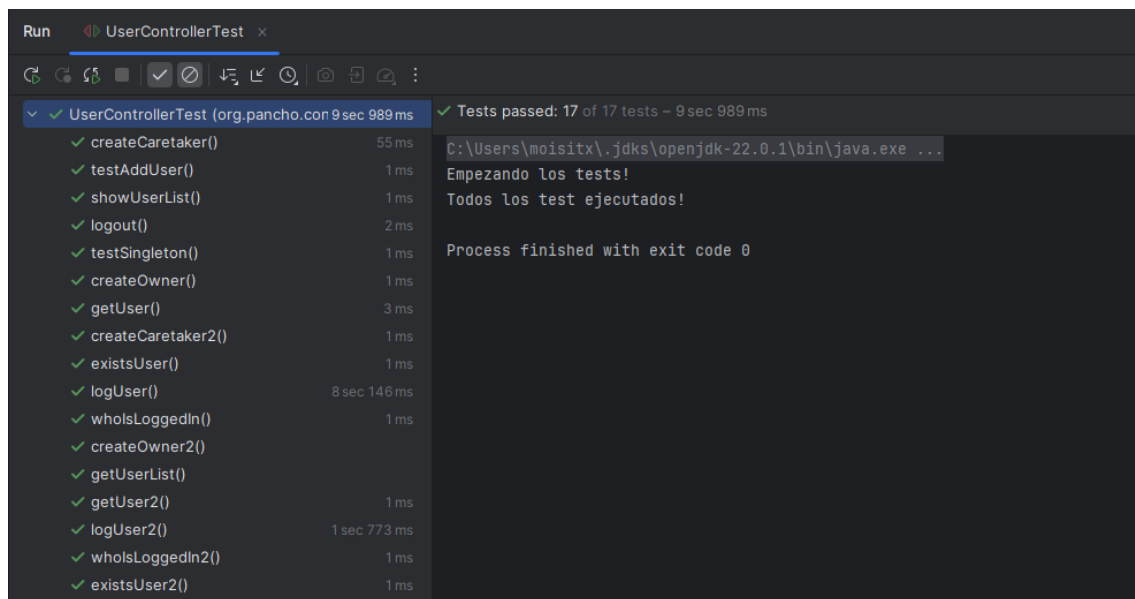
```

```

57 @Test @moisltx
58 void getUserList() {
59     Owner user = new Owner( name: "Edu", address: "Bravo Murillo", LanguageEnum.fromValue("es"), email: "edu@upm.es");
60     Owner user2 = new Owner( name: "jin", address: "Calle pepito", LanguageEnum.fromValue("es"), email: "jin.chen@upm.es");
61     Owner user3 = new Owner( name: "Rafa", address: "Bravo Murillo 22", LanguageEnum.fromValue("es"), email: "rafa@upm.es");
62     userController.addUser(user);
63     userController.addUser(user2);
64     userController.addUser(user3);
65     ArrayList<User> users = userController.getUserList();
66     assertEquals( expected: 3, users.size());
67     assertTrue(users.contains(user));
68     assertTrue(users.contains(user2));
69 }

```

Aquí están todas las pruebas pasadas:



The screenshot shows the 'Run' window of an IDE. The title bar indicates 'UserControllerTest' is running. The main area is divided into two panes. The left pane lists 17 test methods, each preceded by a green checkmark and followed by its execution time. The right pane shows the output of the test run, including the Java command used, the start of the tests, the completion of all tests, and the final exit code.

Test Method	Execution Time
createCaretaker()	55 ms
testAddUser()	1 ms
showUserList()	1 ms
logout()	2 ms
testSingleton()	1 ms
createOwner()	1 ms
getUser()	3 ms
createCaretaker2()	1 ms
existsUser()	1 ms
logUser()	8 sec 146 ms
whosLoggedIn()	1 ms
createOwner2()	
getUserList()	
getUser2()	1 ms
logUser2()	1 sec 773 ms
whosLoggedIn2()	1 ms
existsUser2()	1 ms

Tests passed: 17 of 17 tests – 9 sec 989 ms

```
C:\Users\moisitx\.jdk\openjdk-22.0.1\bin\java.exe ...  
Empezando los tests!  
Todos los test ejecutados!  
Process finished with exit code 0
```

Se puede ver que todas las pruebas se ejecutan correctamente por lo que la creación de un cliente sería posible.

Trazabilidad

Descripción del requisito del registro de usuarios en Redmine.

Requisito_Funcional #14818

Modificar

Tempo dedicado

Monitorizar

Copiar

Borrar

Crear nueva cuenta con google

Añadido por **Fernández Fernández Holofo** hace 3 meses. Actualizado hace 3 meses.

Estado: **Esbozando**

Prioridad: **Alta**

Asignado a: **-**

Fecha de inicio: 2024-02-16

Fecha fin: 2026-02-16 (Finaliza en alrededor de 21 meses)

Descripción

Descripción: Registro de un usuario en la plataforma mediante google

Entidades: Nombre, Correo electrónico, Contraseña, Edad, Dígito entre dueño y cuidador

Procesamiento: Se crea al usuario utilizando las entidades

Salida: No genera salidas

Subtareas

Añadir

Peticiones relacionadas

Añadir

Historia: **Propuesta de cambios**

Actualizado por **Fernández Fernández Holofo** hace 3 meses

Asunto cambiado de Crear nueva cuenta para ser cuidador a Crear nueva cuenta con google

Se actualizó Descripción (dlt)

R1

Reportar en **Issue** | PDF

Modificar

Tempo dedicado

Monitorizar

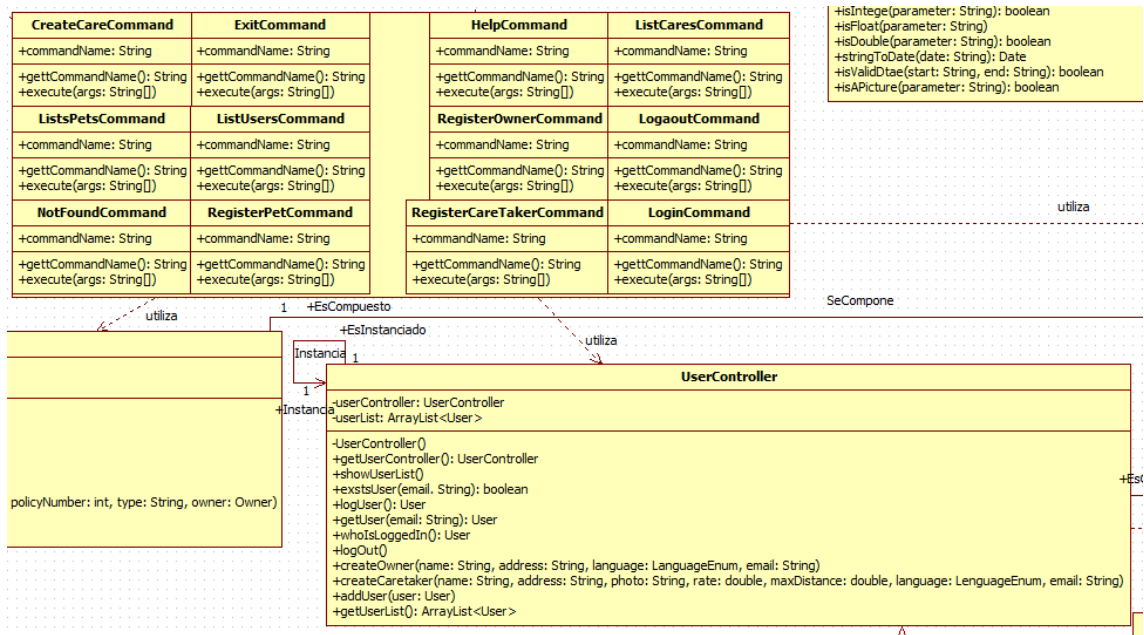
Copiar

Borrar

Para verificar este requisito de Redmine hemos hecho este diagrama de casos de uso



Posteriormente hemos realizado en base a este requisito las siguientes clases Usuario, dueño y cuidador y relaciones del diagrama de análisis.



A partir del diagrama de diseño realizamos la siguiente implementación en IntelIJ.

```

14 public class RegisterCaretakerCommand implements ICommand { //package <module>
15     public static final String COMMAND_NAME = "register-caretaker";
16
17     @Override //module
18     public String getCommandName() { return COMMAND_NAME; }
19
20     @Override //module
21     public void execute(String[] args) { //name, address, photo, rate, maxDistance, language, email
22         UserController userController = UserController.getUserController();
23
24         if (Utilities.validNumberOfParameters(args, expected: 7)) {
25             if (Utilities.isValidLanguage(args[5])) {
26                 if (Utilities.upmEmail(args[0])) {
27                     if (Utilities.isAPicture(args[2])) {
28                         if (Utilities.isDouble(args[3]) && Utilities.isDouble(args[4])) {
29                             if (!userController.existsUser(args[0])) {
30                                 userController.createCaretaker(args[0], args[1], args[2], Double.parseDouble(args[3]), Double.parseDouble(args[4]), LanguageEnum.fromValue(args[5]), args[6]);
31                                 SystemMessagePrinter.print(SuccessMessage.CARETAKER_CREATED);
32                             } else {
33                                 SystemMessagePrinter.print(ErrorMessage.USER_ALREADY_EXISTS);
34                             }
35                         } else {
36                             SystemMessagePrinter.print(ErrorMessage.INVALID_TYPE_OF_DATA);
37                         }
38                     } else {
39                         SystemMessagePrinter.print(ErrorMessage.INVALID_PHOTO);
40                     }
41                 } else {
42                     SystemMessagePrinter.print(ErrorMessage.NO_UPM_EMAIL);
43                 }
44             } else {
45                 SystemMessagePrinter.print(ErrorMessage.INVALID_LANGUAGE);
46             }
47         } else {
48             SystemMessagePrinter.print(ErrorMessage.INVALID_NUMBER_OF_PARAMETERS);
49         }
50     }
51 }

```

```

14 public class RegisterOwnerCommand implements ICommand { 1 usage 1 moisitx *
15     public static final String COMMAND_NAME = "register-owner";
16
17     @Override 1 moisitx *
18     public String getCommandName() { return COMMAND_NAME; }
19
20     @Override 1 moisitx *
21     public void execute(String[] args) { //name, address, language, email
22         UserController userController = UserController.getUserController();
23
24         if (Utilities.validNumberOfParameters(args, expected: 4)){
25             if (Utilities.validLanguage(args[2])) {
26                 if(Utilities.upmEmail(args[3])) {
27                     if(!userController.existsUser(args[3])) {
28                         userController.createOwner(args[0], args[1], LanguageEnum.fromValue(args[2]), args[3]);
29                         SystemMessagePrinter.print(SuccessMessage.OWNER_CREATED);
30                     } else {
31                         SystemMessagePrinter.print(ErrorMessage.USER_ALREADY_EXISTS);
32                     }
33                 } else {
34                     SystemMessagePrinter.print(ErrorMessage.NO_UPM_EMAIL);
35                 }
36             } else {
37                 SystemMessagePrinter.print(ErrorMessage.INVALID_LANGUAGE);
38             }
39         } else {
40             SystemMessagePrinter.print(ErrorMessage.INVALID_NUMBER_OF_PARAMETERS);
41         }
42     }
43 }
44
45 }
46
47 }

```

```

public void createCaretaker(String name, String address, String photo, double rate, double maxDistance, LanguageEnum language, String email){ 4 usages 1 moisitx
    User user = new Caretaker(name, address, photo, rate, maxDistance, language, email);
    addUser(user);
}

```

```

public void createOwner(String name, String address, LanguageEnum language, String email) { 4 usages 1 moisitx
    User user = new Owner(name, address, language, email);
    addUser(user);
}

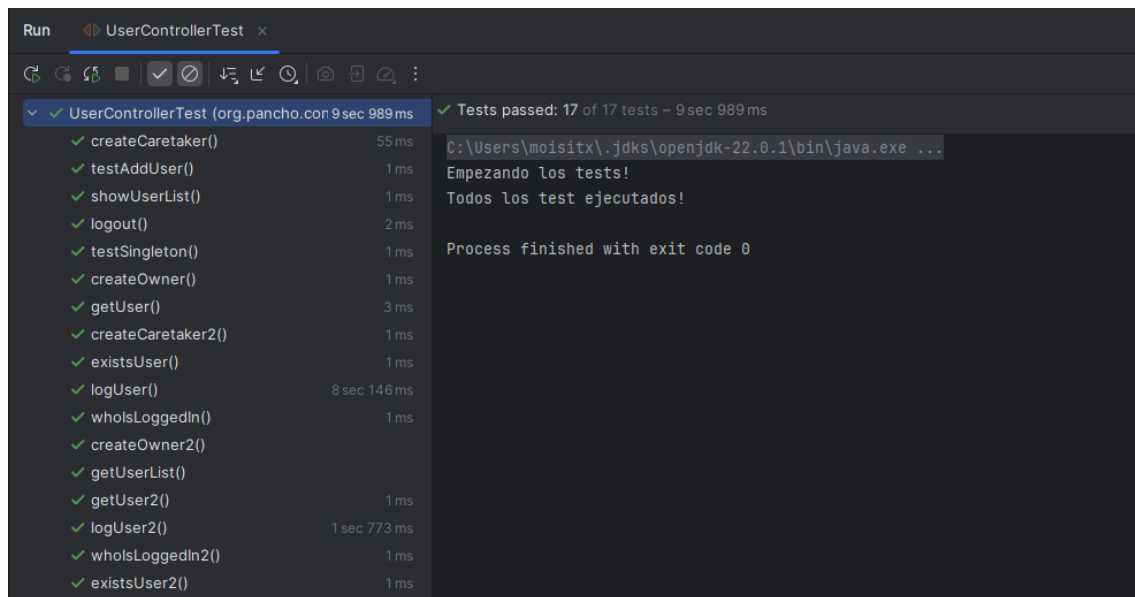
```

```

public void addUser(User user){ 19 usages 1 moisitx
    this.userList.add(user);
}

```

Una vez hecha la implementación hemos realizado las pruebas dando como resultados la siguiente imagen.



Todo esto verifica el requisito funcional de registrar usuario.