

UAB

Universitat Autònoma de Barcelona

ESCUELA DE INGENIERÍA - Curso 2019/2020

LABORATORIO INTEGRADO DE SOFTWARE

TEST: IMPLEMENTACIÓN Y RESULTADOS

Agustín Tamayo - 1492214
Arnau Sarabia - 1492826
Jan Tugores - 1494134
Judith Bellido - 1455984
Angel Sacristán Ruiz - 1457243
Martin Susin Palacios - 1492417
Pau Borda - 1458601
Carles Milanés Horno - 1455091

Equipo docente:
Antonio Manuel López Peña
Lluís Gesa Bote
Jose Luis Gomez Zurita

7 de Junio de 2020



Índice

1	Introducción	2
2	Test Case	3
2.1	Test Case: Login	4
2.2	Test Case: Registro	5
2.2.1	Registro: Campos vacíos	5
2.2.2	Registro: Contraseñas coincidentes	7
2.2.3	Registro: Formato contraseña	8
2.2.4	Registro: Formato del email	9
2.3	Test Case: Validación de resultados	10
3	Revisión Técnica Formal (RTF)	11
4	Exploratory Testing	13
4.1	Exploratory: Creación de usuario	13
4.2	Exploratory: Login	14
4.3	Exploratory: Reconocimiento de voz	15
4.4	Exploratory: Geolocalización	17
4.5	Exploratory: Modificación del perfil	19
4.6	Exploratory: Logout	21
4.7	Exploratory: Creación de alarmas	22
4.8	Exploratory: Modificación de alarmas	24
4.9	Exploratory: Activación de alarmas	25
4.10	Exploratory: Transmisión de alertas	28
5	Back-End	29
5.1	Back-End: Login	29
5.1.1	Login: Login correcto	29
5.1.2	Login: Login con un usuario incorrecto	30
5.1.3	Login: Login con una contraseña incorrecta	31
5.2	Back-End: Cookies	32
5.3	Back-End: Crear alerta	33
5.4	Back-End: Activar alerta	34
5.5	Back-End: Transmitir geolocalización	35

1. Introducción

En este documento se presentan y explican los resultados de los test realizados en el Front-End y Back-End. Es importante recalcar que se han tratado y detallado los siguientes puntos:

- **Prueba unitaria (Test Case)**
- **RTF**
- **Exploratory Testing**
- **Back-End**

Los criterios y estrategia seguida para el desarrollo del *testing* se basa en la segmentación de código en funciones individuales para probar su correcto funcionamiento. La aptitud o rechazo de las pruebas se ha decidido en función del *output* obtenido. Si este coincide con la salida esperada, se determina un correcto funcionamiento y se procede con el siguiente test. Contrariamente, el caso negativo se documenta y comunica al equipo de desarrollo para arreglar las funcionalidades erróneas.

2. Test Case

En esta sección se verán Test Case del Login y Registro. Para ello, hemos utilizado una herramienta proporcionado por el mismo Visual Studio. En esta, se verá que se pueden definir las clases y métodos a testear mediante etiquetas como "[TestClass]" y "[TestMethod]" .

Para poder desarrollar correctamente los TestMethod, decidimos crear MockObjects y hacer que heredaran de la clase que queríamos probar. En la siguiente imagen, por ejemplo, se muestra el código realizado para simular la BBDD y poder centrar el *testing* íntegramente a las partes deseadas como: Login, Registro, entre otros. Mediante este método, conseguimos separar la dependencia de la funcionalidad "Login" con el comportamiento de la base de datos.

```
public sealed class SingletonBD
{
    private SingletonBD()
    {
        User.Add("judith@gmail.com", "judith");
        User.Add("agustin@tamayo.com", "agustin");
        User.Add("martinsusin@gmail.com", "martin");
        User.Add("dasdsadd", "judith");
        User.Add("dsdasdasd", "susin");
    }

    private static SingletonBD instance = null;
    private Hashtable User = new Hashtable();
    public static SingletonBD Instance
    {
        get
        {
            if (instance == null)
            {
                instance = new SingletonBD();
            }
            return instance;
        }
    }
    public void register(String user, String Password)
    {
        User.Add(user, Password);
    }
    public bool getlogin(String username, String password)
    {
        bool logueado = false;
        if (User.ContainsKey(username))
        {
            if (User[username] == password)
            {
                logueado = true;
            }
        }
        return logueado;
    }
}
```

Figura 1: Creación de una instancia Singleton de la BBDD.

2.1. Test Case: Login

El primer Test Case presentado se aplica al Login de la aplicación SHiFT. En la siguiente imagen, se ve pretende comprobar si se contrasta el *input* introducido con los datos de usuarios guardados en la BBDD. También, se quiere saber si ante un usuario existente o inexistente, el proceso de inicio de sesión responde con aceptación o rechazo, respectivamente.

```
[TestMethod]
public void Login()
{
    String[] _username = new String[5] { "judith@gmail.com", "agustin@tamayo.com", "martinsusin@gmail.com", "dasdsadd", "dsdasd" },
    _password = new String[5] { "judith", "agustin", "martin", "susin", "" };
    bool _login;
    bool[] _assert = new bool[5] { true, true, true, false, false };
    for(int i = 0; i < _username.Length; i++)
    {
        _login = _user.Login(_username[i], _password[i]);
        Assert.AreEqual(_login, _assert[i]);
    }
}
```

Figura 2: Test Method para verificar la correcta funcionalidad del Login.

2.2. Test Case: Registro

2.2.1. Registro: Campos vacíos

En el siguiente Test Case se ha querido validar si la función en cuestión coge adecuadamente los *inputs* introducidos (*Check Inputs*). Para ello, se ha desarrollado el siguiente código para automatizar el proceso.

```
[TestMethod]
public async Task CheckInputsEmptyTest()
{
    MockEditUserViewModel mockEditUserViewModel = new MockEditUserViewModel();

    UserModel user = new UserModel();

    user.Password = "";
    user.Email = "";
    user.FirstName = "";
    user.LastName = "";
    user.City = "";
    user.Birthday = new DateTime();
    user.Street = "";
    user.PC = "";
    user.Mobile = 0;

    Assert.IsFalse(mockEditUserViewModel.CheckInputs(user, ""));

    user.Email = "agustin.tamayo@e-campus.uab.cat";
    user.FirstName = "Agustin";
    user.LastName = "Tamayo Quiñones";
    user.City = "Terrassa";
    user.Birthday = new DateTime(1996, 10, 01);
    user.Street = "Rambla Francesc Macia";
    user.PC = "08226";
    user.Mobile = 646808080;

    Assert.IsFalse(mockEditUserViewModel.CheckInputs(user, "Agustin!123"));
```

Figura 3: Función CheckInputsEmptyTest Parte 1.

```

user.Password = "Agustin!123";
user.Email = "";

Assert.IsFalse(mockEditUserViewModel.CheckInputs(user, "Agustin!123"));

user.Email = "agustin.tamayo@e-campus.uab.cat";
user.FirstName = "";

Assert.IsFalse(mockEditUserViewModel.CheckInputs(user, "Agustin!123"));

user.FirstName = "Agustin";
user.LastName = "";

Assert.IsFalse(mockEditUserViewModel.CheckInputs(user, "Agustin!123"));

user.LastName = "Tamayo Quiñones";
user.City = "";

Assert.IsFalse(mockEditUserViewModel.CheckInputs(user, "Agustin!123"));

user.City = "Terrassa";
user.Birthday = new DateTime();

Assert.IsFalse(mockEditUserViewModel.CheckInputs(user, "Agustin!123"));

```

Figura 4: Función CheckInputsEmptyTest Parte 2.

```

user.Birthday = new DateTime(1996, 10, 01);
user.Street = "";

Assert.IsFalse(mockEditUserViewModel.CheckInputs(user, "Agustin!123"));

user.Street = "Rambla Francesc Macia";
user.PC = "";

Assert.IsFalse(mockEditUserViewModel.CheckInputs(user, "Agustin!123"));

user.PC = "08226";
user.Mobile = 0;

Assert.IsFalse(mockEditUserViewModel.CheckInputs(user, "Agustin!123"));

user.Mobile = 646808080;

Boolean task = mockEditUserViewModel.CheckInputs(user, "Agustin!123");

Assert.IsTrue(task);
}

```

Figura 5: Función CheckInputsEmptyTest Parte 3.

En los dos Test Case presentados se puede ver que se utiliza la clase *Assert* proporcionada por la herramienta de visual Studio. Gracias a esta, se han podido usar las funciones *IsTrue* e *IsFalse*.

2.2.2. Registro: Contraseñas coincidentes

Posteriormente se ha comprobado la correcta funcionalidad del Registro ante la introducción de contraseñas coincidentes. Es decir, el campo Contraseña y Verificación de Contraseña tienen los mismos datos. Para verificar esta parte se ha utilizado el Test Case `CheckInputsPasswordTest`.

```
[TestMethod]
public void CheckInputsPasswordTest()
{
    MockEditUserViewModel mockEditUserViewModel = new MockEditUserViewModel();

    UserModel user = new UserModel();
    user.Password = "Agustin!123";
    user.Email = "agustin.tamayo@e-campus.uab.cat";
    user.FirstName = "Agustin";
    user.LastName = "Tamayo Quiñones";
    user.City = "Terrassa";
    user.Street = "Rambla Francesc Macia";
    user.PC = "08226";
    user.Mobile = 646808080;
    user.Birthday = new DateTime(1996,10,01);

    Assert.IsFalse(mockEditUserViewModel.CheckInputs(user, "Agustin!1234"));

    Assert.IsFalse(mockEditUserViewModel.CheckInputs(user, ""));

    Assert.IsTrue(mockEditUserViewModel.CheckInputs(user, "Agustin!123"));

    user.Password = "";

    Assert.IsFalse(mockEditUserViewModel.CheckInputs(user, "Agustin!123"));
}
```

Figura 6: Función `CheckInputsPasswordTest`

2.2.3. Registro: Formato contraseña

Seguidamente, tenemos unos parámetros que nos indican si la contraseña cumple con ciertas características o no. Por ejemplo, esta tiene que tener entre 10 y 14 caracteres, una mayúscula, una minúscula, números y caracteres especiales.

Todo ello se ha agrupado en un Test Case llamado 'IsValidPasswordTest'. Este, prueba todos los posibles errores que se pueden cometer en el formato de la misma.

```
[TestMethod]
public void IsValidPasswordTest()
{
    MockEditUserViewModel mockEditUserViewModel = new MockEditUserViewModel();

    Assert.IsTrue(mockEditUserViewModel.IsValidPassword("Agustin!123"));
    Assert.IsFalse(mockEditUserViewModel.IsValidPassword(""));
    Assert.IsFalse(mockEditUserViewModel.IsValidPassword("Agustin.123"));
    Assert.IsFalse(mockEditUserViewModel.IsValidPassword("agustin!123"));
    Assert.IsFalse(mockEditUserViewModel.IsValidPassword("AGUSTIN!123"));
    Assert.IsFalse(mockEditUserViewModel.IsValidPassword("Agustin123"));
    Assert.IsFalse(mockEditUserViewModel.IsValidPassword("Agustin!"));
    Assert.IsFalse(mockEditUserViewModel.IsValidPassword("!123"));
    Assert.IsFalse(mockEditUserViewModel.IsValidPassword("123"));
    Assert.IsFalse(mockEditUserViewModel.IsValidPassword("Agustin"));
    Assert.IsFalse(mockEditUserViewModel.IsValidPassword("Agustin!1234567890"));
    Assert.IsFalse(mockEditUserViewModel.IsValidPassword("Ag!1"));
    Assert.IsFalse(mockEditUserViewModel.IsValidPassword("!!!!!!!!"));
}
```

Figura 7: Función IsValidPasswordTest

2.2.4. Registro: Formato del email

Para finalizar con los test sobre el registro, se procederá a validar el correo. Para ello hemos creado un Test Case que comprueba los diversos fallos que podemos cometer al introducir un email. Por ejemplo: No poner @ o no introducir el punto”.”.

```
[TestMethod]
public void IsValidEmailTest()
{
    MockEditUserViewModel mockEditUserViewModel = new MockEditUserViewModel();

    Assert.IsTrue(mockEditUserViewModel.IsValidEmail("agustin.tamayo@e-campus.uab.cat"));
    Assert.IsFalse(mockEditUserViewModel.IsValidEmail("agustin.tamayo@e-campusuabcat"));
    Assert.IsFalse(mockEditUserViewModel.IsValidEmail("agustin.tamayoe-campus.uab.cat"));
    Assert.IsFalse(mockEditUserViewModel.IsValidEmail("agustintamayoe-campusuabcat"));
    Assert.IsFalse(mockEditUserViewModel.IsValidEmail("@agustin.tamayoe-campus.uab.cat"));
    Assert.IsFalse(mockEditUserViewModel.IsValidEmail("agustin.tamayoe-campus.uab.cat@"));
}
```

Figura 8: Función IsValidPasswordTest

2.3. Test Case: Validación de resultados

Los dos Test Case presentados corresponden a los puntos 1 y 2 de los escenarios definidos en el documento "**TEST: DEFINICIÓN DE ESCENARIOS**".

La correcta ejecución de los test y la validez de las funcionalidades probadas, se ha comprobando viendo los resultados reflejados en la siguiente imagen:

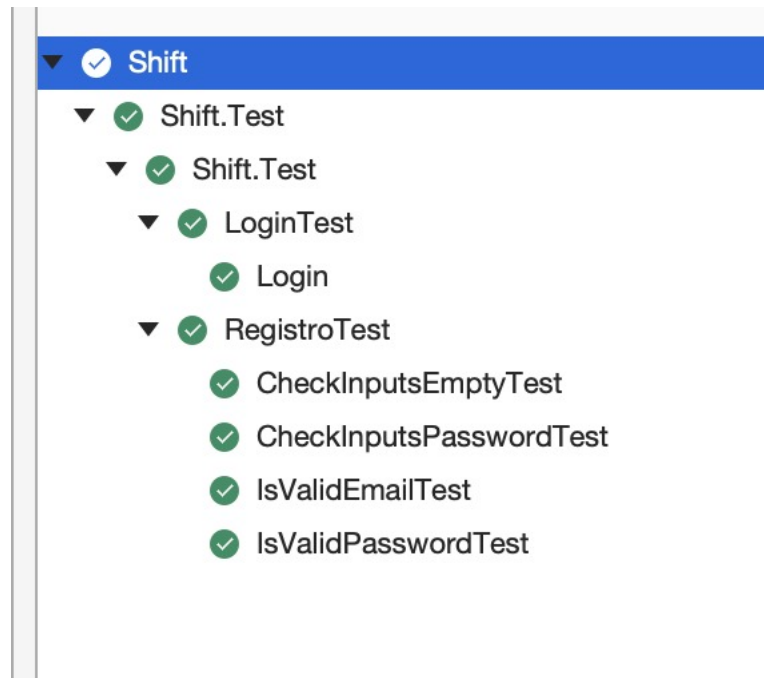


Figura 9: Todas las funcionalidades testeadas son correctas.

3. Revisión Técnica Formal (RTF)

El equipo también ha realizado un análisis de código en estático, es decir, sin ejecutar su funcionalidad. Mediante este proceso se ha evaluado el formato, estilo de codificación, variables, comentarios, entre otros aspectos.

El tiempo invertido en la detección de errores ha sido de 2 días. Durante periodo citado, se anotaron y comunicaron todas las erratas percibidas a los desarrolladores. La comunicación entre equipos fue clave para realizar exitosamente la RTF puesto que hacía falta precisar mucho en el nivel de detalle e implicaciones en los cambios.

Finalmente, destacar que se estipuló un periodo de 1 semana para corregir los puntos mostrados en la siguiente imagen.

Inspection Issue Log

Project:	Origin:	Requirements, Design, Construction, Testing
Inspection ID:	Type:	Missing, Wrong, Extra, Usability, Performance, Style, Clarity, Question
Meeting Date:	Severity:	Major, minor
Recorder:		

Defects Found: _____, Major _____, minor _____.

#	Origin	Type	Severity	Location	Description
1	R	W	M	EditUserViewModel.cs Linia 135	El mensaje de error no es correcto, ya que no tiene relación con el error que es realmente
2	R	W	M	EditUserViewModel.cs Linia 141	Falta que el método devuelva False en el caso de que haya errores.
3	C	W	M	EditUserViewModel.cs Linia 129	Hay que cambiar la igualación y usar el método compare
4	R	W	M	ApiClient.cs Linia 91	Falta añadir condicion action != register
5	D	S	m	EditAlarmViewModel.cs Linia 46-55	Variable pública, tiene que ser privada
6	D	S	m	EditAlarmViewModel.cs Linia 97	Falta añadir otro tipo de alarma "Accidente"
7	R	M	m	EditAlarmViewModel.cs Linia 153	Falta añadir una doble verificación antes de volver atrás
8	D	S	m	ActivatedAlarmViewModel.cs Linia 21-27	Variable pública tiene que ser privada
9	R	M	m	ActivatedAlarmViewModel.cs Linia 86	Falta controlar una segunda verificación
10	D	S	m	AlarmViewModel.cs Linia 28-29	Variable pública tiene que ser privada
11	R	M	m	AlarmListViewModel.cs Linia 110	Falta confirmación a la hora de borrar la alarma
12	R	M	m	AlarmListViewModel.cs Linia 120	Falta mensaje informativo conforme se ha eliminado la alarma
13	D	S	m	EditAlarmViewModel.cs Linia 62-63	Variable tiene que tener formato lowerCase.
14	R	S	m	ActivatedAlarmViewModel.cs Linia 102	Falta mensaje conforme se ha cancelado la escucha

Copyright © 2001 by Karl E. Wiegers. Permission is granted to use, modify, and distribute this document.

Figura 10: Recopilación de errores encontrados al aplicar la RTF.

4. Exploratory Testing

En el Exploratory Testing se ha verificado el correcto comportamiento de las distintas funcionalidades mientras el código se ejecuta en un dispositivo Android.

4.1. Exploratory: Creación de usuario

El primer caso se centra en la creación de un nuevo usuario. Las pruebas que se han desarrollado son las siguientes:

1. Introducción de los campos vacíos.
2. Introducción de un valor que sobrepasa el valor lógico.
3. Introducción errónea en el campo de verificación de contraseña.
4. Introducción de una contraseña que no cumpla los parámetros.
5. Introducción de todos los campos correctamente.

Los resultados obtenidos son los siguientes:

1. El primer paso será la inserción de campos vacíos para ver como reacciona SHiFT. Habiendo visto una ejecución esperada de no dejarnos registrarnos, nos aparece por pantalla el mensaje indicado "Se deben de rellenar todos los campos".
2. A continuación, procedemos a introducir todos los campos. En este caso, en el campo del Código postal (PC) se ha introducido un valor incorrecto: "0800000000000000". La aplicación se comporta de forma esperada sin dejarnos completar el registro y mostrando el mensaje: "No se ha podido completar el registro".
3. Para realizar el registro debemos de introducir 2 veces la contraseña. Si en el campo de verificación de contraseña introducimos una que difiera de la primera, el sistema no permite completar el registro y muestra un error de "Las contraseñas no coinciden".
4. Al introducir una contraseña que no cumple los parámetros necesarios, y esa misma la introducimos en el campo de verificación de contraseña, vemos que nos aparece el error "Formato de contraseña inválido". La contraseña debe de tener entre 10 y 14 caracteres: 1 mayúscula, 1 minúscula y un carácter especial.
5. Finalmente, tras rellenar todos los campos adecuadamente, el registro se completa con éxito y nos permite navegar correctamente a la siguiente pantalla.

4.2. Exploratory: Login

A continuación comprobaremos el comportamiento del Login. Las pruebas que se han desarrollado son las siguientes:

1. Login con un usuario inexistente.
2. Login con un usuario existente pero con una contraseña inválida.
3. Login correctamente y comprobación de navegación entre pantallas.

Los resultados obtenidos son los siguientes:

1. Al introducir un usuario que no existe "Tesst", podemos ver que no nos deja acceder mostrándonos por pantalla el mensaje "The given header was not found.". Puede apreciarse que el mensaje está en un idioma que difiere del Español y que no es suficientemente auto-explicativo.
2. Al introducir una contraseña que no es correcta, podemos ver que nos devuelve el mismo error que el apartado anterior.
3. El Login con un usuario válido nos permite acceder y navegar correctamente a la siguiente pantalla.

4.3. Exploratory: Reconocimiento de voz

Para el reconocimiento de voz, hemos realizado 5 pruebas de sonido. Nuestro objetivo es identificar la tasa de fallos que hay al activar las alarmas mediante los comandos de voz.

1. Primero hemos creado cinco alarmas. Estas se activarán mencionando las siguientes frases:
 - a) Oye Shift incendio.
 - b) Oye Shift accidente.
 - c) Oye Shift necesito un taxi.
 - d) Oye Shift pídemme ayuda.
 - e) Oye Shift agresión.
2. Seguidamente, activaremos cada alarma 3 veces. **Los resultados obtenidos son los siguientes:**
 - a) Oye Shift incendio:
 - 1) Aciertos: 3
 - 2) Errores: 0
 - b) Oye Shift accidente:
 - 1) Aciertos: 3
 - 2) Errores: 0
 - c) Oye Shift necesito un taxi:
 - 1) Aciertos: 2
 - 2) Errores : 1
 - d) Oye Shift pídemme ayuda:
 - 1) Aciertos: 3
 - 2) Errores: 0
 - e) Oye Shift agresión:
 - 1) Aciertos: 3
 - 2) Errores: 0

3. Ahora procedemos a realizar un recuento para observar la tasa de fallos que tenemos. De un total de 15 pruebas únicamente ha fallado 1 vez. Por este motivo, concluimos que al tener una tasa de fallos del 6,66 % se cumple con el requisito REQ-NF-1-04”.

ID	REQ-NF-1-04
Título	Tasa de fallos de la detección de sonidos
Descripción	El sistema SHIFT no puede tener una tasa de fallos en la detección de sonidos superior al 10 %
Prioridad	M
Verificación	T
Padres	REQ-F-1-02

Figura 11: Requisito cumplido debido que la tasa de fallos es inferior al 10 %.

4.4. Exploratory: Geolocalización

El sistema SHiFT tiene un requisito definido que impone un margen de error inferior a 5 metros durante el proceso de geolocalización.

ID	REQ-NF-1-07
Título	Rango geolocalización
Descripción	El sistema SHiFT tiene que permitir guardar detectar la geolocalización con un margen de error de 5 metros.
Prioridad	M
Verificación	T
Padres	REQ-F-1-09

Figura 12: Requisito definido para imponer una tasa de error inferior a 5 metros en la geolocalización.

Para verificar este punto se han realizado los siguientes pasos:

1. Activación de una alerta.
2. Acceder al servicio web de emergencias para verificar la correcta emisión de la alerta.



The screenshot shows the SHiFT web interface. At the top left is the SHiFT logo with a signal icon. Below it is a 'Sign Out' link. The main content is a table with the following columns: ID_Warning, ID_User, FirstName, LastName, Email, Tel, Date, GEO-XY, and Active. There are two rows of data. The first row (ID 97) shows an 'Unactive' status with 'Turn Off' and 'Maps' buttons. The second row (ID 98) shows an 'Active' status with 'Turn Off' and 'Maps' buttons.

ID_Warning	ID_User	FirstName	LastName	Email	Tel	Date	GEO-XY	Active
97	18	judith	bellido	judith@gmail.com	111234567	2020-06-04 11:04:05		Unactive
98	36	judd	bellido	user6@gmail.com	123456789	2020-06-04 16:49:31	41.4259369,2.138525	Active

Figura 13: La alerta activada aparece en la página web de los servicios de emergencia.

3. Acceder a Google Maps y ver dónde SHiFT nos ubica.
4. Acceder a Google Maps mediante un dispositivo independiente al usado y ver la diferencia entre coordenadas. También se observa si el sitio indicado coincide geográficamente con la ubicación en la que nos encontramos.

En la siguiente fotografía podemos ver como la chincheta y el punto azul están en la misma posición. Siendo las coordenadas las mismas que hay en la plataforma web.

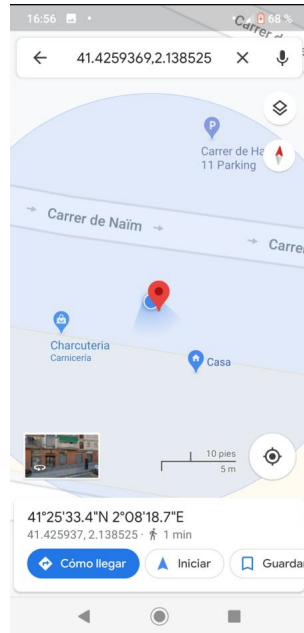


Figura 14: Contraste entre coordenadas detectadas por SHiFT y otro dispositivo independiente.

Gracias a los puntos tratados y las imágenes obtenidas podemos verificar que el requisito "REQ-NF-1-07" se cumple correctamente.

4.5. Exploratory: Modificación del perfil

También se han realizado pruebas para verificar la modificación del perfil. Se han seguido los siguientes pasos:

1. Primero se ha creado un usuario con los siguientes campos:
 - a) Password = “Agustin!1234”
 - b) Email = “correo@electronico”
 - c) FirstName = “Nombre”
 - d) LastName = “Apellido”
 - e) City = “bcn”
 - f) Birthday = ”03/01/1900”
 - g) Street = “calle”
 - h) PC = “12345”
 - i) Mobile = 123456789



Figura 15: Perfil de usuario antes de modificar los campos.

2. Después procedemos a cambiar los datos por los siguientes:

- a) Password = "Agustin!1234"
- b) Email = "correo@electronico"
- c) FirstName = "NombreModificado"
- d) LastName = "ApellidoModificado"
- e) City = "barcelona"
- f) Birthday = "26/01/1900"
- g) Street = "carrer"
- h) PC = "54321"
- i) Mobile = 987654321



Figura 16: Perfil de usuario después de modificar los campos.

Se puede concluir que la funcionalidad "Modificar Perfil" es correcta. Durante la demostración no se ha modificado email para poder comprobar que es el mismo usuario.

4.6. Exploratory: LogOut

Por otro lado hemos comprobado que podemos salir de la app mediante el botón "Cerrar Sesión" del menú de usuario. Como puede apreciarse en la siguiente imagen, se ha realizado el Logout correctamente.

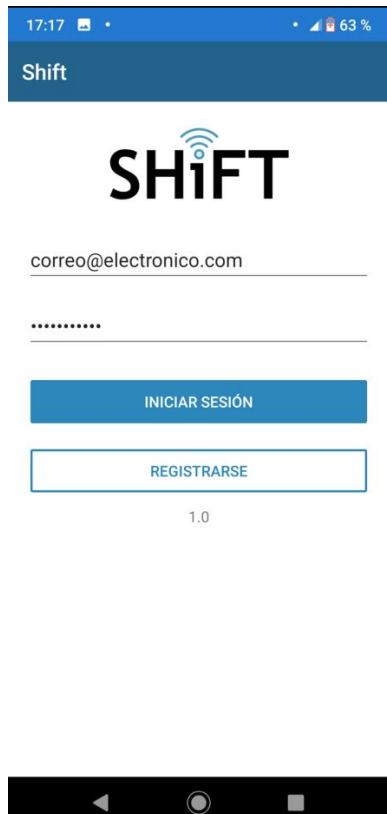


Figura 17: Pantalla mostrada en la aplicación SHiFT tras presionar el botón de "Cerrar Sesión"

4.7. Exploratory: Creación de alarmas

Uno de los aspectos a los que le hemos dado más énfasis es el proceso de creación de alarmas. Para ello hemos seguido los siguientes pasos:

1. Crear una alarma de tipo Incendio.
2. Configurar "Incendio" como nombre de la alarma.
3. Configurar la invocación de voz de la alarma. En este caso la palabra clave es "Incendio".

El resultado del procedimiento citado es el siguiente:

En el panel de alarmas nos aparece la alerta "Incendio" configurada:

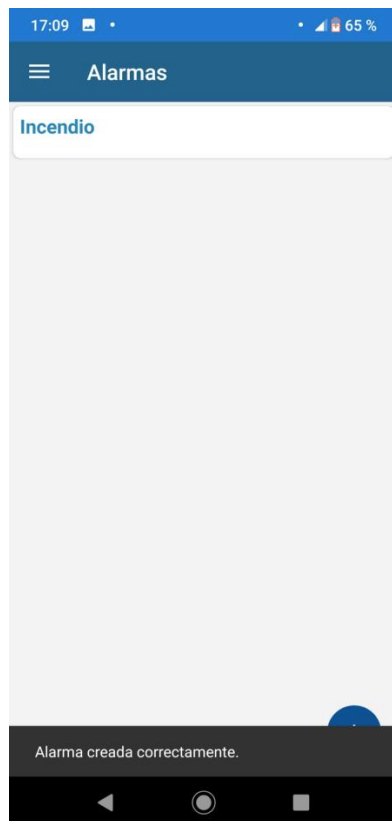


Figura 18: Pantalla mostrada en la aplicación SHiFT tras crear la alarma "Incendio".

Si presionamos la alarma configurada, apreciamos que su detalle coincide con las configuraciones realizadas:

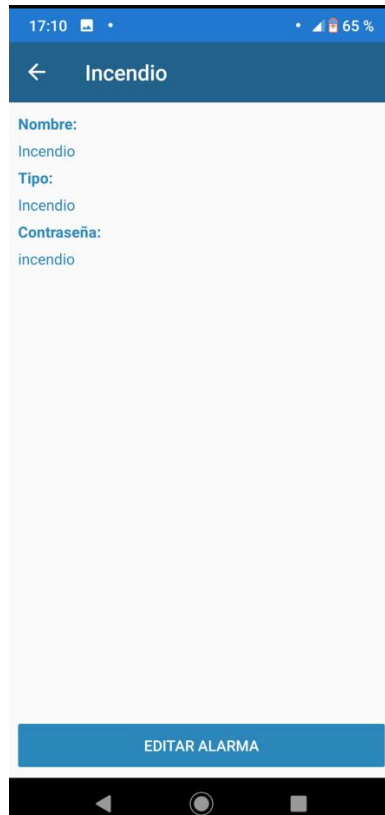


Figura 19: Pantalla mostrada en la aplicación SHiFT tras entrar en los detalles de la alarma "Incendio".

4.8. Exploratory: Modificación de alarmas

Ahora que ya tenemos la alerta creada, vamos a modificarla. Para ello modificaremos el nombre de la alarma llamada "Incendio" por el nombre "IncendioModificada".



Figura 20: Pantalla mostrada en la aplicación SHiFT tras entrar en los detalles de la alarma "Incendio".

4.9. Exploratory: Activación de alarmas

Finalmente procederemos a activar la alarma creada y modificada para verificar su correcto comportamiento. Para ello, se han seguido los siguientes pasos:

1. Situarnos en la pantalla principal de SHiFT.

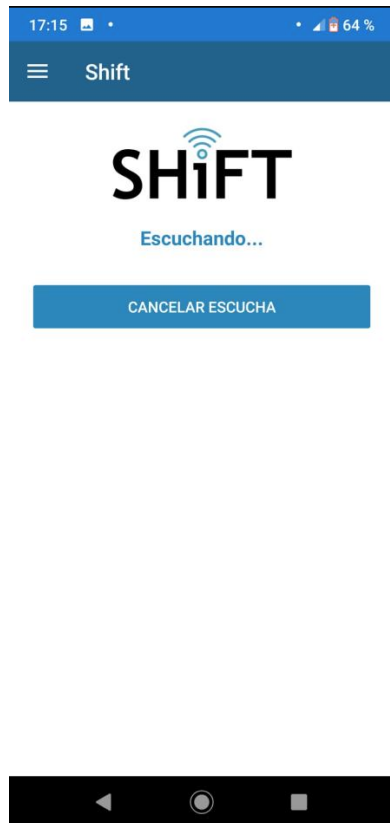


Figura 21: Pantalla principal de SHiFT.

2. Iniciar la activación de la alarma diciendo la frase de inicio "Oye Shift".

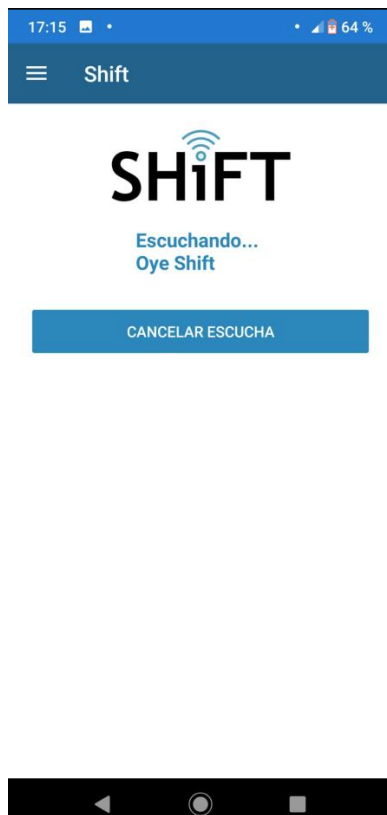


Figura 22: Pantalla principal con la frase de activación mencionada.

3. Invocamos y completamos la activación de la alarma diciendo la palabra secreta configurada: "Incendio".

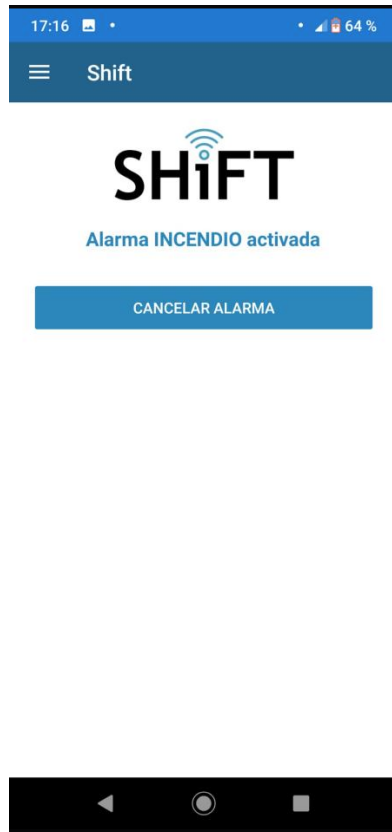


Figura 23: Pantalla principal con la alarma de tipo "Incendio" activada.

4.10. Exploratory: Transmisión de alertas

El último Exploratory Test pretende comprobar la correcta transmisión de alarmas activadas entre el Front-End y Back-End.

Como puede apreciarse a continuación, la alarma activada en el apartado anterior, se refleja correctamente en el portal web de los servicios de emergencia.



ID_Warning	ID_User	FirstName	LastName	Email	Tel	Date	GEO-XY	Active		
97	18	judith	bellido	judith@gmail.com	111234567	2020-06-04 11:04:05		Unactive	Turn Off	Maps
99	38	NombreModificado	ApellidoModificado	correo@electronico.com	987654321	2020-06-04 17:15:59	41.4259362,2.1385211	Active	Turn Off	Maps

Figura 24: Pantalla del navegador web donde se puede ver la alerta activada.

5. Back-End

Para hacer los test de Back-End hemos utilizado la herramienta *online* ReqBin. Esta nos permite crear paquetes POST con distintas características para ver el comportamiento y respuesta del servidor.

5.1. Back-End: Login

Una vez teníamos la base de datos creada y el servidor funcional, se realizó un *testing* para verificar su correcta configuración y respuesta. Primeramente mostraremos las pruebas de Login, que a diferencia de las vistas en el apartado [2 - Test Case](#), estas se dirigen al servidor.

Cabe destacar que algunos de los usuarios registrados en la BBDD son:

- judit@gmail.com con contraseña "judith".
- agustintamayo@gmail.com con contraseña "agustin".
- martinsusin@gmail.com con contraseña "martin".

5.1.1. Login: Login correcto

Verificamos si podemos realizar Login con el usuario "judith@gmail.com".

Post HTTP Requests Online

Send HTTP requests to the server and check server responses

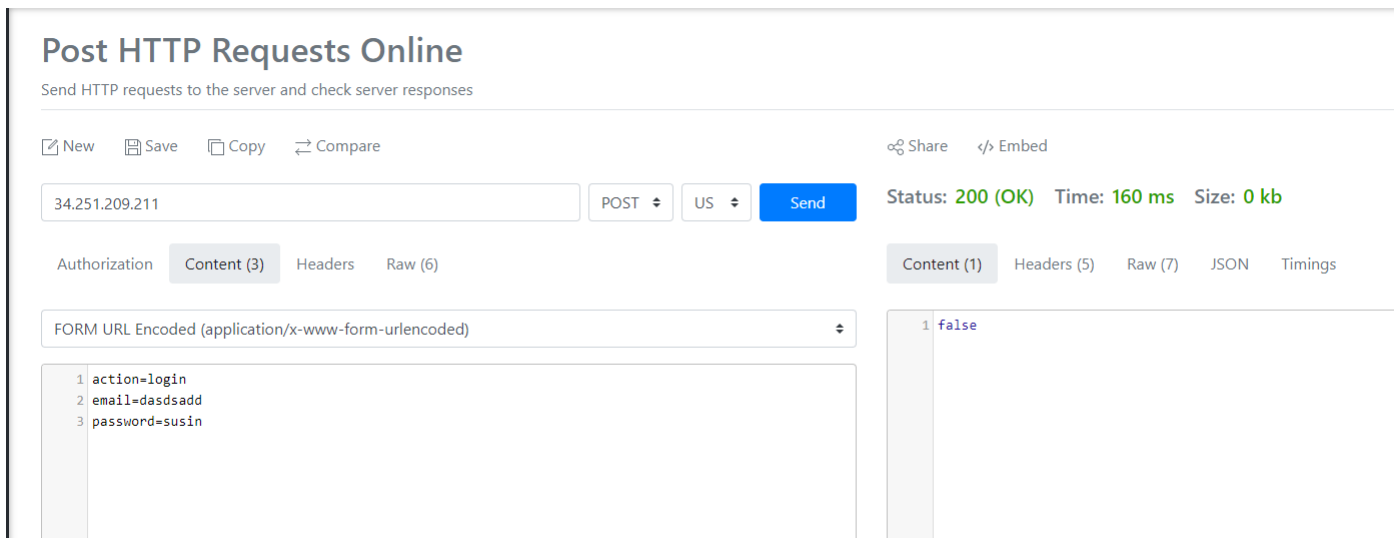
The screenshot shows the Post HTTP Requests Online interface. At the top, there are buttons for 'New', 'Save', 'Copy', and 'Compare'. Below these, the URL '34.251.209.211' is entered, and the method is set to 'POST'. The 'Send' button is highlighted. To the right, the status is '200 (OK)', time is '330 ms', and size is '0 kb'. Below the URL bar, there are tabs for 'Authorization', 'Content (3)', 'Headers (1)', and 'Raw (7)'. The 'Content (3)' tab is selected, showing the request body in 'FORM URL Encoded (application/x-www-form-urlencoded)' format. The body contains three lines: '1 action=login', '2 email=judith@gmail.com', and '3 password=judith'. To the right of the request body, there are tabs for 'Content (1)', 'Headers (6)', 'Raw (8)', 'JSON', and 'Timings'. The 'Content (1)' tab is selected, showing the response body as '1 true'.

Figura 25: Login con usuario judith@gmail.com.

Podemos observar que nos devuelve un 200 OK y *true* en el cuerpo de la respuesta. Gracias a la imagen mostrada concluimos que la funcionalidad de Login, para usuarios existentes, se desarrolla correctamente en el Back-End.

5.1.2. Login: Login con un usuario incorrecto

En el siguiente caso observaremos el comportamiento del servidor ante un usuario inexistente intentando hacer Login. El email introducido es "dasdsadd" y la contraseña "susin".



The screenshot shows the Postman interface for an HTTP request. The URL is 34.251.209.211, the method is POST, and the content type is US. The request body is FORM URL Encoded (application/x-www-form-urlencoded) with the following parameters: action=login, email=dasdsadd, and password=susin. The response status is 200 (OK), the time is 160 ms, and the size is 0 kb. The response body is a single line: 1 false.

Request	Response
<pre>1 action=login 2 email=dasdsadd 3 password=susin</pre>	<pre>1 false</pre>

Figura 26: Login con un usuario inexistente.

El resultado obtenido coincide con el esperado. Como puede apreciarse, el servidor devuelve un 200 OK indicando que la petición se ha procesado y respondido adecuadamente. Sin embargo, el cuerpo del mensaje enviado por el Back-End contiene un *false*. Partiendo de los índices reflejados, podemos concluir que el servidor reacciona adecuadamente ante Logins incorrectos.

5.1.3. Login: Login con una contraseña incorrecta

Seguidamente, comprobaremos si el servidor tiene el comportamiento adecuado ante un usuario válido "martinsusin@gmail.com" y una contraseña incorrecta "susin". En este caso,

The screenshot shows a web client interface with the following details:

- URL:** 34.251.209.211
- Method:** POST
- Port:** US
- Status:** 200 (OK)
- Time:** 160 ms
- Size:** 0 kb
- Content Type:** FORM URL Encoded (application/x-www-form-urlencoded)
- Request Body:**

```
1 action=login
2 email=martinsusin@gmail.com
3 password=susin
```
- Response Body:**

```
1 false
```

Figura 27: Login con el usuario "martinsusin@gmail.com" pero con una contraseña incorrecta.

El resultado obtenido coincide con el esperado, tal y como se indica en el punto anterior.

5.2. Back-End: Cookies

Para proceder con los test que se mostrarán a continuación, necesitamos indicarle al servidor que ya hemos iniciado sesión correctamente. Para ello, en la cabecera de la petición POST configuramos la Cookie que nos retorna el servidor ante Logins correctos. Esta contiene el correo electrónico del usuario con el que interactúa.

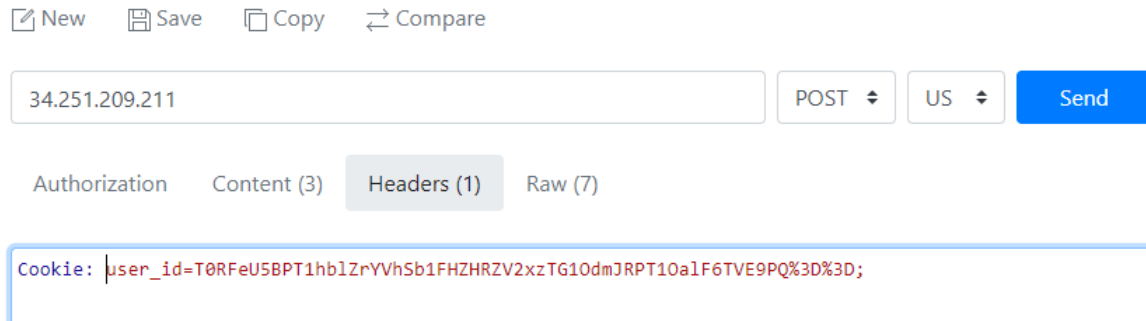


Figura 28: Configuración de la Cookie en la *header* de la petición POST.

5.3. Back-End: Crear alerta

Llegados a este punto, la petición POST esta configurada para que el servidor interprete que ya hemos hecho Login. Así pues, procederemos a verificar la correcta creación de alertas. Para hacerlo, mandaremos el JSON mostrado en la Figura 29.

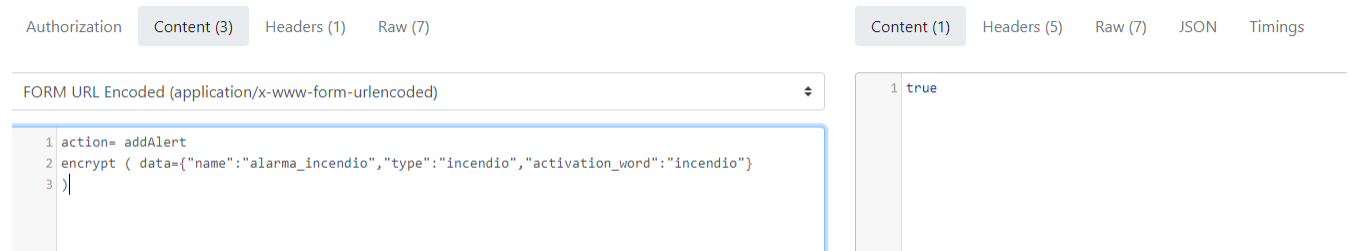


Figura 29: Creación de la alerta mediante el JSON al servidor.

Sin embargo hay una pequeña modificación la cual no aparece en la imagen para facilitar la comprensión del proceso. Como puede apreciarse, este aparece entre *encrypt (JSON)*. Esto es porque primeramente debe de encriptarse el JSON mediante el siguiente proceso:

```
function encrypt($data){  
    //Encrypting data  
    $back_salt = base64_encode(rand(1000, 9999));  
    $front_salt = base64_encode(rand(1000, 9999));  
    $info = base64_encode($data);  
    $encrypt = $back_salt.'.'.$info.'.'.$front_salt;  
    $encrypt = base64_encode($encrypt);  
    return $encrypt;  
}
```

Figura 30: Proceso para encriptar la información a mandar al servidor.

Como puede apreciarse, el valor retornado por el servidor es *true*, verificando el correcto comportamiento de la creación de alertas.

5.4. Back-End: Activar alerta

Una vez creada la alerta, se procederá a activarla. Para ello enviaremos un paquete de activación que contendrá el JSON encriptado con la alarma que queremos activar (la que se ha generado en el apartado anterior). Para facilitar la comprensión del proceso, se vuelve a poner el JSON en claro para realizar la captura.

The screenshot shows a web client interface with a URL field containing '34.251.209.211', a dropdown menu set to 'POST', and a 'Send' button. Below the URL field, there are tabs for 'Authorization', 'Content (3)', 'Headers (1)', and 'Raw (7)'. The 'Content (3)' tab is selected, showing the request body in 'FORM URL Encoded (application/x-www-form-urlencoded)' format. The request body contains the following JSON data:

```
1 action = activateAlert
2 encrypt(
3 data = {"type": "incendio"})
```

On the right side, the response status is '200 (OK)' with a time of '205 ms' and a size of '0 kb'. Below the status, there are tabs for 'Content (1)', 'Headers (5)', 'Raw (7)', 'JSON', and 'Timings'. The 'Raw (7)' tab is selected, showing the raw response data:

```
HTTP/1.1 200 OK
Date: Thu, 04 Jun 2020 08:41:09 GMT
Server: Apache/2.4.29 (Ubuntu)
Content-Length: 5
Content-Type: application/json

true
```

Figura 31: Proceso para activar una alerta.

La siguiente figura, conjuntamente con la anterior, demuestra el correcto funcionamiento del proceso testado.

ID_Warning	ID_User	FirstName	LastName	Email	Tel	Date
97	18	judith	bellido	judith@gmail.com	111234567	2020-06-04 11:04:05

Figura 32: Pagina web con la alerta activa.

5.5. Back-End: Transmitir geolocalización

Finalmente, se simulará el envío de coordenadas y dispositivos cercanos tal y como lo haría un dispositivo móvil.

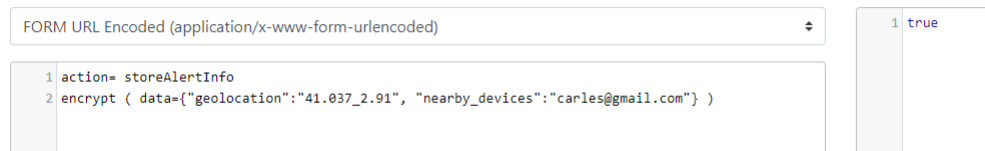


Figura 33: Simulación de paquetes enviados mientras un usuario tiene una alerta activada.

Mediante la respuesta *true* por parte del servidor verificamos el correcto comportamiento de la funcionalidad. Para justificar más el escenario, podemos dirigirnos a las Figuras 24 y 13.