

4. MessagesFX

Unidad 4. Ejercicio final

Programación de Servicios y Procesos



Esta obra está bajo una [Licencia Creative Commons Atribución-NoComercial-CompartirIgual 4.0 Internacional](https://creativecommons.org/licenses/by-nc-sa/4.0/).

Tabla de contenido

1. Introducción y primeros pasos.....	3
1.1. Configurar el proyecto.....	3
1.2. Vista <i>login</i>	3
1.3. Vista <i>register</i>	4
1.4. Vista <i>messages</i>	4
1.5. Eliminar un mensaje.....	5
1.6. Enviar un mensaje a un usuario.....	5
2. Cómo implementar.....	7
2.1. Clases <i>Model</i> y <i>Response</i>	7
2.2. Cargando diferentes vistas.....	7
2.3. Serializar <i>LocalDate</i> en JSON.....	7
2.4. Mostrar imágenes en un <i>TableView</i>	8
2.5. Mostrar la imagen en un <i>ImageView</i>	8
3. Mejora opcional.....	9
4. Reglas de evaluación.....	10
4.1. Parte obligatoria.....	10
4.2. Acerca de las mejoras opcionales.....	10

1. Introducción y primeros pasos

En este ejercicio final, se pedirá que se implemente una aplicación JavaFX para los servicios implementados en el ejercicio final de la Unidad 3. Si tienes alguna duda sobre cómo implementar algo, busca primero en la sección Cómo implementar de este documento, y si no encuentras la solución allí, no dudes en preguntar en los foros, etc.

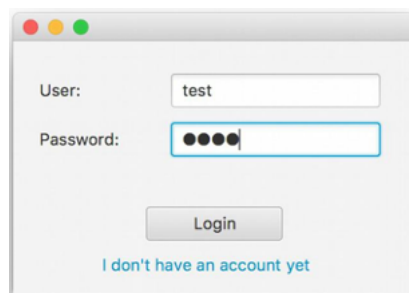
1.1. Configurar el proyecto

Crea un proyecto JavaFX llamado **MessagesFX**. Este proyecto constará de 3 vistas diferentes: *register*, *login* y *messages*.

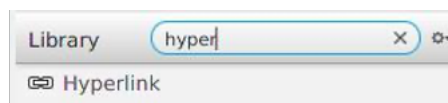
1.2. Vista *login*

Esta vista proporcionará un formulario para que el usuario pueda indicar su nombre y contraseña. Una vez que esa información se envíe al servicio **/login** (POST), se debe verificar la respuesta del servidor y, en caso de que el inicio de sesión sea correcto, el usuario deberá ser redirigido a la vista **messages**.

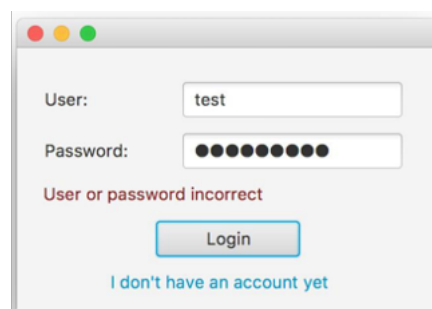
No olvides almacenar el *token* en la clase *ServiceUtils* llamando al método **setToken()**.



Esta vista también muestra el texto “*I don't have an account yet*”. Este elemento es un hipervínculo y se comporta exactamente como un botón en JavaFX. Al hacer clic, te llevará a la vista **register**.



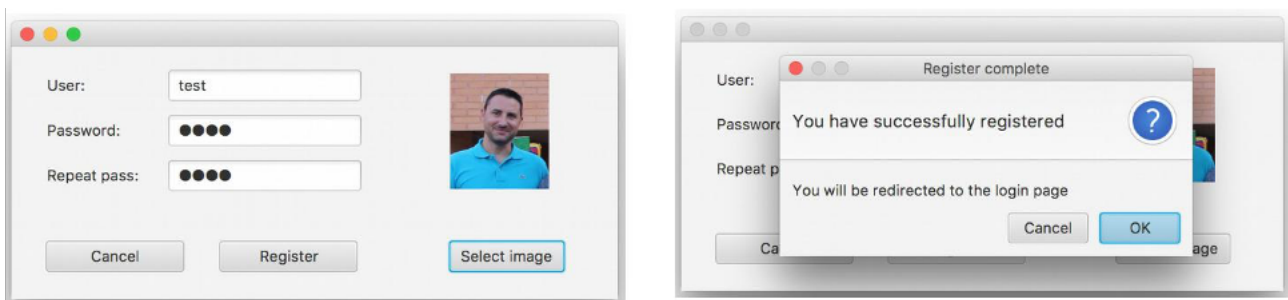
En caso de que ocurra algún error durante el proceso de inicio de sesión, se deberá informar al usuario:



1.3. Vista *register*

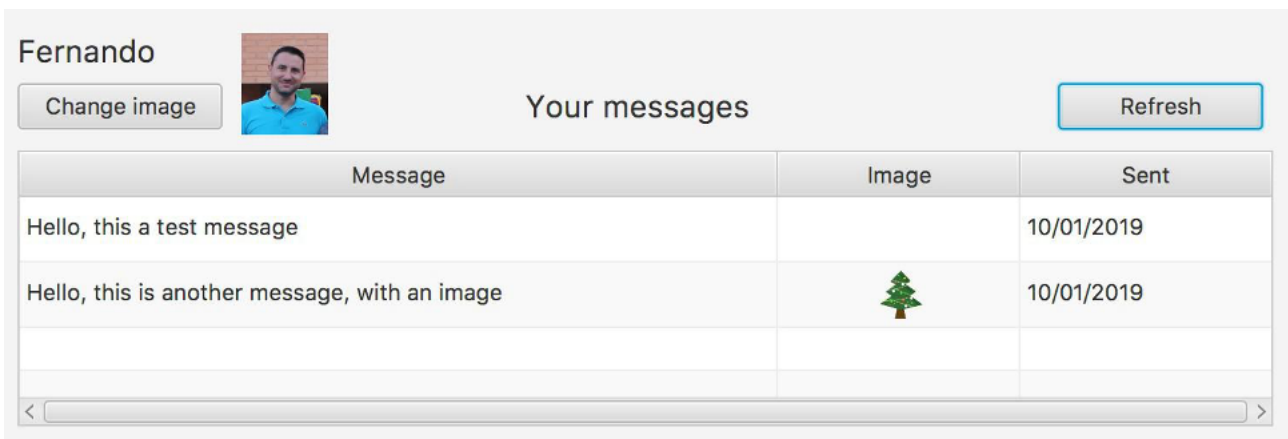
Esta vista proporcionará un formulario para que se puedan registrar nuevos usuarios. Debes comprobar que ningún campo esté vacío y también que las 2 contraseñas sean iguales (se debe mostrar un error cuando las contraseñas no coincidan). Si todo está bien, llama al servicio **/register** (POST).

El usuario debe ser informado en caso de que algo fallara, o si todo fue correcto, se debe mostrar un *Alert* con un mensaje para indicar que será redirigido a la página de inicio de sesión (y también debe realizar la redirección):



1.4. Vista *messages*

Cuando un usuario inicia sesión, deberá aparecer automáticamente una vista con sus mensajes (ver sección 2.4 para saber cómo cargar imágenes en la tabla):




Como puedes ver, no solo se muestran los mensajes, sino también el nombre y la imagen de avatar del usuario registrado.

También debes proporcionar alguna forma de llamar al servicio **/messages** (GET) manualmente para recuperar los mensajes nuevamente, por ejemplo, con un botón *Refresh*, como se muestra en la captura de pantalla anterior.

En cuanto a la información del usuario, cualquier usuario registrado debe poder cambiar su imagen, permitiéndole elegir una nueva. Se debe llamar al servicio **/user** (PUT) para enviar la nueva imagen al servidor. Como se explicó en el ejercicio anterior, se genera una nueva ruta y se actualiza el perfil de usuario en la base de datos.

1.5. Eliminar un mensaje

Si el usuario selecciona un mensaje en la lista, se debe habilitar un botón (deshabilitado por defecto) para llamar al servicio `/messages` (DELETE) para eliminar el mensaje seleccionado de la base de datos:




Message	Image	Sent
Hello, this a test message		10/01/2019
Hello, this is another message, with an image		10/01/2019

1.6. Enviar un mensaje a un usuario


Debajo de la lista de mensajes del usuario registrado, se debe mostrar la lista de todos los usuarios registrados para permitir una forma de enviar mensajes a cualquiera de ellos:

Send a message

Users

Avatar	Nick name
	Fernando
	Ana
	Juan

Message




Para enviar un mensaje, se debe seleccionar el usuario “a” y se debe proporcionar al menos el texto del mensaje (el campo de imagen es opcional). Cuando se escribe algo en el cuadro de texto, el botón **Send message** debe estar habilitado (debe estar deshabilitado de forma predeterminada).

A continuación, tienes una vista completa de la ventana que contiene tanto el perfil de usuario, las listas (mensajes y usuarios) como todos los botones necesarios:


Fernando

Change image



Your messages




Refresh

Message	Image	Sent
Hello, this a test message		10/01/2019
Hello, this is another message, with an image		10/01/2019

Delete message


Send a message

Users

Avatar	Nick name
	Fernando
	Ana
	Juan

Message

Hello, this a message with a picture



Select image

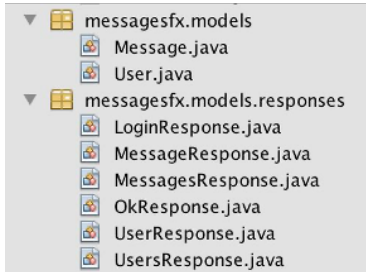
Send message

Programación de Servicios y Procesos - Desarrollo de servicios con Node.js

6

2. Cómo implementar

2.1. Clases *Model* y *Response*



Crea un paquete para las clases modelo que representan un usuario y un mensaje. También cree las clases necesarias (si estás usando la biblioteca *Gson*) para mapear todas las posibles respuestas que provienen del servidor.

Una buena práctica será que todo tipo de respuestas hereden de *OkResponse* (*ok* → *boolean*, *error* → *String*).

2.2. Cargando diferentes vistas

Consulta la Unidad 1 Parte V (Más conceptos de JavaFX) para saber cómo cargar una nueva vista o escena (Sección Aplicaciones con múltiples vistas). Una buena práctica sería tener un método global para cargar una nueva escena:

```
public class ScreenLoader {  
    public static void loadScreen(String viewPath, Stage stage) throws IOException {  
        Parent view = FXMLLoader.load(ScreenLoader.class.getResource(viewPath));  
        Scene view1Scene = new Scene(view);  
  
        stage.hide();  
        stage.setScene(view1Scene);  
        stage.show();  
    }  
}
```

2.3. Serializar *LocalDate* en JSON

Si eliges el campo “*sent*” en la clase *Message* para que sea *LocalDate* (mejor práctica) en lugar de un *String*, deberás enviar información adicional a la biblioteca *Gson*, para que sepa cómo administrarla:

De *LocalDate* a *String* (Objeto → JSON)

```
Gson gson = new GsonBuilder().registerTypeAdapter(LocalDate.class,  
    (JsonSerializer<LocalDate>)(date, typeOfSrc, context) -> {  
        return new JsonPrimitive(date.toString());  
    }).create();
```

De *String* a *LocalDate* (JSON → Objeto)

```
Gson gson = new GsonBuilder().registerTypeAdapter(LocalDate.class,  
    (JsonDeserializer<LocalDate>)(json, type, jsonDeserializationContext) -> {  
        return ZonedDateTime.parse(json.getAsJsonPrimitive().getString()).toLocalDate();  
    }).create();
```

2.4. Mostrar imágenes en un *TableView*

Para mostrar una imagen dentro de una celda en una tabla, simplemente configura el tipo de datos de esa columna como un *ImageView*:

```
private TableColumn<Message, ImageView> msgImageCol;  
private TableColumn<User, ImageView> userImageCol;
```

Luego, vincula esa columna a un *getter* llamado “*getImageView*” en la clase *User* o *Message*:


```
msgImageCol.setCellValueFactory(new PropertyValueFactory("imageView"));  
userImageCol.setCellValueFactory(new PropertyValueFactory("imageView"));
```

Crea el objeto *ImageView* en el método, establece una altura fija y devuélvelo:

```
class Message {  
    ...  
    public ImageView getImageView() {  
        ImageView imgView = new ImageView(ServiceUtils.SERVER + "/" + image);  
        imgView.setFitHeight(30);  
        imgView.setPreserveRatio(true);  
  
        return imgView;  
    }  
    ...  
}
```

También es mejor centrar verticalmente el texto en todas las columnas (y centrar la imagen en la columna correspondiente). Selecciona la columna y ve a la sección de propiedades (CSS):

Columna imagen → 

Cada otra columna → 

2.5. Mostrar la imagen en un *ImageView*

Si la imagen es un objeto *File*, puedes ponerla en un *ImageView* así:

```
msgImgView.setImage(new Image(msgImgFile.toURI().toString()));
```

Si es una URL, simplemente configura la cadena con la dirección completa:

```
msgImgView.setImage(new Image(ServiceUtils.SERVER + "/" + message.getImage()));
```


3. Mejora opcional

- Añade dos columnas a la lista de mensajes para que se pueda identificar al remitente de cada mensaje, mostrando su nombre e imagen de avatar.

4. Reglas de evaluación

4.1. Parte obligatoria

Para obtener la nota final, se aplicarán los siguientes criterios:

- La aplicación está bien estructurada con múltiples vistas: **0,5 puntos**
- Página de inicio de sesión: **1 punto**
- Página de registro: **1,5 puntos**
- Actualización de la imagen del usuario registrado: **1 punto**
- Obtener todos los mensajes y mostrarlos en la tabla: **1 punto**
- Obtener todos los usuarios y mostrarlos en la tabla: **1 punto**
- Actualización de listas de mensajes y usuarios: **0,5 puntos**
- Eliminar un mensaje: **1 punto**
- Envío de un mensaje: **1,5 puntos**
- Comprobando y mostrando mensajes de error, código limpio y comentarios: **1 punto**

4.2. Acerca de las mejoras opcionales

El cambio opcional propuesto puede mejorar tu nota hasta 1 punto extra, pero solo si tu nota mínima en la parte obligatoria es de al menos 7.