



## ACTIVITAT AVALUABLE

Programació  
CFGS DAW

Autors:

Joan V. Cassany – [jv.cassanycoscolla@edu.gva.es](mailto:jv.cassanycoscolla@edu.gva.es)

Guillermo Garrido – [g.garridoportes@edu.gva.es](mailto:g.garridoportes@edu.gva.es)

2022/2023

### Llicència



**[CC BY-NC-SA 3.0 ES](https://creativecommons.org/licenses/by-nc-sa/3.0/es/) Reconeixement – No Comercial – Compartir Igual (by-nc-sa)**

No es permet un ús comercial de l'obra original ni de les possibles obres derivades, la distribució de les quals s'ha de fer amb una llicència igual a la que regula l'obra original.

## ‘AMPLIACIÓ ADDRESSAPP’

Arxiu

Editar

Ajuda

Nom

Cognoms

Detalls de perfil

Nom

Cognoms

Domicili

Ciutat

Codi Postal

Data de Naixement

Tabla sin contenido

Nou

Editar

Eliminar



## 1. INTRODUCCIÓ

L'objectiu d'aquest exercici consisteix principalment a fer-vos utilitzar el conjunt de conceptes apresos, tant durant el tema d'Interfícies Gràfiques, com en el tema de Fitxers.

Per tant, en aquest exercici es donen especificacions molt concretes (sense indicar la solució) perquè s'utilitzin les instruccions i estructures Java adequades. És molt important que les seguïu, ja que la rubrica contemplarà si es fa ús d'aquestes o no.

Igualment, donada la complexitat de les dependències, és important que vos ajusteu a la nomenclatura específica que es proporciona al llarg de l'exercici.

## 2. DESCRIPCIÓ DE LA AMPLIACIÓ

Com s'ha pogut comprovar durant la realització de l'aplicació **UF12AddressApp**, hi ha diversos botons i opcions de menú, que no s'han implementat.

Per aquest motiu, la pràctica avaluable consistirà a acabar l'aplicació seguint els passos que s'indiquen més endavant.

S'ha de tindre molt en compte que la realització d'aquesta pràctica depèn del bon funcionament de l'aplicació **UF12AddressApp**, per la qual cosa és molt recomanable no modificar cap element de la guia proporcionada abans de començar aquesta pràctica.

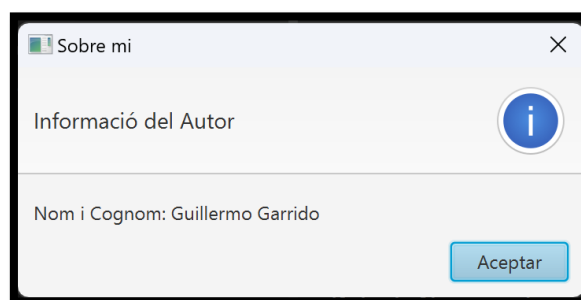
## 3. PASSOS A SEGUIR

**3.1.** S'ha de crear una funció per a mostrar un quadre de diàleg on mostrar la informació de l'autor.

**3.1.1.** Es crearà la funció en el controlador que conte el menú de l'aplicació.

**3.1.2.** S'assignarà a l'opció del menú "Sobre mi" la funció a cridar.

Aquesta imatge podria ser un exemple a seguir, amb el nom de l'alumne que l'està realitzant:



Editar i crear nou contacte necessiten un poc més d'elaboració.

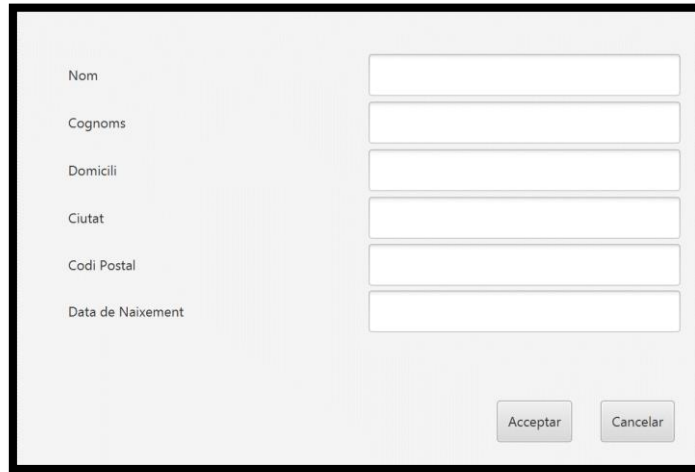
**3.2.** Codificació d'un nou quadre de diàleg que permetrà crear i editar.

**3.2.1.** Crea un nou arxiu FXML anomenat ContactEditDialog.fxml dins del paquet view.

**3.2.2.** Crea un controlador per a aquesta vista nomenat ContactEditDialogController.java.

3.2.3. A la vista `ContactEditDialog.fxml` (que conte un `AnchorPane`), afegeix un panell de graella (`GridPane`) amb unes mesures de 560 d'ample i 300 d'altura. Deixant 20px de marge dalt, a la dreta i a l'esquerra del `AnchorPane`.

3.2.4. En el panell de graella afig etiquetes (`Label`), camps de text (`TextField`) i botons (`Button`) per a crear una finestra de diàleg com la següent:

Aquesta imatge mostra un disseny d'interfície d'usuari per a una finestra de diàleg. A l'interior d'un rectangle gris clar, hi ha una columna de labels a l'esquerra: "Nom", "Cognoms", "Domicili", "Ciutat", "Codi Postal" i "Data de Naixement". A la dreta de cada label hi ha un camp de text blanc. A la part inferior dreta, hi ha dos botons: "Acceptar" i "Cancelar".

3.2.5. Els botons tindran un marge intern de 10px i el seu contenidor estarà situat a 20px de dreta i 20px de la part inferior del `AnchorPane`. I canviem el nom dels botons a "Acceptar" i "Cancel·lar"

3.2.6. S'ha de afegir estil als seus elements utilitzant la fulla d'estils de l'aplicació. Per tant, la vista ha de quedar de la següent manera:

Aquesta imatge mostra el mateix disseny d'interfície d'usuari que l'anterior, però amb un estil aplicat. El fons de la finestra és de color fosc. Els labels i els camps de text són blancs, mentre que els botons "Acceptar" i "Cancel·lar" són de color gris clar amb un efecte de relleu.

3.2.7. Per a poder capturar els elements del formulari, en `ContactEditDialogController.java` es crearan **variables per a cada `TextField`** i s'assignaran a cada `TextField` de la vista.

3.2.8. A més a més, hem de tindre un objecte de tipus `Stage` (`dialogStage`), una variable de tipus `boolean` (`acceptClicked`) inicialitzada a `falsa` i un objecte de tipus `Contact` (`contacte`).

3.2.9. Crearem una funció `set` per a modificar l'objecte `dialogStage` que tindrà el nom de **`setDialogStage()`**.

3.2.10. Crearem una funció `get` per a obtenir el valor de la variable `acceptClicked` que tindrà el nom de **`getAcceptClicked()`**.

**3.2.11.** Crearem una funció **loadContacte()** que tindrà un paràmetre d'entrada de tipus **contacte**. Aquest obtindrà el **contacte** a partir del que està seleccionat per l'usuari en la vista **Index.fxml**.

El primer que farà aquesta funció és inicialitzar la variable **contacte** de la classe amb el **contacte** que li aplega com a paràmetre.

A continuació, omplirà tots els **TextFields** amb els atributs de **contacte**.

**3.2.12.** Una altra funció que cal crear per a poder tancar el quadre de diàleg és la funció **cancel()** i l'enllaçarem al botó amb el text "**Cancel·lar**". Aquesta funció sols utilitzarà la variable **dialogStage** per a tancar el quadre de diàleg.

**3.2.13.** Ara s'ha de crear al **ContactEditDialogController.java** la funció **ok()** i l'enllaçarem al botó amb el text "**Acceptar**".

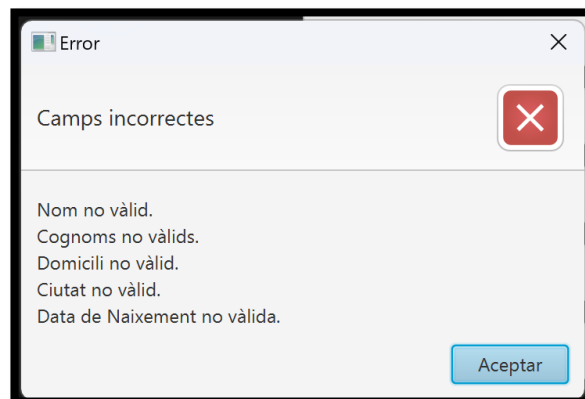
**3.2.14.** En la funció **ok()** primer es valida l'entrada de l'usuari mitjançant l'execució del mètode **areFormInputsValid()** que retornarà un valor boolean.

Si la funció **areFormInputsValid()** retorna **true** la funció **ok()** modificarà els valors dels atributs de **contacte** amb el text que existeix en els **TextFields**, canviarà el valor de **acceptedClicked** a **verdader** i tancarà el quadre de diàleg.

**3.2.15.** La funció **areFormInputsValid()** crearà una variable **missatge** que anirà incrementant-se amb un missatge d'error en cada camp sols quan es complisquen alguna de les següents condicions:

- Comprovarà que el text de cada **TextField** siga **null**
- Comprovarà que el nombre de caràcters siga **0**.
- En cas de ser un valor diferent de cadena s'haurà de validar que el tipus de valor siga **correcte**.

Finalment, en cas que el **missatge continga** algun error, la funció **areFormInputsValid()**, a més de retornar **false**, mostrarà un quadre de diàleg que informará el usuari dels errors.



En cas que no hi haja errors retornarà un **true**.

### 3.3. Funcionalitat des de `indexController.java`

3.3.1. Per a poder connectar el boto “Nou” necessitem crear una funció `newContact()` sense paràmetres d’entrada. Aquesta funció crearà una instància de `contacte (tempContact)` utilitzant el seu constructor sense valors.

Seguidament, assignarà el valor de la variable interna amb el nom `acceptClicked` utilitzant la instància de `address_app` que té, per a cridar al mètode `showContactEditDialog()` a la que li passarà la instància de `contacte` creada anteriorment com a argument.

A continuació, comprovarà si `acceptClicked` té el valor de vertader. En eixe cas, i fent ús de la instància de `address_app`, obtindrà els `contactes` amb `getContactes()` i li afegirà el nou `contacte tempContact`.

3.3.2. Per altra banda, per a poder connectar el boto “Editar” necessitem crear una funció `editContact()` sense paràmetres d’entrada. Aquesta funció crearà una variable de tipus `contacte (selectedContact)` que s’obtindrà mitjançant el mètode `getSelectionModel().getSelectedItem()` de `contact_table`.

Seguidament, es comprovarà que `selectedContact` no es nul·la i s’assignarà el valor de la variable interna amb el nom `acceptClicked` utilitzant la instància de `address_app` que té, per a cridar al mètode `showContactEditDialog()` a la que li passarà la instància de `contacte` seleccionat anteriorment com a argument.

A continuació, comprovarà si `acceptClicked` té el valor de vertader. En aquest cas es cridarà a la funció interna `showContactDetails()` passant-li el `contacte`, perquè mostre els detalls del `contacte`.

A més, si es comprova que `selectedContact` és nul·la, es mostrarà un missatge d’error amb un quadre de diàleg.

3.4. Ara que ja està el quadre d’edició i el seu codi necessitem que es pugui obrir i passar-li el `contacte` a carregar.

3.4.1. Anem a `UF12AddressApp.java` i afegim la funció `showContactEditDialog()`.

3.4.2. Després, carreguem la vista per al quadre de diàleg (Igual que fem en `indexShow()`).

3.4.3. Crearem un `Stage (dialogStage)` i farem una nova instància d’aquest.

3.4.4. Comprovarem si el nom del `contacte` es `cadena buida ("")`. De ser així li posarem al `Stage` el títol de “Nou `contacte`”. En cas de no ser igual posarem el títol de “`Editar NOMCONTACTE COGNOMSCONTACTE`”.

3.4.5. Utilitzarem les funcions `initModality(Modality.WINDOW_MODAL)` i `initOwner(primaryStage)` de la instància de `Stage`, per a indicar el tipus de quadre de diàleg a obrir i enllaçar el `Stage primari` sobre qui volem obrir el quadre de diàleg.

3.4.6. Crearem una nova escena a la qual li passarem el `AnchorPane` que s’ha creat al carregar la vista.

3.4.7. Ara s’utilitzarà el mètode `setScene()` per a passar-li l’escena que s’ha creat a la instància de `Stage`.

3.4.8. Per últim, sols queda enviar les dades del contacte al quadre de diàleg. Per a poder fer-ho es crearà una instància de **ContactEditDialogController** amb el **loader.getController()**.

3.4.9. Li passarem al controller, la instància de **Stage** que hem creat anteriorment amb el mètode **setDialogStage()**.

3.4.10. Li passarem al controller, la instància de **Contacte**, que ens ha arribat com a paràmetre d'entrada, amb la funció **loadContacte()**.

3.4.11. Obrirem el quadre de diàleg i esperarem.

3.4.12. Per últim, retornarem en la funció un valor booleà que obtindrem en la funció **getAcceptClicked()** del controlador.

3.4.13. En cas de produir-se qualsevol excepció retornarem el valor fals.

#### 4. Exemple de execució

