Développement d'une application de gestion de tâches personnelles en javaFX

Contexte :

Dans la vie de chacun d'entre nous, il est très utile de pouvoir organiser ses activités, ses tâches à faire dans la journée ou dans les jours à venir.

L'objectif de ce projet sera de concevoir une application de bureau en utilisant javaFX ergonomique et fonctionnelle permettant ainsi à chaque utilisateur de bien organiser sa liste de tâches.

Objectifs de l'application :

Cette application propose les fonctionnalités suivantes :

- Créer et supprimer des tâches.
- Modifier des tâches existantes.
- Marquer une tâche comme terminée ou non terminée (soit la cocher ou la barrer).
- Indiquer le pourcentage d'avancement dans une tâche.
- Filtrer l'affichage des tâches selon leur état (toutes, en cours, terminées).
- Offrir une interface simple et intuitive.

Fonctionnalités bonus (non obligatoires) :

- Authentification (avec une page d'inscription et de login).
- Connexion avec une base de données.
- Notifications systèmes (alertes pour les tâches).
- Gestion multi-utilisateur.

Description fonctionnelle détaillée :

Gestion des tâches :

1. Ajouter/ créer une tâche : L'utilisateur doit pouvoir ajouter une tâche en remplissant un formulaire par exemple (le titre de la tâche, sa description, date de création, date de fin de la tâche, priorité). Il faut créer des champs obligatoires comme le titre et la date de fin et si ces champs ne seront pas remplis on peut générer un message d'erreur.
2. Modifier une tâche : L'utilisateur peut modifier les valeurs des champs de chaque tâche.
3. Supprimer une tâche : L'utilisateur peut supprimer une tâche. (Une confirmation de suppression est recommandée.)
4. Marquer une tâche comme terminée : L'utilisateur peut changer l'état d'une tâche de non réalisée à terminée (le changement peut se faire soit avec une case à

cocher ou un bouton, la tâche terminée doit être différente donc soit barrée, soit colorée.)

Affichage des tâches :

Les tâches doivent être affichées dans une ListView qui affichent le titre, la description ainsi que tous les détails de la tâche.

Les tâches terminées doivent être distinguées visuellement.

Filtrage de tâches :

L'utilisateur peut filtrer la liste affichée selon plusieurs critères (toutes, actives, terminées).

Installation javaFX :

Dans un premier temps, il vous faut pouvoir utiliser JavaFX, car cette librairie est désormais dissociée du JDK depuis Java 11. En principe, votre IDE intègre déjà javaFX. Pour le vérifier, soit vous avez la possibilité de créer directement un nouveau projet JavaFX, soit, depuis votre projet existant, l'instruction :

```
import javafx.application.Application;
```

Si ça ne compile pas directement, il vous faut dans un premier temps télécharger JavaFX depuis le projet officiel : https://openjfx.io/

Installation sur VS Code :

1. Installer les extensions : Extension Pack for Java (Microsoft) , JavaFX Support (optionnel mais utile).
2. Configurer le projet : Méthode simple (sans Maven) : créer un projet java, ajouter JavaFX au classpath.

3. Ajouter les VM options (très important) : Dans .vscode/launch.json, on ajoute :

```
</> JSON
{
  "version": "0.2.0",
  "configurations": [
    {
      "type": "java",
      "name": "Launch JavaFX",
      "request": "launch",
      "mainClass": "com.example.Main",
      "vmArgs": "--module-path C:/javafx-sdk-21/lib --add-modules
javafx.controls,javafx.fxml"
    }
  ]
}
```

Remplacer le chemin dans VmArgs par le vôtre.

Première application JavaFX :

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.stage.Stage;

public class Main extends Application {
@Override
public void start(Stage stage) {
Button btn = new Button("Hello JavaFX");
Scene scene = new Scene(btn, 300, 200);
stage.setScene(scene);
stage.show();
}

public static void main(String[] args) {
launch();
}
}
```

import javafx.application.Application; → Importe la classe Application qui est la classe de base obligatoire pour toute application javaFX qui gère le cycle de vie de l'application (démarrage, arrêt etc.).

import javafx.scene.Scene ; → Une scène c'est le contenu de la fenêtre (comme une page).

import javafx.scene.control.Button;→ Importe le composant bouton qui permet de créer un bouton.

import javafx.stage.Stage; → importe Stage qui est la fenêtre principale.

public class Main extends Application { → Tu crées une classe Main et extends Application qui est obligatoire pour JavaFX.

La classe principale qui va lancer l'affichage (celle contenant un main) doit hériter de la classe Application. Appelons la MainGui. Hériter de cette classe vous oblige à implémenter la méthode

@Override
public void start(Stage stage) { → start() est le point d'entrée de JavaFX (équivalent de main mais pour la UI.)

Stage stage → la fenêtre fournie automatiquement.

Elle prend en paramètre un objet de type Stage, représentant la fenêtre de votre application. Vous pouvez ainsi modifier la taille de la fenêtre grâce aux méthodes setWidth et setHeight de cet objet, ainsi que le titre, avec la méthode setTitle.

Button btn = new Button("Hello JavaFX"); → crée un bouton avec le texte « Hello JavaFX ».

Scene scene = new Scene(btn, 300, 200); → Création de la scène qui contient ce qu'on affiche dans la fenêtre.

stage.setScene(scene); → associe la scène à la fenêtre.

stage.show(); → Rend la fenêtre visible.

Autres attributs :

1) Zone de texte permettant de saisir du texte :

- un objet de type TextArea appelé textToSend

2) Zone de texte affichant les messages reçus :

- un objet de type ScrollPane appelé scrollReceivedText
- un objet de type TextFlow appelé receivedText

3) Bouton permettant d'envoyer du texte

- un objet de type Button appelé sendBtn

4) Bouton permettant d’effacer la zone de saisie

- un objet de type Button appelé clearBtn