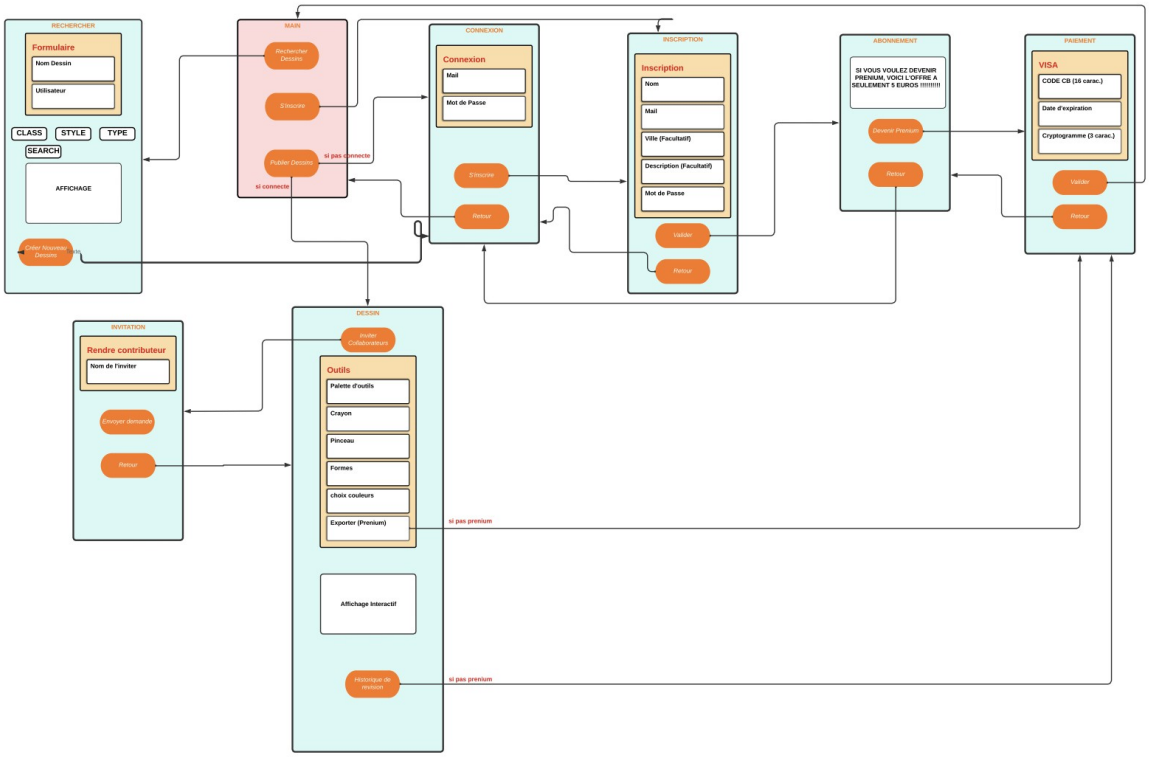
HUNOT-MARTIN Alaric
FLAMENT Théo
M1 Imagine
Université Montpellier

Compte Rendue Projet Mobile HAI811

Groupe 30 HMAFT : Sujet Image

Architecture de l’application idéale



Concernant l'avancement du projet, nous nous sommes pour l'instant concentrés essentiellement sur la création du formulaire d'inscription et de connexion, ainsi que sur la partie conception du dessin pour obtenir un rendu intéressant. Des ébauches concernant la gestion de l'enregistrement des dessins ont été faites, mais nous n'avons pas encore réussi à obtenir de résultats. Il s'agira de la prochaine grosse étape dans l'avancement du projet.

MainActivity

Ce code représente la logique de l'activité principale (`MainActivity`) de l'application de dessin. Voici ce qu'il accomplit :

1. Il lie les éléments de l'interface utilisateur définis dans le fichier XML (`activity_main.xml`) aux variables correspondantes dans le code Java à l'aide de la méthode `findViewById`.
2. Il définit des écouteurs de clic pour chaque bouton. Lorsque l'un des boutons est cliqué, il démarre une nouvelle activité correspondante en utilisant des intents.
3. Plus spécifiquement :

    - Le bouton "S'inscrire" (`inscrireButton`) démarre l'activité `SignUpActivity`.
    - Le bouton "Créer dessin" (`creerDessinButton`) démarre l'activité `DrawingActivity`.
    - Le bouton "Rechercher dessin" (`searchButton`) démarre l'activité `SearchActivity`.

Sur le plan du rendu :

- Les boutons ont un style défini par `@style/Widget.AppCompat.Button.Colored`, ce qui leur donne un aspect visuel cohérent avec le thème de l'application.
- Le fond des boutons est un dégradé défini par `@drawable/mondrawable`, ce qui leur donne un aspect esthétique attractif.

DrawingActivity

XML (activity_drawing.xml) :

1. Layout général :
    - Le layout est un RelativeLayout, qui permet de positionner les éléments de manière relative les uns aux autres.
    - Le fond (`android:background`) est défini comme "@drawable/cbo", indiquant probablement une image de fond.
2. Spinner pour l'épaisseur du trait (spinner_thickness) :
    - Il permet à l'utilisateur de sélectionner l'épaisseur du trait parmi plusieurs options.
    - Lorsqu'une option est sélectionnée, cela affecte la largeur du trait dans la vue de dessin.
3. Spinner pour la couleur du trait (spinner_color) :
    - Similaire au Spinner précédent, mais permet de choisir la couleur du trait.
4. Boutons :
    - `button_clear` : Efface le dessin.
    - `button_back` : Retourne à l'activité précédente.
    - `button_save` : Censé enregistrer le dessin, mais selon la description, il envoie seulement un message de confirmation sans enregistrer réellement le dessin.
5. Vue de dessin (DrawingView) :
    - C'est une vue personnalisée qui hérite de la classe `View`.
    - C'est où l'utilisateur dessine.
    - L'épaisseur et la couleur du trait sont contrôlées à l'aide des Spinners et sont affichées en conséquence.

Classe DrawingActivity :

1. Initialisation des composants et gestion des événements :
    - Les composants de l'interface utilisateur sont initialisés dans `onCreate()`.
    - Des écouteurs d'événements sont configurés pour les boutons et les Spinners pour répondre aux actions de l'utilisateur.
2. Gestion des Spinners :
    - Lorsque l'utilisateur sélectionne une option dans les Spinners, les attributs `currentColor` et `currentThickness` sont mis à jour en conséquence.
3. Méthode saveDrawing() :
    - Censée enregistrer le dessin en tant qu'image.
    - Crée un bitmap à partir de la vue de dessin, mais il semble qu'il y ait un problème dans la sauvegarde réelle du fichier.

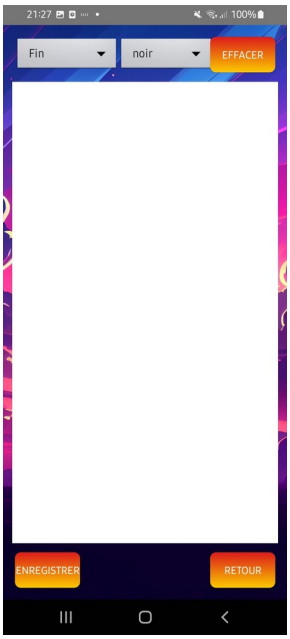
Classe DrawingView :

1. Initialisation et configuration du pinceau :
    - Les paramètres du pinceau, y compris la couleur et l'épaisseur, sont initialisés dans `setupPaint()`.
2. Gestion du dessin :
    - L'utilisateur dessine sur cette vue.
    - L'épaisseur du trait est contrôlée par l'utilisateur via les Spinners.
3. Méthode clearDrawing() :
    - Efface le dessin actuel en supprimant les chemins, les couleurs et les épaisseurs de trait.

Fonctionnalités manquantes ou problèmes :

- L'enregistrement du dessin ne semble pas fonctionner correctement. Bien qu'un message de confirmation soit affiché, il semble qu'il y ait un problème lors de la sauvegarde réelle du dessin en tant qu'image.

En résumé, cette activité permet à l'utilisateur de dessiner sur l'écran en choisissant la couleur et l'épaisseur du trait, mais l'enregistrement du dessin ne fonctionne pas correctement.



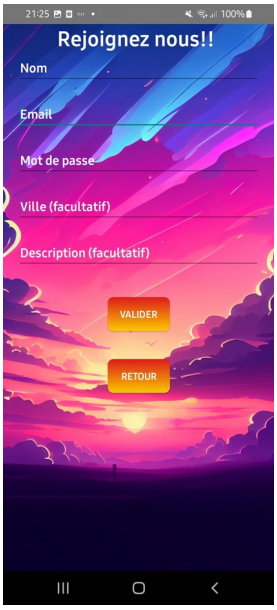
SignUpActivity

Ce code XML représente le layout de l'activité d'inscription (`SignUpActivity`). Il contient différents éléments d'interface utilisateur, tels que des TextViews, des EditTexts et des Buttons, disposés dans un ConstraintLayout. Voici une explication étape par étape :

1. `TextView "Rejoignez nous!!"` : Affiche un message de bienvenue.
2. `EditText pour le nom, l'email, le mot de passe, la ville et la description` : Permet à l'utilisateur de saisir son nom, son email, son mot de passe, sa ville et une description optionnelle.
3. `Button "Valider"` : Permet à l'utilisateur de valider ses informations d'inscription et de passer à l'étape suivante.
4. `Button "Retour"` : Permet à l'utilisateur de retourner à l'écran de connexion (`MainActivity`).

Le fichier Java associé (`SignUpActivity.java`) définit le comportement des boutons. Lorsque l'utilisateur appuie sur le bouton "Valider", il est redirigé vers l'écran de connexion

(`LoginActivity`). Lorsqu'il appuie sur le bouton "Retour", il revient à l'écran d'accueil (`MainActivity`).



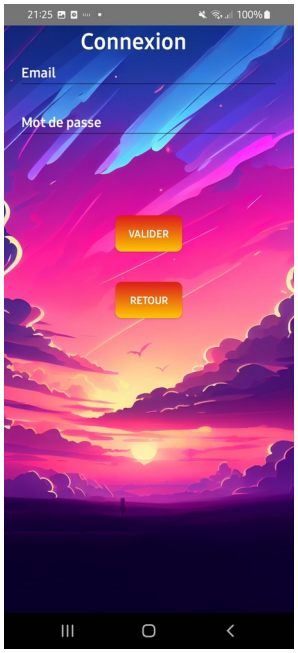
LoginActivity

Ce code XML représente le layout de l'activité de connexion (`LoginActivity`). Il comprend des éléments d'interface utilisateur tels que des TextViews, des EditTexts et des Buttons, disposés dans un ConstraintLayout. Voici une explication étape par étape :

1. `TextView "Connexion"` : Affiche un titre pour l'écran de connexion.
2. `EditText pour l'email et le mot de passe` : Permet à l'utilisateur de saisir son email et son mot de passe.

3. `Button "Valider"` : Permet à l'utilisateur de valider ses informations de connexion et de passer à l'étape suivante.
4. `Button "Retour"` : Permet à l'utilisateur de revenir à l'écran précédent (peut-être l'écran d'accueil ou l'écran d'inscription).

Le fichier Java associé (`LoginActivity.java`) définit le comportement du bouton "Valider". Lorsque l'utilisateur appuie sur ce bouton, il est redirigé vers l'écran de la page d'adhésion Premium (`PremiumMembershipActivity`).



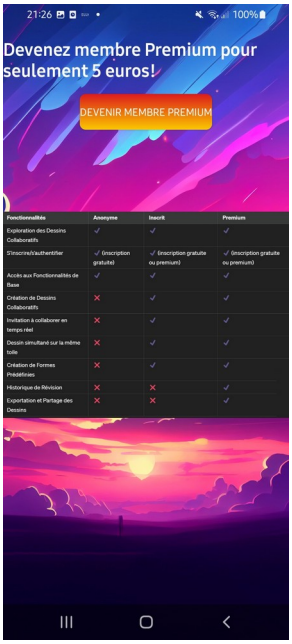
PremiumMembershipActivity

Ce code XML définit le layout de l'activité PremiumMembership. Voici une explication étape par étape :

1. `TextView "Devenez membre Premium"` : Affiche un titre incitant l'utilisateur à devenir membre premium. Le texte est centré horizontalement et a une taille de police de 30sp. Il est en gras et de couleur blanche. Il est également centré verticalement.
2. `Button "Devenir membre Premium"` : Un bouton permettant à l'utilisateur de devenir membre premium. Il est centré horizontalement et positionné en dessous du TextView. Il a un fond personnalisé et un texte en blanc.
3. `ImageView avec une image de perroquet` : Affiche une image de perroquet. La source de l'image est définie par le fichier drawable "table.png". La description du contenu de l'image est "Image de perroquet".

Le fichier Java associé (`PremiumMembershipActivity.java`) configure l'activité. Il utilise EdgeToEdge pour gérer les bords de l'écran et ajuster le padding en conséquence.

Ce layout présente une invitation claire à devenir membre premium, avec un titre accrocheur et un bouton d'action bien visible. L'image de perroquet peut être un élément décoratif ou illustratif ajoutant de l'intérêt visuel à l'écran.



SearchActivity

Ne Fonctionne pas encore servira pour enregistrer les dessins