DAA – Étude de cas
App Widgets

January 10, 2025

Émilie Bressoud **Sacha Butty** **Loïc Herman**

Introduction

Les widgets sont des composants essentiels de l'écosystème Android, permettant aux utilisateurs d'accéder rapidement à des informations ou des fonctionnalités sans ouvrir l'application complète. Ce projet explore la création d'un widget Android moderne en utilisant les dernières technologies recommandées par Google, notamment Jetpack Compose et Jetpack Glance.

L'objectif est de créer un widget responsive et interactif, en suivant les bonnes pratiques de développement Android. Nous verrons comment configurer, implémenter et personnaliser un widget pour offrir une expérience utilisateur optimale.

Conception

Architecture

L'architecture du widget se compose de plusieurs éléments essentiels:

- Le fichier de configuration XML
- La classe d'implémentation du widget
- Le layout du widget

Mise en place d'un widget Android

Configuration du widget

La configuration d'un widget se fait principalement via un fichier XML qui définit:

- Les dimensions du widget
- Les options de configuration et redimensionnement

```xml
<!-- app/src/main/res/xml/my_app_widget_info.xml -->
<?xml version="1.0" encoding="utf-8"?>
<appwidget-provider xmlns:android="http://schemas.android.com/apk/res/android"
        android:initialLayout="@layout/glance_default_loading_layout"
        android:widgetFeatures="configuration_optional|reconfigurable"
        android:targetCellWidth="3"
        android:targetCellHeight="1"
        android:minResizeHeight="50dp"
        android:minResizeWidth="50dp"
        android:minHeight="50dp"
        android:minWidth="50dp"
        android:resizeMode="horizontal|vertical"
        android:description="@string/widget_picker_description">
</appwidget-provider>
```

- `tagetCellWidth` et `targetCellHeight`: Taille du widget en nombre de cellules. Elle permet de définir la taille du widget par défaut en fonction de la grille de l'écran d'accueil.
- `minHeight` et `minWidth`: Taille minimale du widget.
- `minResizeHeight` et `minResizeWidth`: Taille minimale du widget après redimensionnement.
- `resizeMode`: Mode de redimensionnement du widget. Il faut le définir pour permettre à l'utilisateur de redimensionner le widget.
  - il est possible de rendre le widget redimensionnable uniquement horizontalement ou verticalement ou les deux en utilisant `horizontal` ou `vertical`.
- `description`: Description du widget affichée dans le sélecteur de widgets.

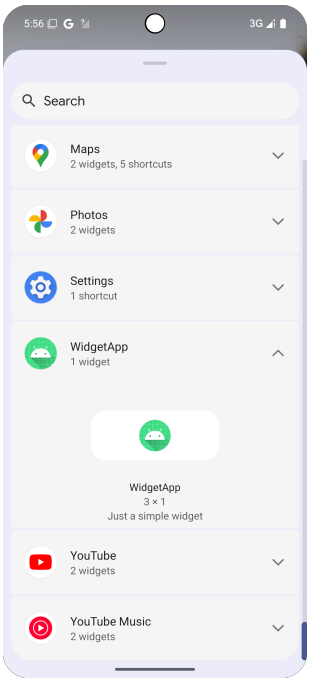

Fig. 1. – Example of a widget configuration

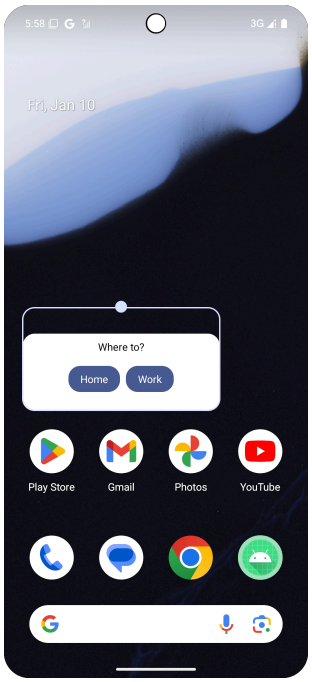

Fig. 2. – Example of a small widget

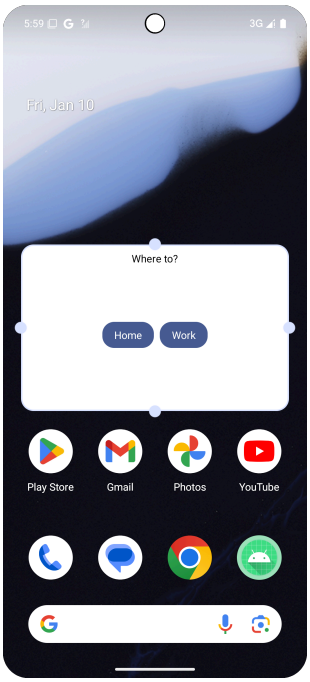

Fig. 3. – Example of a large widget (resized)

Manifest

Comme pour une activité, il est nécessaire de déclarer le widget dans le fichier `AndroidManifest.xml`:

```
<!-- app/src/main/AndroidManifest.xml -->
<receiver
      android:name=".MyAppWidgetReceiver"
      android:icon="@mipmap/ic_launcher"
      android:exported="true"
      android:permission="android.permission.BIND_APPWIDGET">
  <intent-filter>
      <action android:name="android.appwidget.action.APPWIDGET_UPDATE" />
  </intent-filter>
  <meta-data
         android:name="android.appwidget.provider"
         android:resource="@xml/my_app_widget_info" />
</receiver>
```

- `android:name`: Nom de la classe qui gère le widget

Approches de développement

Comparaison entre XML et Glance

Dans ce rapport, nous nous concentrons sur l'implémentation moderne des widgets Android avec Jetpack Glance. Cependant, il est important de noter qu'il existe une approche plus traditionnelle utilisant XML. Voici quelques différences notables entre les deux approches:

Approche traditionnelle (XML)

- Utilisation de fichiers XML pour définir le layout
- AppWidgetProvider pour la définition du comportement du widget
- Possibilité de créer une `preview` dans la page d'ajout de widget à partir d'un layout
- Possibilité de linker un layout à une taille spécifique en utilisant la classe `RemoteViews`

Approche Glance

- Syntaxe déclarative similaire à Jetpack Compose
- Utilisation de GlanceAppWidgetReceiver pour la gestion du widget
- Possibilité de créer une `preview` à partir d'une image seulement
- Afin de pouvoir lier un layout à une taille spécifique, il faudra impérativement utiliser un layout XML, similaire à l'approche traditionnelle (classes `AndroidRemoteViews` et `RemoteViews`)

Utilisation avec Jetpack Compose et Jetpack Glance

Il est possible d'utiliser Jetpack Compose et Jetpack Glance pour créer des widgets. Pour cela, il faut ajouter différentes dépendances pour utiliser Android Glance.

```
dependencies {

    // For AppWidgets support
    implementation(libs.androidx.glance.appwidget)

    // For interop APIs with Material 3
    implementation(libs.androidx.glance.material3)

    // For interop APIs with Material 2
    implementation(libs.androidx.glance.material)
    (...)
}
```

Les 2 dernières ne sont pas obligatoires, mais elles permettent d'utiliser les composants Material 2 et Material 3 dans le widget.

Classe principale du widget

L'implémentation du widget nécessite une classe héritant de `GlanceAppWidgetReceiver` qui permet de gérer les événements du widget et de `GlanceAppWidget` qui permet de définir le layout du widget.

Layout du widget

SizeMode

Il existe trois modes de gestion de la taille des widgets:

SizeMode.Single

Mode par défaut qui indique qu'un seul type de contenu est fourni. Même si la taille disponible du widget change, le contenu ne s'adapte pas.

SizeMode.Responsive

Permet de définir un ensemble de mises en page réactives limitées par des tailles spécifiques. Pour chaque taille définie, le contenu est créé et mappé lors de la création ou de la mise à jour du widget. Le système sélectionne ensuite la meilleure correspondance en fonction de la taille disponible.

SizeMode.Exact

Ce mode demande le contenu du widget chaque fois que la taille disponible change (par exemple, lorsque l'utilisateur redimensionne le widget sur l'écran d'accueil). Il offre la plus grande flexibilité mais peut être plus coûteux en performances.

Il est possible d'afficher ou non certains éléments selon la taille du widget, si l'on décide de rendre le widget « responsive ». Par exemple, on peut afficher un seul bouton si le widget est petit et plusieurs boutons si le widget est plus grand.

On définit des tailles possibles du widget dans la classe principale du widget:

```
 companion object {
        private val SMALL_SQUARE = DpSize(100.dp, 100.dp)
        private val HORIZONTAL_RECTANGLE = DpSize(150.dp, 100.dp)
        private val BIG_SQUARE = DpSize(250.dp, 250.dp)
    }

    override val sizeMode = SizeMode.Responsive(
        setOf(
            SMALL_SQUARE, HORIZONTAL_RECTANGLE, BIG_SQUARE
        )
    )
```

Ensuite, on peut décider d'afficher ou non certains éléments en fonction de la taille du widget. Cela permet d'optimiser l'espace disponible et de « render » le widget parfaitement pour chaque taille.

```
@Composable
override fun Content() {
  Button(
    text = "Home",
    onClick = actionStartActivity<MainActivity>()
  )
  if (size.width >= HORIZONTAL_RECTANGLE.width) {
      Spacer(modifier = GlanceModifier.width(8.dp))
      Button(
          text = "Work",
          onClick = actionStartActivity<MainActivity>()
      )
  }
}
```

Ici, on affiche un bouton « Home » par défaut. Le bouton « Work » est affiché seulement si la largeur du widget est supérieure à la largeur définie.

Actions et interactions avec les widgets

Les widgets Android via Jetpack Glance offrent plusieurs mécanismes pour répondre aux interactions utilisateur.

Types d'actions disponibles

Le framework propose quatre types d'actions principales qui peuvent être attachées aux composants interactifs :

- `actionStartActivity` : Lance une activité spécifique
- `actionStartService` : Démarre un service en arrière-plan
- `actionSendBroadcast` : Émet un message broadcast
- `actionRunCallback` : Exécute une action personnalisée

Chacune de ces actions peut être déclenchée via le modificateur `GlanceModifier.clickable()` ou le paramètre `onClick` des composants interactifs.

Exemples d'implémentation

```
@Composable
fun ContenuWidget() {
   Column {
       // Lancement d'activité
       Button(
           text = "Dashboard",
           onClick = actionStartActivity<DashboardActivity>()
       )
```

```
        // Démarrage d'un service
        Image(
            provider = ImageProvider(R.drawable.sync),
            modifier = GlanceModifier.clickable(
                actionStartService<ServiceSync>(isForegroundService = true)
            ),
            contentDescription = "Synchroniser"
        )

        // Action personnalisée via callback
        Text(
            text = "Rafraîchir",
            modifier = GlanceModifier.clickable(
                actionRunCallback<ActionRafraichissement>()
            )
        )
    }
}
```

Paramètres d'actions

Les actions peuvent être enrichies avec des paramètres via l'API `ActionParameters` :

```
private val cleNavigation = ActionParameters.Key<String>("destination")

// Utilisation dans une action
actionStartActivity<NavigationActivity>(
    actionParametersOf(cleNavigation to "accueil")
)
```

Sur Android 12 et les versions ultérieures, le lancement d'activités depuis des lambdas exécutées dans un service est restreint. L'utilisation de `actionStartActivity` pour ces cas d'usage est recommandée.

Il est important de noter que les actions longues ou consommatrices de ressources devraient être déléguées à un `Worker` pour garantir une exécution fiable.

Limitations des widgets Android

Les widgets Android, bien que souvent décrits comme des « mini-applications », sont soumis à des contraintes spécifiques qui impactent leur conception et leur fonctionnement. La principale limitation concerne les gestes tactiles disponibles : seuls les touches simples et les slides verticaux sont supportés. Cette restriction est due à la nécessité de coexistence harmonieuse avec la navigation native de l'écran d'accueil, où les glissements horizontaux sont déjà réservés pour le défilement entre les pages.

Ces contraintes gestuelles ont des répercussions directes sur les éléments d'interface utilisateur pouvant être intégrés dans un widget. Certains composants UI standards, qui dépendent de gestes plus complexes, ne sont pas disponibles pour les widgets. Il est donc essentiel de prendre en compte ces limitations lors de la conception de widgets pour garantir une expérience utilisateur cohérente avec les directives de la plateforme Android.

Conclusion

Ce projet nous a permis d'explorer en profondeur le développement de widgets Android modernes. L'utilisation de Jetpack Glance, combinée à Jetpack Compose, offre une approche déclarative et intuitive pour créer des widgets interactifs et responsives.

Les principaux apprentissages incluent:

- La configuration et la déclaration correcte d'un widget dans le système Android
- L'utilisation des nouvelles APIs de Jetpack Glance pour une approche moderne du développement
- La gestion du responsive design pour s'adapter aux différentes tailles d'affichage
- L'implémentation d'interactions utilisateur fluides et efficaces

Cette expérience démontre l'évolution des outils de développement Android, rendant la création de widgets plus accessible tout en maintenant un haut niveau de personnalisation et de performance.

Bibliographie

[1] Google LLC, « Android Developers Documentation - App widgets ». [En ligne]. Disponible sur: https://developer.android.com/develop/ui/views/appwidgets/overview

[2] Google LLC, « Android Developers Documentation - Jetpack Glance ». [En ligne]. Disponible sur: https://developer.android.com/develop/ui/compose/glance

[3] Google LLC, « Android Developers Documentation - Jetpack Glance Interoperability ». [En ligne]. Disponible sur: https://developer.android.com/develop/ui/compose/glance/interoperability

Ce rapport a bénéficié de l'assistance d'intelligences artificielles génératives pour l'amélioration stylistique, la vérification orthographique et la structuration du contenu. Ces outils ont été utilisés comme support à la rédaction tout en préservant l'intégrité du contenu technique et des analyses originales. Le code en exemple reprend des extraits de documentation officielle et a été adapté pour illustrer les concepts abordés dans ce rapport.