République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Abdelhafid Boussouf University Center of Mila

3ème année Licence

Cours Applications Mobiles

Chapitre 3

Responsable Matière

Dr. MEGUEHOUT Hamza

L'application **MOBILE** diffère des applications **BUREAU**

Ex : l'interaction d'un utilisateur avec l'application !

ne commence pas toujours au même endroit …

Le parcours de l'utilisateur commence souvent de manière

non déterministe

Si vous ouvrez une application de messagerie à partir de votre écran d'accueil,

vous pouvez voir une liste d'e-mails

Si vous utilisez une autre application qui lance ensuite votre application de messagerie,

vous pouvez accéder directement à l'écran de l'application de messagerie pour composer un e-mail

Lorsqu'une application en appelle une autre

l'application appelante invoque une **ACTIVITÉ** dans l'autre application

plutôt que l'application dans son ensemble

Notion d'Activité

Composante fondamentale d'Android

Son rôle principal d'interagir avec l'utilisateur

Pour avoir deux écrans (un écran de sélection et un écran d'affichage des services).
Vous aurez généralement deux activités :

1. La **première** qui gère la partie sélection ;
2. La **seconde** qui gère l'affichage des services.

La plupart des applications contiennent **plusieurs** écrans

↓

Chaque **écran** de l'application représente **une activité**

↓

Elles comprennent **plusieurs activités**

(*une seule activité est lancée à la fois*)

En règle générale, une activité dans une application est spécifiée comme **activité principale**, qui est le **premier écran à apparaître** lorsque l'utilisateur lance l'application.

Chaque activité peut alors démarrer **une autre activité** afin d'effectuer des actions différentes.

Application de messagerie

↓

L'activité principale

↓

Boîte de ***réception de courrier électronique***

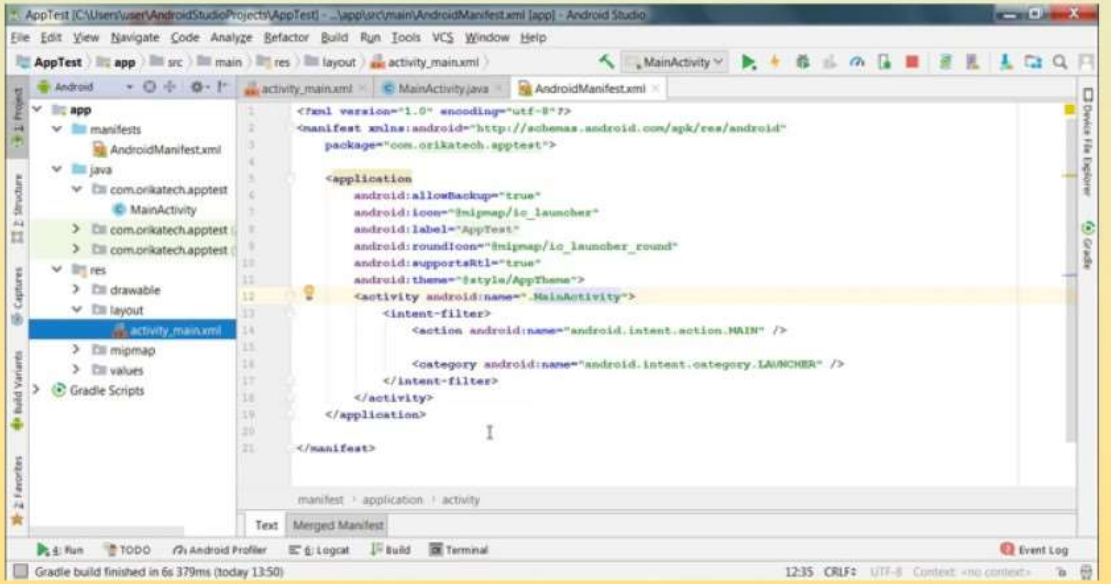
↓

Activité principale peut lancer d'autres activités qui fournissent des écrans

*la **rédaction d'e-mails**, l'**ouverture d'e-mails individuels**, etc.*

Pour utiliser des activités dans votre application

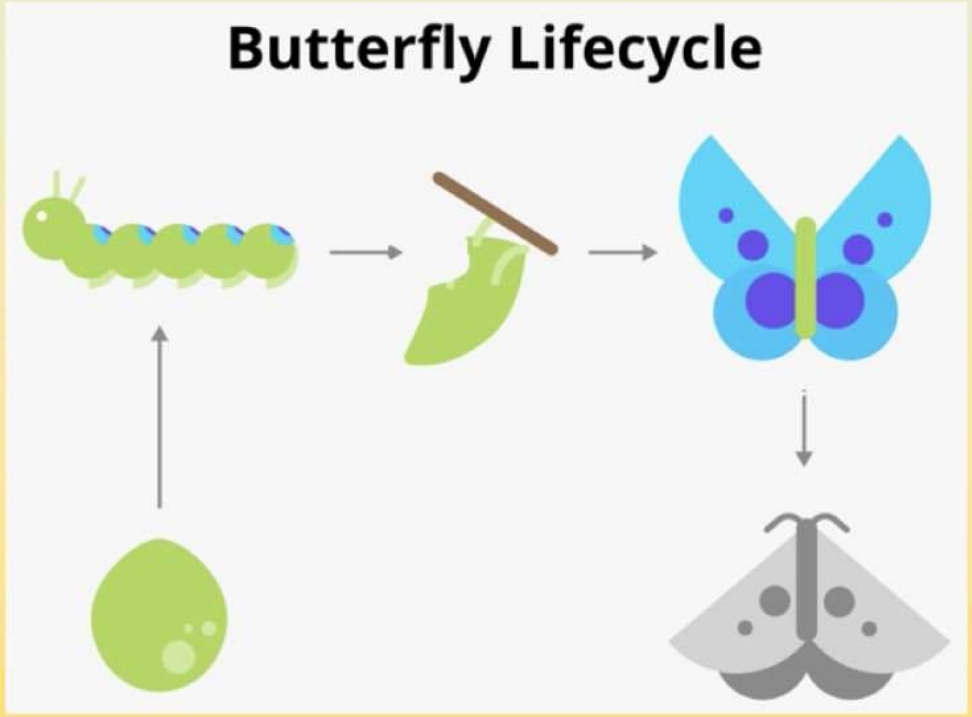
- Enregistrer des informations dans le **manifeste** de l'application
- Gérer les **cycles de vie** des activités de manière appropriée



Cycle de vie d'une activité

L'utilisateur a l'impression que l'écran qui lui est présenté existe aussi longtemps que l'application s'exécute

Mais dans la réalité, l'activité associée est peut-être **détruite** et **recréée** plusieurs fois durant la vie de l'application


Butterfly Lifecycle

Pourquoi ???

L'environnement Android est généralement plus limité en ressources qu'un ordinateur classique

Il est donc important de **limiter** au mieux **l'usage de ces ressources** (*batterie, mémoire, etc.*).

Pour cela, le système **Android gère un cycle de vie des activités** en **privilégiant l'économie de ressources**.

L'utilisateur **répond au téléphone** alors qu'une **activité s'exécute**

↓

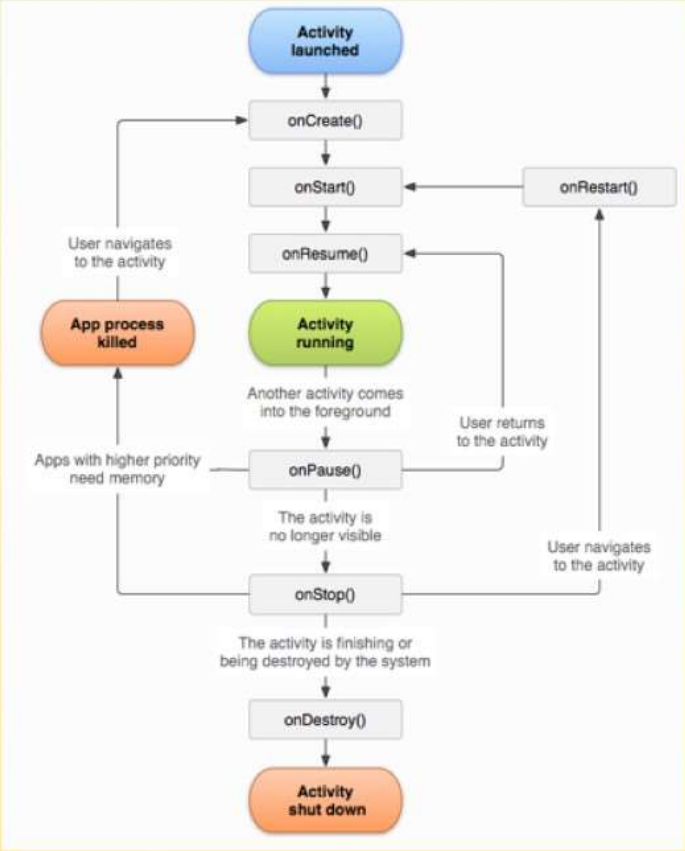
Elle est **SUSPENDUE** par le système et, si nécessaire, le système peut décider de la **SUPPRIMER** pour économiser des ressources

↓

Lorsque l'utilisateur voudra **reprendre** l'utilisation de l'application, l'activité sera intégralement **RECRÉÉE**

Ensemble des états qu'une activité peut être pendant toute sa durée de vie, de sa **création** à sa **destruction** et à la **récupération** de ses ressources par le système.

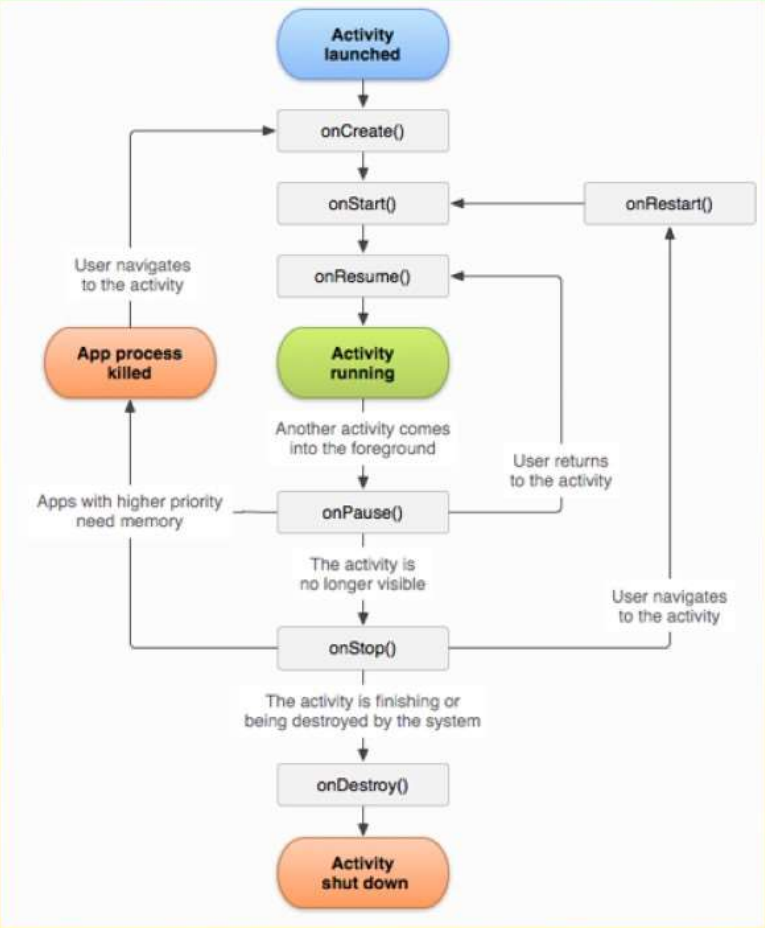
Lorsqu'un utilisateur navigue entre les activités de votre application (*ainsi que hors de celle-ci*), les activités **PASSENT** d'un état à un autre au cours de leur cycle de vie.



Application de consultation de mail → **activité** pour présenter la liste des mails.

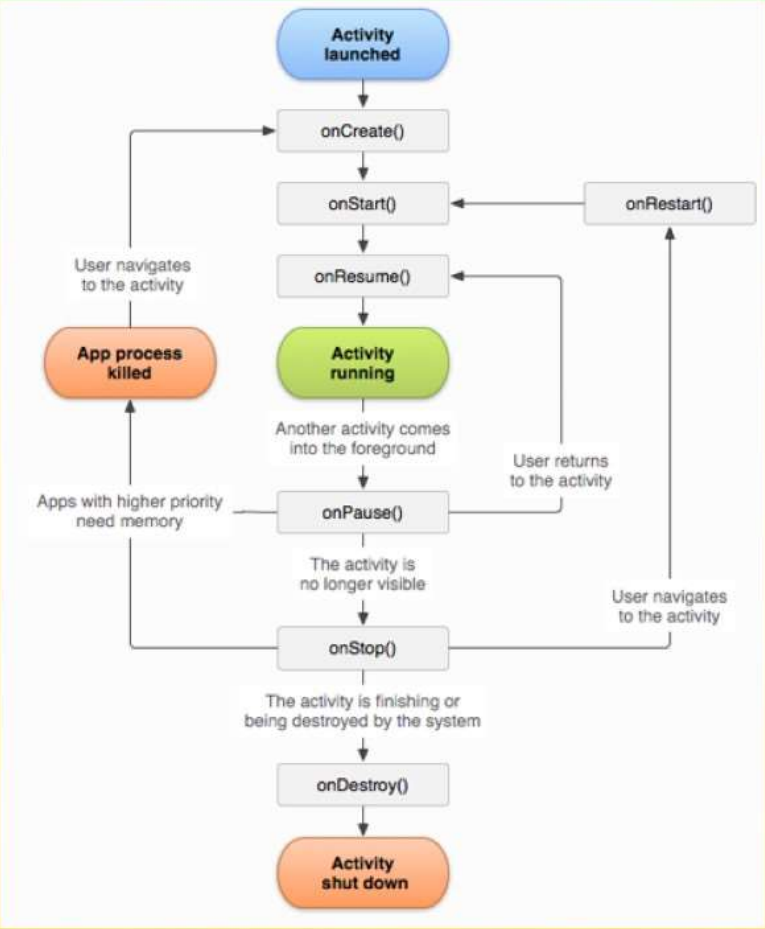
Activité est **CRÉÉE** par le système et ses méthodes **onCreate**, **onStart** et **onResume** sont appelées à la suite.

Activité est affichée à l'écran.



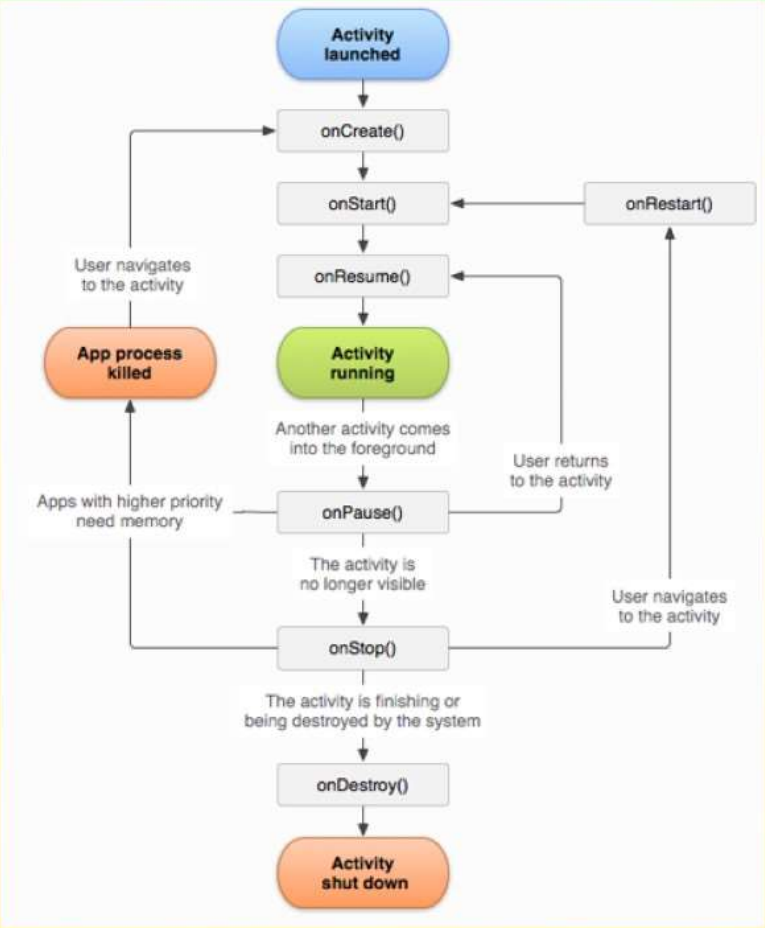
L'utilisateur clique sur un mail pour en consulter le détail → Cette action crée et affiche une nouvelle activité.

L'activité de **liste des mails** est **ARRÊTÉE** (onPause puis onStop sont appelées) → elle n'est plus au premier plan.



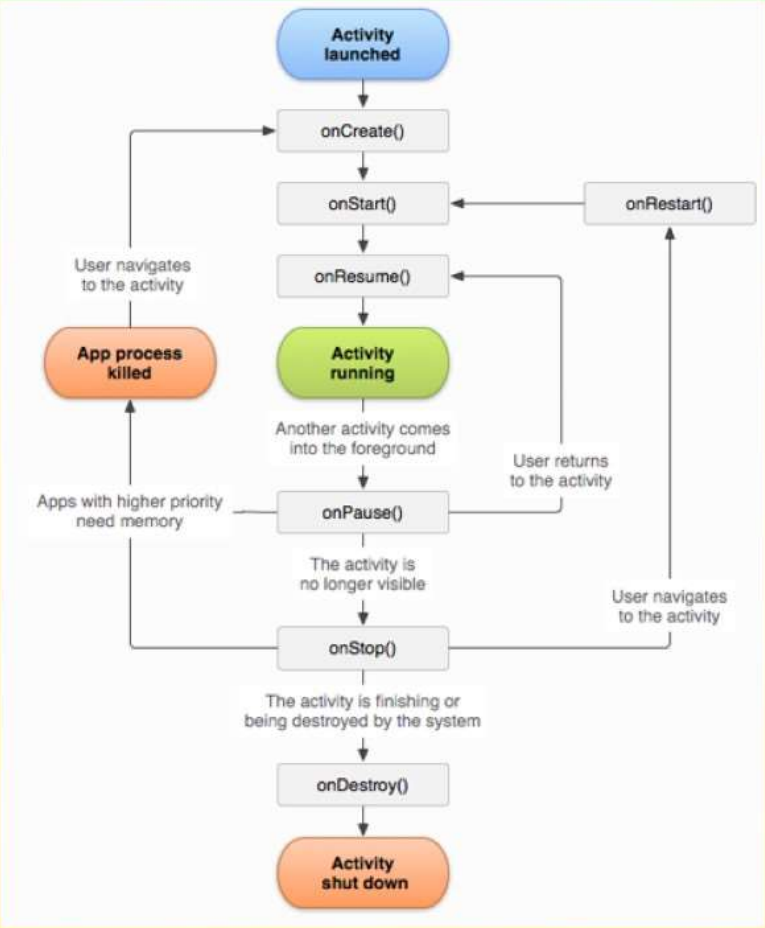
L'utilisateur appuie sur le **bouton Back** pour ramener la liste des mails.

L'activité doit être à **NOUVEAU AFFICHÉE** et ses méthodes **onRestart, onStart, onResume** sont appelées avant cela.



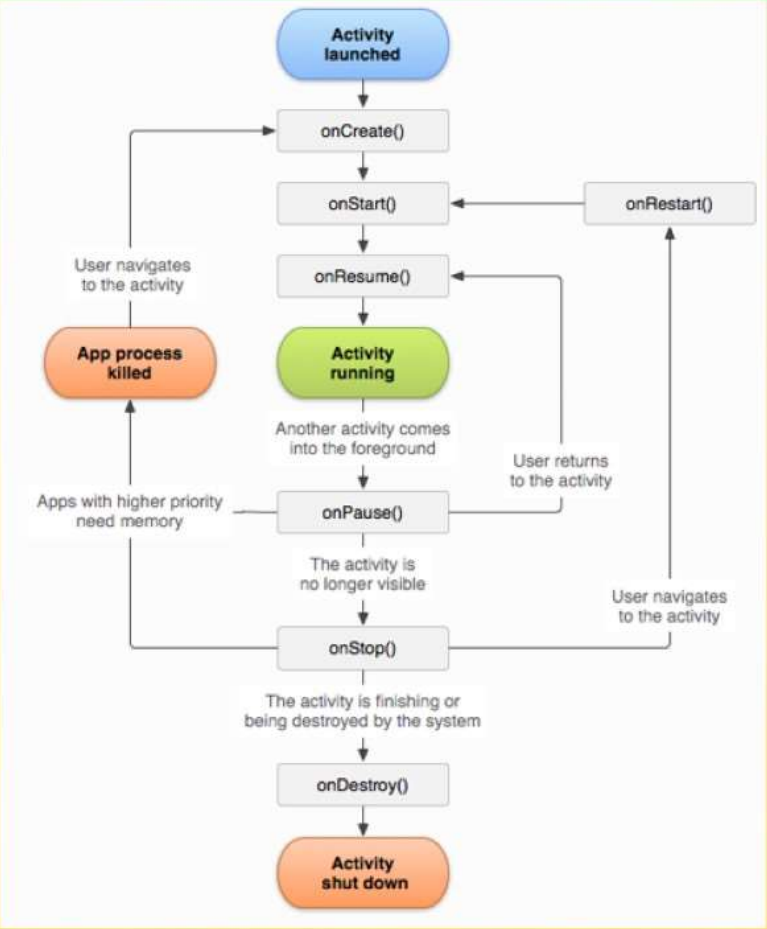
L'utilisateur **reçoit** un coup de téléphone et répond → Application de téléphonie en premier plan.

Activité de liste des mails est à nouveau **ARRÊTÉE** (**onPause**, **onStop**).



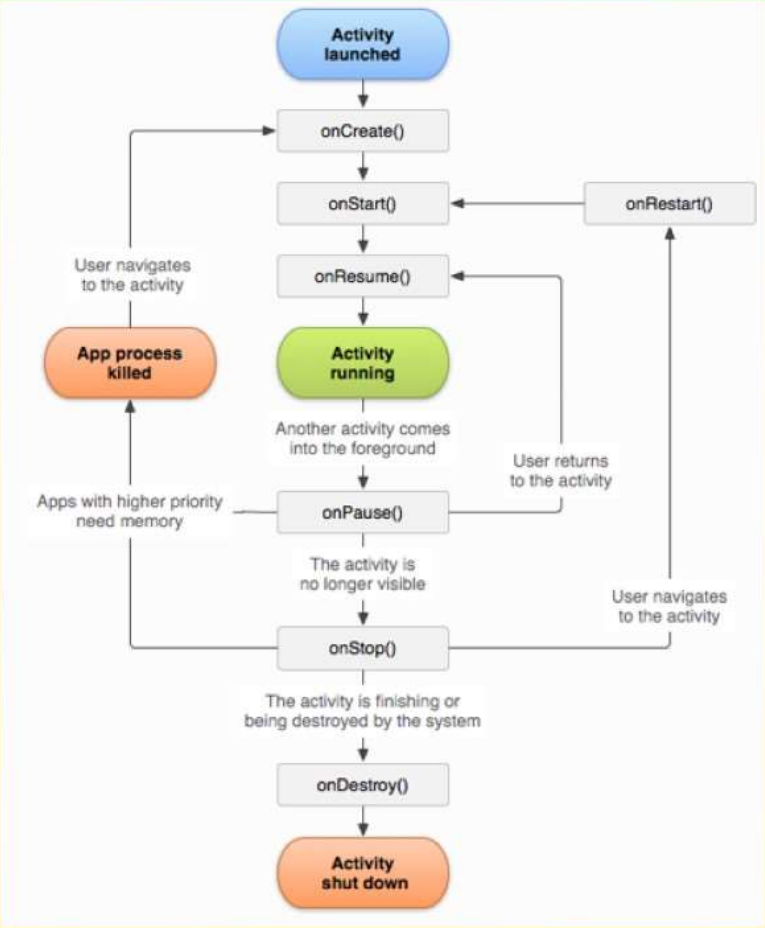
Communication téléphonique dure un certain temps → le système décide de **libérer des ressources**...

Lorsque l'utilisateur raccroche et revient à l'application de mails, l'activité est entièrement **RECRÉÉE** (**onCreate**, **onStart**, **onResume**).



L'utilisateur décide d'**ARRÊTER** l'application de mail.

Activité est alors complètement **DÉTRUITE** (**onDestroy**).



États d'un cycle de vie

onCreat

Elle est appelée lorsque votre activité est créée par le système. Généralement, les opérations effectuées dans cette méthode servent à mettre en place **l'interface graphique, à initialiser les variables, etc.**

C'est dans cette méthode que doivent être réalisés les traitements à exécuter **une seule fois pour toute la vie de l'activité.**

À ce stade, l'activité est créée, mais l'utilisateur **ne la voit pas encore** et ne peut pas interagir avec les différents éléments.

États d'un cycle de vie

onStart

Permets de savoir quand une activité va être rendue visible à l'utilisateur. **L'interface graphique devient visible à l'utilisateur**, mais il **ne peut pas encore interagir** avec les différents éléments.

Par exemple : Si votre activité joue une animation, cette méthode est une bonne candidate pour lancer automatiquement l'animation.

États d'un cycle de vie

onResume

L'activité devient **entièrement opérationnelle**. L'utilisateur peut utiliser l'application et cliquer sur les différents éléments graphiques. C'est le bon moment pour déclencher des opérations consommatrices de ressources (*activation de la caméra, ouverture de connexions réseau, etc.*)

L'application **reste dans cet état tant qu'il n'y a pas d'interruption** : la réception d'un appel téléphonique, le démarrage d'une nouvelle activité, l'affichage d'une boîte de dialogue, etc.

États d'un cycle de vie

onPause

Une activité passe dans cet état par exemple **lorsqu'une AlertDialog est affichée** : l'activité sous-jacente est toujours visible, mais elle n'est pas au premier plan.

Cette méthode est **l'inverse** de la méthode onResume : tout ce qui est initié dans onResume() doit être mis en pause dans cette méthode.

C'est **le moment de suspendre les opérations consommatrices de ressources** (*désactivation de la caméra, fermeture des connexions réseau, etc.*). Attention, une **activité en pause peut toujours être visible à l'utilisateur**. Vous ne devriez donc pas utiliser cette méthode pour modifier l'interface graphique ou stopper des animations.

Cette activité peut quand même être détruite par le système si ce dernier se trouve en manque de mémoire par exemple.

États d'un cycle de vie

onStop

Lorsqu'une nouvelle activité est démarrée, l'activité actuelle va se retrouver dans cet état. Elle n'est donc **plus du tout visible à l'utilisateur**.

Si votre activité a besoin de **sauvegarder des informations** (notamment sur disque) pour pouvoir redémarrer convenablement, vous pouvez effectuer ces traitements à ce moment-là. Vous pouvez utiliser cette redéfinition pour **stopper les opérations graphiques** (*animations, lecteur de vidéo, etc.*).

États d'un cycle de vie

onDestroy

Elle est appelée lors de la **mort de l'activité** (naturelle ou par le système).

Remarque : Quand une activité est dans un des états « onPause, onStop et onDestroy », elle est dans une situation où **elle peut être détruite pour libérer temporairement des ressources**. Cela ne signifie donc pas que l'activité a été simplement fermée par l'utilisateur. Parfois, l'urgence du système détruira l'activité sans même appeler onDestroy.

L'état « onPause » est le seul qui a la possibilité de s'exécuter complètement avant que l'activité ne soit détruite complètement. Il convient donc de **sauvegarder l'état de l'application dans cette méthode**.

États d'un cycle de vie

onRestart

Permets de savoir qu'une **activité va être à nouveau démarrée après avoir été stoppée**.

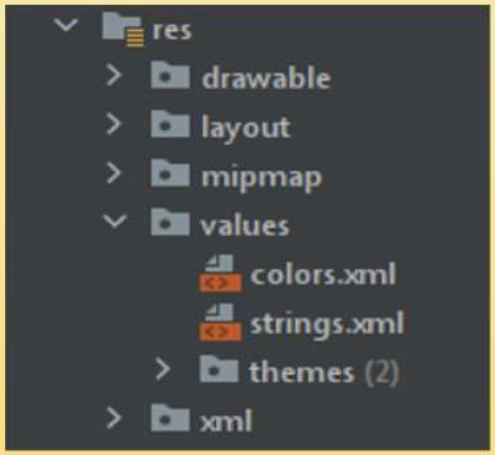
L'appel à cette méthode se situe entre onStop et onStart. Cela permet plus facilement de faire la **différence entre le démarrage normal de l'activité et son redémarrage**.

Centre Universitaire de MILA

Ressources

Centre Universitaire de MILA

Nous pouvons distinguer plusieurs types de ressources (*images, vidéo, audio, chaînes de caractères, etc.*) et toutes ces ressources se trouvent dans votre application sous le répertoire **res/**.



Organisation des ressources

res\drawable : ce répertoire contient les ressources qui peuvent être dessinées. Parmi ces ressources, l'on peut citer les images (jpeg, gif, etc.), des formes génériques (cercles, carrés, etc.) définies dans des fichiers XML, etc.;

res\layout : les différentes vues nécessaires pour la mise en page ;

res\values : des fichiers XML qui contiennent des valeurs simples, telles que des chaînes de caractères, des entiers et des couleurs, les styles, etc. ;

Utilisation Ressources

Chaines de caractères

Centraliser le texte pour le réutiliser.

- Stockées dans ***res/values/strings.xml***
- Utilisées avec ***@string/nom***
- Bonne pratique : aucun texte "en dur" dans les layouts

```
<string name="app_name">MonApp</string>
```

Drawables

Gérer les images et formes XML

- Stockées dans ***res/drawable/***
- Types : PNG, JPG, XML, etc.
- Exemple : @drawable/logo

rounded_button.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<shape
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">
    <solid android:color="@color/second_color" />
    <corners android:radius="30dp" />
</shape>
```

Styles

Uniformiser la présentation des vues (couleur, taille, coins, etc.)

- Définis dans res/values/themes.xml
- Utilisés avec style="***@style/MonStyle"Parent ....***"
- Avantage : facile à appliquer à plusieurs composants.

```
<style name="MyCustomButton" parent="TextAppearance.Compat.Notification">
    <item name="android:background">@drawable/rounded_button</item>
    <item name="android:textColor">#FFFFFF</item>
    <item name="android:textSize">18sp</item>
    <item name="android:padding">12dp</item>
    <item name="android:textStyle">bold</item>
</style>
```

```
<style name="Theme.TP_05" parent="android:Theme.Material.Light.NoActionBar" >
    <item name="android:colorBackground">@color/background</item>
    <item name="android:buttonStyle">@style/MyCustomButton</item>
</style>
```

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Enter"
    style="@style/MyCustomButton"
    />
```

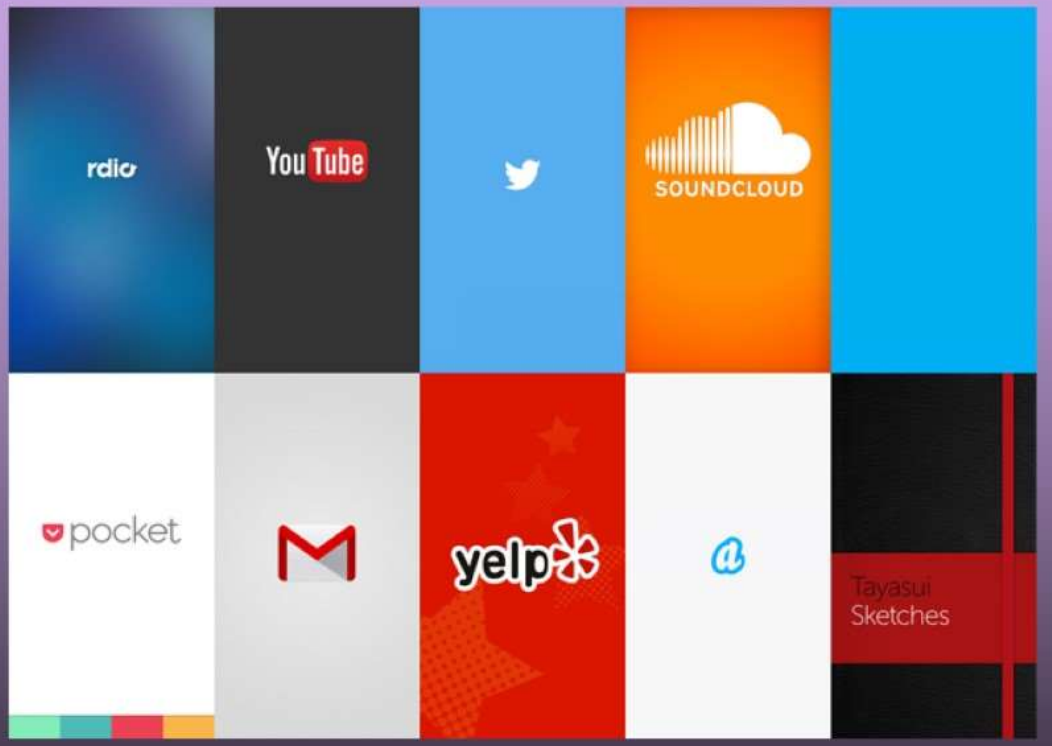
Animations

Ajouter du mouvement aux vues

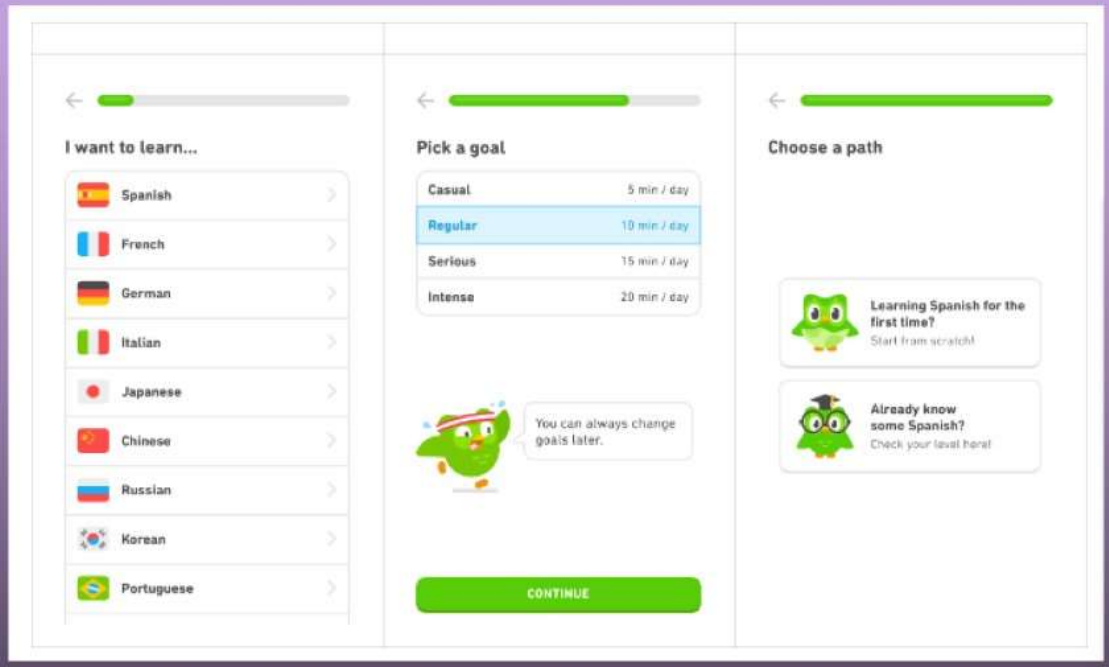
- Types : alpha, scale, rotate, translate
- Définies dans res/anim/
- Utilisées avec AnimationUtils.loadAnimation()

Le saviez-vous ?

Splash



Onboarding


I want to learn...
Spanish
French
German
Italian
Japanese
Chinese
Russian
Korean
Portuguese
Pick a goal
Casual 5 min / day
Regular 10 min / day
Serious 15 min / day
Intense 20 min / day
You can always change goals later.
CONTINUE
Choose a path
Learning Spanish for the first time?
Start from scratch!
Already know some Spanish?
Check your level here!

Projet
Guide Touristique
Mila

Travaux Pratique

Idées étudiants

Manar Boussouar

Application pour gérer salle des fêtes

Belmerabet kaouther

تطبيق البحث عن فرص عمل للطلاب والخريجين

Kah Guerbah Maroua

فكرة العقارات + فكرة رقمنة الانتخابات لتفادي الغش والتزوير +تطبيق لإمكانية رؤية طابور الصف المطعم في الاقامات لتفادي الاكتظاظ و الوقوف لساعات طويلة

Wissal Graichi

Oui, Applications travailleurs indépendants Une plateforme qui regroupe des artisans indépendants tels que les menuisiers, les électriciens et d'autres travailleurs indépendants.

Commence par faire le nécessaire, puis fait ce qu'il
est possible de faire et tu réaliseras l'impossible
sans t'en apercevoir