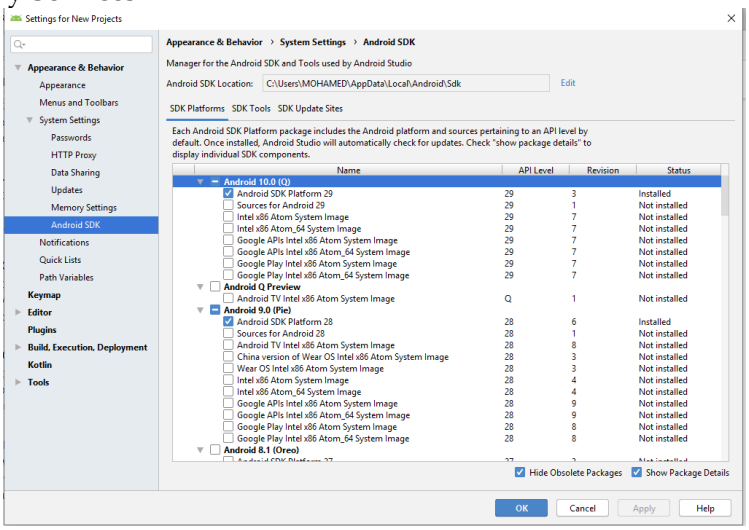
Atelier 1 : Prise en main d'Android Studio

Installation

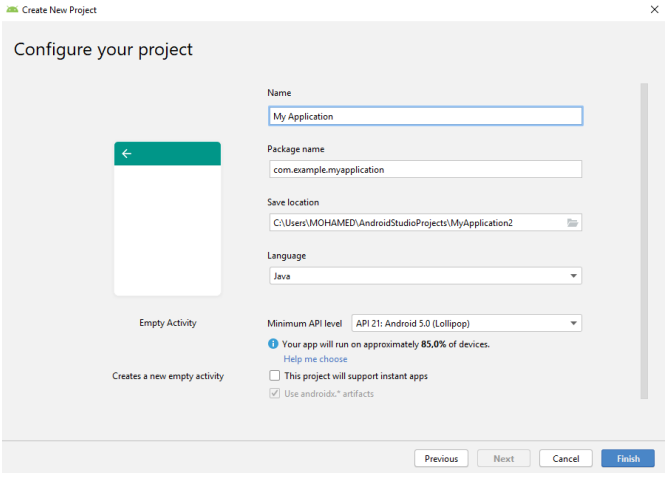
1. Téléchargez le dernier JDK (Java Development Kit) que vous pouvez trouver sur le site d'Oracle : https://www.oracle.com/technetwork/java/javase/downloads/index.html
2. Désinstallez des éventuelles versions antérieures du JDK
3. Installez le nouveau JDK
4. Téléchargez Android Studio. Il contient l'environnement de développement, le SDK (Software Development Kit) Android avec la dernière version de la plateforme, ainsi qu'un émulateur. https://developer.android.com/studio/index.html#downloads
5. Lancez l'exécutable pour démarrer l'installation et suivez le wizard. Si le répertoire Java n'est pas détecté automatiquement, il faudrait définir une variable d'environnement JAVA_HOME qui indique le répertoire où vous avez installé le JDK (ex : C :\Program Files\Java\jdk1.8…)
6. Lancez Android Studio
7. Nous commencerons par nous assurer que nous possédons tout ce qu'il faut pour développer. Dans la page de démarrage, sélectionnez Configure > SDK Manager. Dans le gestionnaire vous verrez la version du SDK installé (avec les mises jour disponibles) et aussi la version de l'API (Application Programming Interface) installée et la version du OS pour laquelle elle vous permettra de développer. Installez les éventuelles mises à jour. Assurez-vous de cocher au moins un System Image pour l'émulateur.
8. Dans l'onglet SDK Tools assurez-vous d'avoir au moins

- Android SDK Build Tools
- Android SDK Tools
- Android SDK Platform Tools
- Android Support Library
- Android Support Repository
- Google Repository
- Google Play Services



Première application Android

1. Dans le menu welcome, sélectionnez « Start a new Android Studio Project », Sélectionnez Empty Activity et cliquez Next puis renseignez les informations comme dans la figure



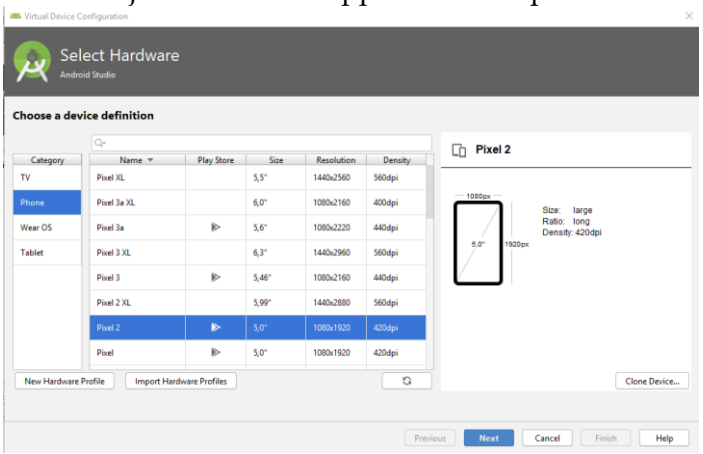
- N**ame** : c'est le nom qui va apparaître dans la liste des applications sur l'appareil et dans le Play Store.
- **Package name** : il est utilisé comme identifiant de l'application, il permet de considérer différentes versions d'une application comme étant une même application. Il doit être unique parmi tous les packages installés sur le système.
- **Minimum API level** : c'est la version Android la plus ancienne sur laquelle l'application peut tourner. Il faut éviter de remonter trop en arrière, ça réduirait les fonctionnalités que vous pourriez donner à votre application.
- **This project will support instant apps** : Il s'agit d'une fonctionnalité qui permet d'accéder à une application sans avoir à les installer

2. Cliquez sur Finish, le projet est créé.

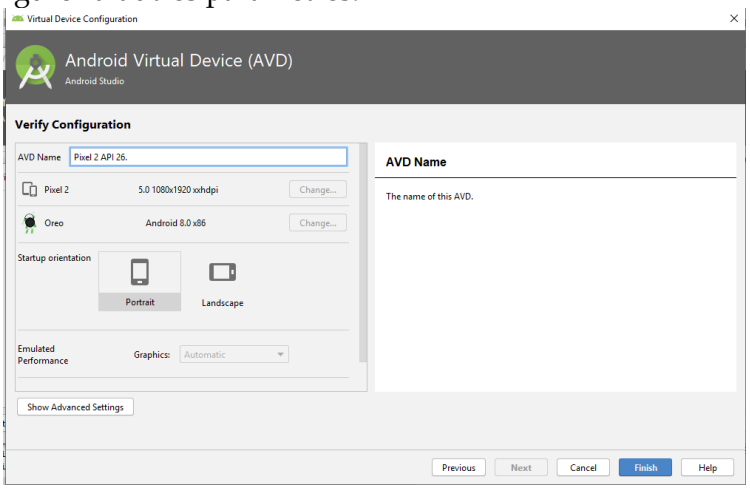
Exécution de l'application

Sur un émulateur :

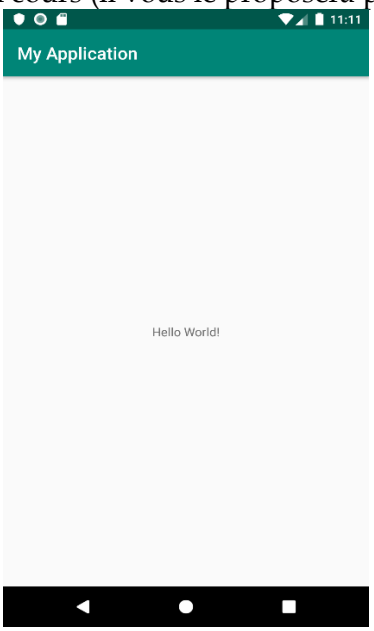
Un émulateur permet de reproduire le comportement d'un appareil réel d'une façon virtuelle. L'utilisation d'un émulateur nous évite d'avoir à charger à chaque fois l'application dans un appareil pour la tester. On pourra ainsi lancer l'application dans l'IDE et elle s'exécutera sur un appareil virtuel appelé Android Virtual Device AVD qui émule le comportement d'un téléphone, une tablette ou autre.
Pour configurer un émulateur, allez dans Tools > AVD Manager, un émulateur existe par défaut, mais vous pourriez rajouter d'autres appareils en cliquant sur Create Virtual Device.



Nous sélectionnons ensuite le processeur qui sera émulé. Vous pouvez télécharger d'autres images systèmes avec d'autres versions Android. En cliquant sur Next Vous avez ensuite la possibilité de configurer d'autres paramètres.



Notez qu'à la création de l'appareil sa résolution vous est signalée. Ceci est important à noter pour l'intégration d'images dans l'application.
Pour lancer l'exécution sur l'émulateur, appuyez sur le bouton d'exécution et sélectionnez l'émulateur sur lequel vous souhaitez lancer l'application.
Rassurez-vous, vous n'aurez pas à le relancer à chaque fois que vous compilez votre projet, laissez-le ouvert et à chaque fois que vous compilez et relancez votre application, elle pourra être chargée dans l'émulateur en cours (il vous le proposera parmi les Running devices).



Sur un appareil réel

Connectez l'appareil par câble USB à l'ordinateur et installez le pilote si nécessaire.
Activez l'option de débogage USB sur votre appareil (dans les paramètres, sous développement ou option de développement). Lancez l'application depuis Android Studio comme précédemment. Si on vous demande de choisir l'appareil, sélectionnez « Choose a running device », puis votre téléphone ou tablette. Android Studio installera l'application sur votre appareil et la lancera.
Une fois que votre application est compilée, un fichier .apk est créé dans le dossier **app\build\outputs\apk** de votre répertoire de travail.

C'est l'exécutable de votre application. C'est ce fichier que vous devez déployer pour distribuer votre application. Le contenu de ce fichier peut être inspecté à l'aide de n'importe quel logiciel standard de compression/décompression de fichiers

Construire une interface graphique avec Layout Editor

Vous allez maintenant créer l'interface graphique de l'application Bienvenue. L'éditeur de layout de l'IDE vous permet de construire votre interface par glisser-déposer des vues, comme les TextView , ImageView et Button, sur l'éditeur de mise en page. Par défaut, la mise en page de l'interface graphique pour une application basée sur le modèle d'activité vide est stockée dans un fichier XML appelé activity_main.xml, situé dans le dossier **res** du projet dans le sous-dossier layout. Nous allons utiliser l'éditeur de layout et la fenêtre arborescence des composants pour construire l'interface graphique. Vous pouvez modifiez le XML dans activity_main.xml, changer la mise en page utilisée pour organiser les TextView et ImageView de cette application.

1. **Ajout d'une image au projet** :

Les appareils Android ont différentes tailles d'écran, des résolutions et des densités de pixels (points par pouce ou DPI), on vous fournit généralement des images dans diverses résolutions que le système d'exploitation choisit sur la base de la densité de pixels du périphérique. Ceux-ci sont placés dans le dossier **drawable** avec différentes densités. Par exemple, les images pour les appareils qui sont semblables en densité à la Google Nexus 6 (560 dpi) seraient placés dans le dossier drawable-xxxhdpi. Les images des appareils avec des densités de pixels inférieurs sont placés dans les autres dossiers qui représente la densité de pixels la plus proche du périphérique réel.

Effectuer les étapes suivantes pour ajouter les images à ce projet :

➔ Dans la fenêtre Projet, développez le dossier res du projet.

➔ Copiez le fichier bug.png, puis dans la fenêtre Projet Android studio sélectionner le sous dossier drawable du dossier res et collez le fichier dans ce sous-dossier.

➔ Dans la boîte de dialogue qui apparaît, cliquez sur OK.

L'image peut maintenant être utilisé dans l'application.

2. **Ajout d'une icône au projet** :

Pour ajouter l'icône de lancement de l'application, cliquez droit sur le dossier res, puis sélectionnez New> Image Asset.

Cela ouvrira la fenêtre Asset Studio, ce qui vous permet de configurer l'icône de l'application à partir d'une image existante, un morceau de clip art ou d'un texte.

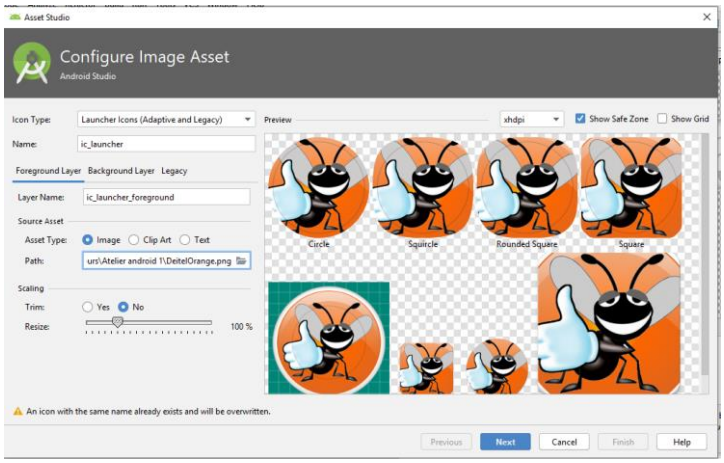
Pour cette application, nous avons choisi l'image DeitelOrange.png. Pour utiliser cette image :

➔ Cliquez sur le bouton de sélection Image de Asset Type Path.

➔ Naviguez vers le dossier contenant l'image.

➔ Sélectionnez DeitelOrange.png et cliquez sur OK. Des aperçus des images redimensionnées sont affichées dans la zone de prévisualisation de la boîte de dialogue.

➔ Cliquez sur Suivant, puis cliquez sur Terminer.



L'IDE crée plusieurs versions mises à l'échelle de l'image, chaque ic_launcher.png nommé, et les place dans les sous-dossiers de mipmap du projet du dossier res. Les sous-dossiers de mipmap sont similaires aux sous-dossiers drawable, mais sont spécifique pour l'icône de l'application. Lorsque vous téléchargez une application sur Google Play, vous pouvez télécharger plusieurs versions de l'application pour différentes tailles d'appareils et résolutions d'écran. Toutes les images dans les dossiers mipmap sont téléchargés avec toutes les versions de votre application, alors que vous pouvez supprimer des dossiers drawable supplémentaires pour des densités spécifiques de pixels à partir d'une version de l'application donnée afin de minimiser la taille totale de l'installation.

3. Changer ConstraintLayout en un LinearLayout

Lorsque vous ouvrez un fichier XML de mise en page (layout), la conception de la mise en page apparaît dans l'éditeur de mise en page et les vues de la mise en page et leurs relations hiérarchiques apparaissent dans la fenêtre arborescence des composants. Pour configurer le layout view, vous pouvez le sélectionner dans l'éditeur de mise en page ou dans l'arborescence des composants, puis utilisez la fenêtre Propriétés de l'arborescence des composants pour spécifier la valeur des propriétés de la vue sans modifier le XML directement. Lors de la conception et la modification de layout plus complexes, il est souvent plus facile de travailler directement dans l'arborescence des composants.
Pour certaines modifications telles que la modification du ConstraintLayout par défaut en un LinearLayout vous devez modifier directement le fichier XML. Pour ce faire :

➔ Cliquez sur l'onglet Text en bas de l'éditeur de mise en page pour passer du mode Création au texte XML.
➔ En haut du XML, double cliquez sur le nom de l'élément XML ConstraintLayout pour le sélectionner, puis commencez à taper LinearLayout.
➔ En même temps la balise fermante change pour les maintenir en synchronisation, et une fenêtre complétion de code apparaît contenant les noms d'éléments qui commencent par les lettres que vous avez tapées jusqu'ici. Une fois LinearLayout apparaît dans la fenêtre de code d'achèvement et est en surbrillance, appuyez sur Entrée (ou Retour) pour sélectionner LinearLayout et activer Android Studio pour auto-compléter.
➔ Enregistrez les modifications et revenir à l'onglet Création de l'éditeur de mise en page.

4. Modification du Id et de l'orientation du LinearLayout

Pour modifier l'orientation du LinearLayout, cliquez sur le fond blanc de l'écran du téléphone virtuel dans l'éditeur de mise en page pour afficher les propriétés, puis sélectionnez verticale pour l'orientation. Entrez le nom « bienvenueLinearLayout » dans le champ Id.

Vous remarquerez que dans le fichier XML la ligne « android:id="@+id/bienvenueLinearLayout" » s'ajoute indiquant grâce à @+id qu'un nouvel id est ajouté dont le nom suit le /.

5. **Configuration id et text des propriétés de la TextView**

Cliquez sur le TextView dans l'éditeur de mise en page, puis dans la boîte de propriété qui apparaît régler l'id: bienvenueTextView et appuyez sur Entrée

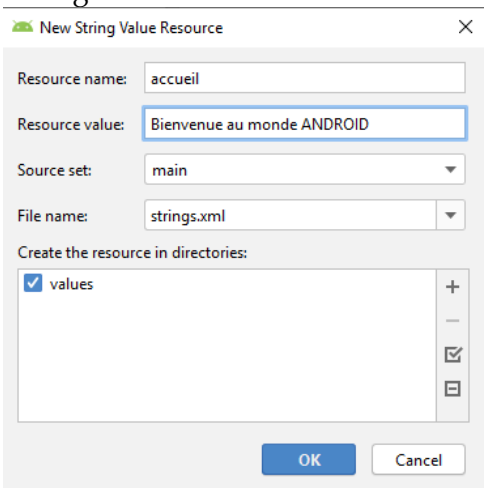
Selon la documentation Android pour les ressources d'application

http://developer.android.com/guide/topics/resources/index.html

Il est considéré comme une bonne pratique de placer les chaînes de caractères, les tableaux, les images, les couleurs, les tailles de police, les dimensions et d'autres ressources dans des fichiers XML dans les sous-dossiers du dossier res du projet, de sorte que ces ressources peuvent être gérées séparément du code Java de votre application. Ceci est connu comme l'externalisation des ressources.

Pour définir la propriété texte du TextView, créer une nouvelle ressource de chaîne dans le fichier strings.xml comme suit :

1. Double cliquez sur bienvenueTextView dans l'éditeur de mise en page ou sélectionnez bienvenueTextView et localiser sa propriété text dans la fenêtre Propriétés
2. Cliquez sur le bouton (barre droite : Pick a resource) à droite de la valeur de la propriété pour afficher la boîte de dialogue Pick a resource
3. Dans la boîte de dialogue Pick a resource, cliquez sur add new resource, puis sélectionnez New String Value Resource ... pour afficher la boîte de dialogue New String value resource et remplir comme indiqué dans la figure suivante.



6. Configuration de de la propriété textSize

Les tailles peuvent être spécifiées dans différentes unités de mesure. La documentation vous recommande d'utiliser la densité indépendante des pixels (density-independent pixels) pour les dimensions des vues et d'autres éléments de l'écran, et les pixels d'échelle indépendante (scale-independent pixels) pour les tailles de police :

http://developer.android.com/guide/practices/screens_support.html

px : pixel ; dp ou dip: density-independent pixel ; sp : scale-independent pixel ; in : inches ; mm : millimeters ;

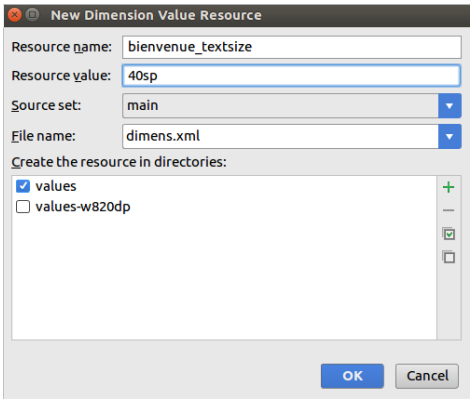
Définir vos interfaces graphiques avec dp permet la mise à l'échelle de l'interface graphique, basée sur la densité de pixel de l'écran d'un périphérique donné. Un dp est équivalente à un pixel sur un écran de 160 dpi. Sur un 240 ppp écran, chaque dp sera mise à l'échelle par un facteur de 240/160

Création d'une ressource Dimension pour la taille de police sur un appareil.

Vous allez maintenant augmenter la taille de police :

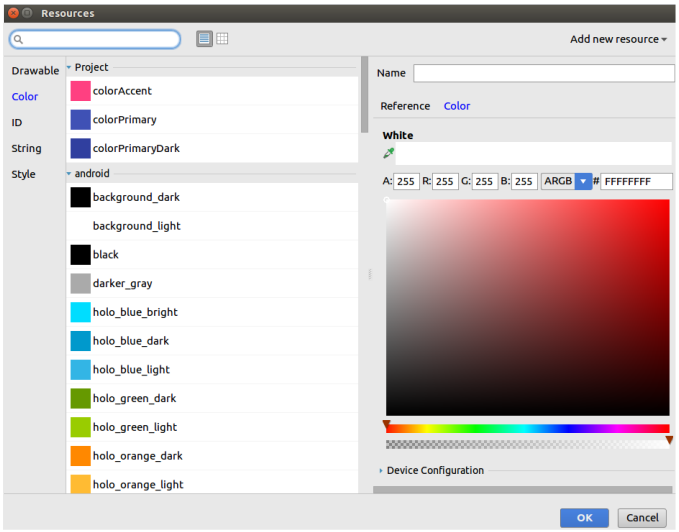
1. Sélectionnez le bienvenueTextView dans l'éditeur de mise en page.

2. Repérez la propriété textSize, puis cliquez dans la colonne de droite pour révéler le bouton de sélection (...) et cliquez sur le bouton pour afficher la boîte de dialogue Ressources.
3. Dans la boîte de dialogue Ressources, cliquez sur Nouvelle ressource, puis sélectionnez New Dimension Value pour afficher la boîte de dialogue Nouvelle Dimension Valeur des ressources.
4. Dans la boîte de dialogue qui apparaît, spécifiez bienvenue_textsize le nom de la ressource et 40sp pour la valeur des ressources, puis cliquez sur OK. Les lettres sp dans la valeur 40sp indiquent que ceci est une mesure de pixel échelle indépendante. Les lettres dp une valeur de dimension (par exemple, 10DP) indiquent une mesure de pixel indépendante de la densité.
Remarquer les changements dans le fichier XML.

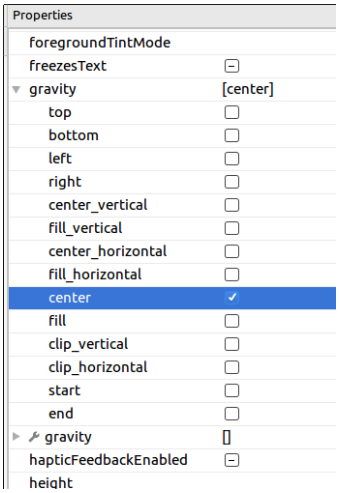


Création d'une ressource Dimension pour la taille de police sur un grand périphérique : Tablette.
Android peut automatiquement choisir différentes valeurs de ressources en fonction de la taille des périphériques, des orientations, des densités de pixels, les langues parlées, locales et plus. Pour spécifier une taille de police distincte pour des appareils plus gros tels que des tablettes :
1. Ré-ouvrez la boîte de dialogue New Dimension Value resource.
2. Entrez bienvenue_textsize le nom de ressource (les noms de ressources doivent correspondre pour Android pour sélectionner les différentes valeurs des ressources automatiquement) et entrez 80sp pour la valeur des ressources.
3. Ensuite, vous allez créer un nouveau dossier des valeurs de ressources qui est spécifique à des appareils plus gros qui ont des largeurs et hauteurs qui sont au moins 600dp. Dans la boîte de dialogue New Dimension Value resource, décochez la case des valeurs, puis cliquez sur le bouton Ajouter (+) pour ouvrir la boîte de dialogue New Resource Directory. Dans Available qualifiers, sélectionnez Screen Width, puis cliquez sur le bouton >> pour ajouter la largeur de l'écran. Ensuite, entrez 600 dans le champs Screen width.
4. Ensuite, ajouter le Screen Height à la liste et entrez 600 pour la hauteur de l'écran.
5. Cliquez sur OK pour créer une values-xlarge nouveau dossier de ressource nommé.
6. Dans la boîte New Dimension Value Resource, cochez la case values-w600dp-h600dp, puis cliquez sur OK. Cela crée une autre ressource de dimension bienvenue_textsize dans un fichier dimens.xml qui est stocké sur le disque dans le dossier res/values-w600dp-h600dp du projet.
Android va utiliser cette ressource pour les appareils avec une grande largeur d'écran et des hauteurs qui sont au moins 600dp. Le nouveau fichier de ressources dimens.xml apparaît dans Android Studio à res/values/dimens.xml du projet dimens.xml (w600dp-h600dp)
7. **Réglage de la propriété textColor**
Lorsque vous avez besoin de couleurs personnalisées dans vos applications, les directrices de Google recommandent d'utiliser les couleurs de la palette de couleurs Material Design à : http://www.google.com/design/spec/style/color.html

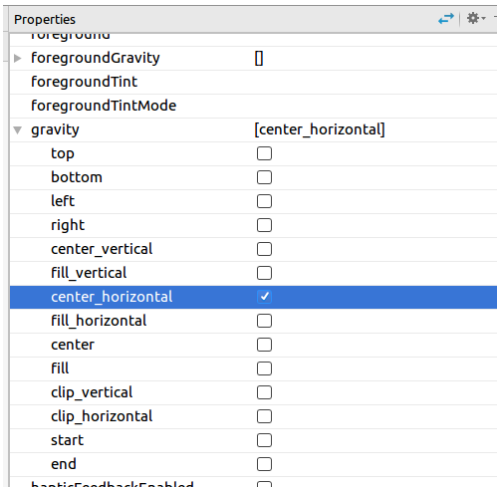
Les couleurs sont indiquées en tant que RVB (rouge-vert-bleu) ou ARGB Valeurs (alpha-rouge-vertbleu).
Une valeur RGB se compose de valeurs entières entre 0-255 qui définissent les quantités de respectivement rouge, vert et bleu dans la couleur. Les couleurs personnalisées sont définies au format hexadécimal, de sorte que les composantes RVB sont des valeurs dans la plage 00 (la valeur hexadécimale de 0) à FF (la valeur hexadécimale de 255).
Android prend également en charge les valeurs alpha (transparence) dans la gamme 00 (complètement transparent) à FF (complètement opaque). Pour utiliser alpha, vous spécifiez la couleur comme #AARRGGBB, où les deux premiers chiffres hexadécimaux représentent la valeur alpha.
Si les deux chiffres de chaque composante de couleur sont les mêmes, vous pouvez utiliser les formats de valeur #RGB ou #ARGB abrégé. Par exemple, la valeur RVB # 9AC est équivalente à #99AACC et ARGB valeur # F9AC est équivalente à # FF99AACC.
Pour définir la propriété textColor du TextView à une nouvelle ressource de couleur :
1. Dans la fenêtre Propriétés cliquez sur le bouton de sélection (...) de textColor pour afficher la boîte de dialogue ressources, puis cliquez sur Nouvelle ressource et sélectionnez New Color Value
2. Entrez bienvenue_text_color pour le nom de la source et # 2196F3 pour la valeur des ressources, puis cliquez sur OK.
Vérifier le changement dans colors.xml



8. **Réglage de la propriété gravity**
Pour centrer le texte dans la TextView si elle est sur plusieurs lignes, vous pouvez définir sa propriété de gravity à center



9. **Réglage de la propriété gravity du textView layaout**
Dans cette application, nous aimerions centrer le TextView horizontalement dans le LinearLayout.
Pour ce faire, vous définissez sa disposition : la propriété gravity à centrer horizontalement comme suit :
1. Avec le TextView sélectionné, développez layout:gravity dans la fenêtre Propriétés.
2. Cliquez sur le champ center_horizontal.



Vérifier le changement dans activity_main.xml
10. **Réglage de la propriété layout:weight du textView**
Un LinearLayout peut proportionnellement dimensionner ses enfants en fonction de leur layout:weights, qui précis la taille relative de la vue par rapport à d'autres layout. Par défaut, le layout:weight est 0 pour chaque vue que vous ajoutez à un LinearLayout, indiquant que la vue ne devrait pas être de taille proportionnelle.
Dans cette application, nous aimerions que le TextView et ImageView occupent la moitié de la linéarité de l'espace vertical du layout. Vous accomplissez cela en définissant layout:weight de chaque vue à la même valeur. Le LinearLayout utilise le rapport layout:weight de chaque vue au total layout:weight à allouer. Dans cette application, vous définissez le layout:weight à 1 pour le TextView et l'ImageView. Le layout:weightt total sera 2 et chaque vue occupera la hauteur de ½ du layout.
Si vous voulez que le TextView occupe un tiers de la hauteur du LinearLayout, vous pourrait définir sa layout:weight à 1 et celle de l'ImageView à 2.
Réglez le layout:weight à 1. L'éditeur de layout affiche un warning « Warning : Use a `layout_height` of `0dp` instead of `wrap_content` for better performance », et vous suggère des corrections, cliquer sur « Replace size attribute with 0dp » pour appliquer la recmandation.
Ce changement permet au LinearLayout de calculer les tailles de ses enfants plus efficacement.
11. Ajout d'un ImageView pour afficher l'image
Pour ajouter un ImageView à l'interface graphique pour afficher l'image « bug.png » que vous avez ajouté au projet dans la section 1. Vous faites cela en faisant glisser un ImageView de la section Widgets de la palette sur la toile en dessous de la TextView. Lorsque vous faites glisser une vue sur la toile, l'éditeur de mise en page affiche des lignes de guidage orange, lignes de guidage verts et une info-bulle :
• Les lignes de guidage orange montrent les limites de chaque vue existant dans la mise en page.
• Les lignes de guidage vertes indiquent l'endroit où la nouvelle vue sera placé par rapport aux vues existantes par défaut, les nouvelles vues sont ajoutées au bas d'un LinearLayout

vertical, à moins que vous placiez la souris au-dessus de la boîte orange qui limite le plus haut point de vue de la mise en page.
• Les infobulles montrent comment la vue sera configuré si vous déposez à la position courante.
Pour ajouter et configurer l'ImageView : Dans la section Widgets de la palette, faites glisser un ImageView sur la toile. Avant de relâcher la souris, veiller à ce que le centre apparaît dans l'infobulle au sommet de la conception, cela indique que ImageView layout:gravity sera centrer horizontalement dans le LinearLayout. Lorsque vous déposez l'ImageView en relâchant la souris, l'éditeur de mise en page suppose que l' ImageView layout:weight devrait être le même que le TextView et définit layout:weight à 1. Il définit également le layout_height à 0DP comme nous l'avons fait pour le TextView. La nouvelle ImageView apparaît sous le TextView dans la conception et au-dessous du bienvenueTextView dans l'arborescence des composants. La fenêtre Resources apparaît choisissez « bug.png ».
Changer l'id de l'imageView à bugImageView.

12. Rendre l'application accessible
Android contient des fonctionnalités d'accessibilité pour aider les personnes souffrant de certains handicaps à utiliser leurs appareils. Pour les personnes ayant une déficience visuelle, Android TalkBack peut synthétiser du texte de l'écran ou le texte que vous fournissez (lors de la conception de votre interface graphique ou par programmation) pour aider l'utilisateur à comprendre le but d'une vue. Android fournit également Explorer au toucher, ce qui permet à l'utilisateur d'entendre TalkBack ce qui est sur l'écran où l'utilisateur touche. Lorsque TalkBack est activé et que l'utilisateur touche une vue pour laquelle le texte d'accessibilité est spécifié, l'appareil vibre pour indiquer que l'utilisateur a touché une vue significative et TalkBack synthétise le texte de l'accessibilité. Vous devez activez TalkBack dans paramètres de l'application sous l'accessibilité. Sur cette page, vous pouvez également activer d'autres fonctions d'accessibilité Android comme une plus grande taille de texte par défaut et la capacité d'utiliser des gestes qui amplifient les zones de l'écran. TalkBack n'est pas actuellement pris en charge sur AVDS, vous devez donc exécuter cette application sur un appareil pour entendre TalkBack.
Activation de TalkBack pour l'ImageView
Dans l'application Bienvenue, nous n'avons pas besoin de texte plus descriptif pour le TextView, parce que TalkBack va lire le contenu de la TextView. Pour un ImageView, cependant, il n'y a pas de texte pour TalkBack. Il est considéré comme une bonne pratique dans Android pour veiller à ce que chaque point de vue peut être utilisé avec TalkBack en fournissant le texte pour la propriété contentDescription de toute vue qui ne comporte pas de texte.
Pour cette raison, l'IDE nous a avertis que quelque chose clochait en affichant un warning dans l'éditeur lors de l'ajout de l'ImageView indiquant "[Accessibility] Missing contentDescription attribut on image." Le texte que vous fournissez devrait aider l'utilisateur à comprendre le but de la composante. Pour un ImageView, le texte doit décrire l'image.
Pour ajouter un contentDescription à l'ImageView (et éliminer l'avertissement) :
1. Sélectionnez le bugImageView dans l'éditeur.
2. Dans la fenêtre Propriétés, cliquez sur le bouton de sélection à droite de la propriété contentDescription pour ouvrir la boîte de dialogue Ressources.
3. Cliquez sur Nouvelle ressource, puis sélectionnez New String Value pour afficher la boîte de dialogue Nouvelle valeur de chaîne de ressources.
4. Dans le champ Nom de ressources spécifient deitel_logo et dans le champ de la valeur des ressources spécifier " image du logo Deitel", puis appuyez sur OK. La nouvelle ressource de chaîne est choisie automatiquement la valeur de contentDescription.
Après avoir défini contentDescription, l'éditeur de mise en page supprime le warning.

Test de l'App avec TalkBack activé
Exécutez cette application sur un appareil avec TalkBack activé, puis appuyez sur la TextView et ImageView entendre parler TalkBack le texte correspondant.
Créé dynamiquement des vues
Certaines applications créent dynamiquement des vues en réponse aux interactions de l'utilisateur.
Vous pouvez définir par programme le texte d'accessibilité. Pour plus d'informations sur ce sujet et d'autres fonctions d'accessibilité d'Android, et pour une liste de contrôle à suivre pour le développement d'applications accessibles, visitez:
http://developer.android.com/design/patterns/accessibility.html
http://developer.android.com/guide/topics/ui/accessibility

13. Internationaliser Votre Application
Pour atteindre le plus grand public possible, vous devriez envisager la conception de vos applications afin qu'ils puissent être personnalisés pour divers endroits et langues. Ensuite, si vous avez l'intention de publier votre application, en italie, vous devez traduire ses ressources (textes, fichiers audios, etc.) en italien. Vous pouvez également choisir d'utiliser différentes couleurs, des graphiques et des sons sur la base des paramètres régionaux. Pour chaque langue, vous aurez, un ensemble personnalisé distinct de ressources. Lorsque l'utilisateur lance l'application, Android trouve automatiquement et charge les ressources qui correspondent aux paramètres régionaux de l'appareil. Concevoir une application de sorte qu'elle peut être personnalisé est connu comme l'internationalisation. La personnalisation des ressources d'une application pour chaque environnement local est connue comme la localisation.
Un avantage clé de la définition de vos valeurs de chaîne en tant que ressources de chaîne (comme nous l'avons fait dans cette application) est que vous pouvez facilement localiser votre application en créant des fichiers supplémentaires de ressources XML pour les ressources de chaîne dans d'autres langues. Dans chaque fichier, vous utilisez les mêmes noms string-ressources, mais fournissez la chaîne traduite. Android peut alors choisir le fichier de ressources approprié en fonction de la langue choisie par l'utilisateur du périphérique.
Renommer les dossiers pour les ressources localisées
Les fichiers de ressources XML contenant des chaînes localisées sont placées sur le disque dans les sous-dossiers du dossier res du projet. Android utilise un système spécial folder-naming scheme pour choisir automatiquement une localisation correcte des ressources, par exemple, le dossier des values-en contiendrait un fichier strings.xml pour l'anglais, le dossier values-es contiendrait un fichier strings.xml pour l'espagnol. Si les ressources localisées ne sont pas fournies pour un endroit donné, Android utilise les ressources par défaut de l'application.
Ajout d'une chaîne de traduction au projet.
Android Studio fournit un éditeur de Traductions en ajoutant des traductions rapidement et facilement pour les chaînes existantes dans votre application. Suivez ces étapes pour ajouter des chaînes traduites au projet :
1. Dans la fenêtre Projet, développez le noeud values, puis ouvrez le fichier strings.xml.
2. Dans le coin supérieur droit de l'éditeur, cliquez sur le lien Open editor pour ouvrir l'éditeur de traduction.
3. Dans le coin supérieur gauche de l'éditeur de traductions, cliquez sur le bouton Add Locale, puis sélectionnez l'italien (it). Après avoir sélectionné les paramètres régionaux dans la liste, une nouvelle strings.xml (it) fichier sera créé et placé dans le noeud strings.xml dans la fenêtre Projet.

L'éditeur de traductions affiche également une nouvelle colonne pour les traductions en italien.
4. Pour ajouter une traduction en italien pour une ressource de chaîne donnée, cliquez sur la cellule pour l'italien la traduction de la ressource, puis dans la traduction : champ en bas de la fenêtre entrer le texte espagnol. Si une chaîne ne doit pas être traduit (par exemple, une chaîne qui n'a jamais affiché à l'utilisateur), cochez la case intraduisible pour cette ressource String.
Répétez les étapes précédentes pour chaque langue que vous souhaitez soutenir.
Exécuter sur un simulateur en changeant sa langue dans les paramètres de langue.