

## 4. *Layouts*

---

Présentation des stratégies de *layouts* de widgets et comment *Android Studio* facilite leur utilisation.

*Android Studio Hedgehog Essentials, Kotlin Edition* : chapitres 23 à 28

### 4.1. *Présentation Powerpoint*

1. Présentez la présentation portant sur les vues, groupes de vues et *layouts*

### 4.2. *Créer une nouvelle application*

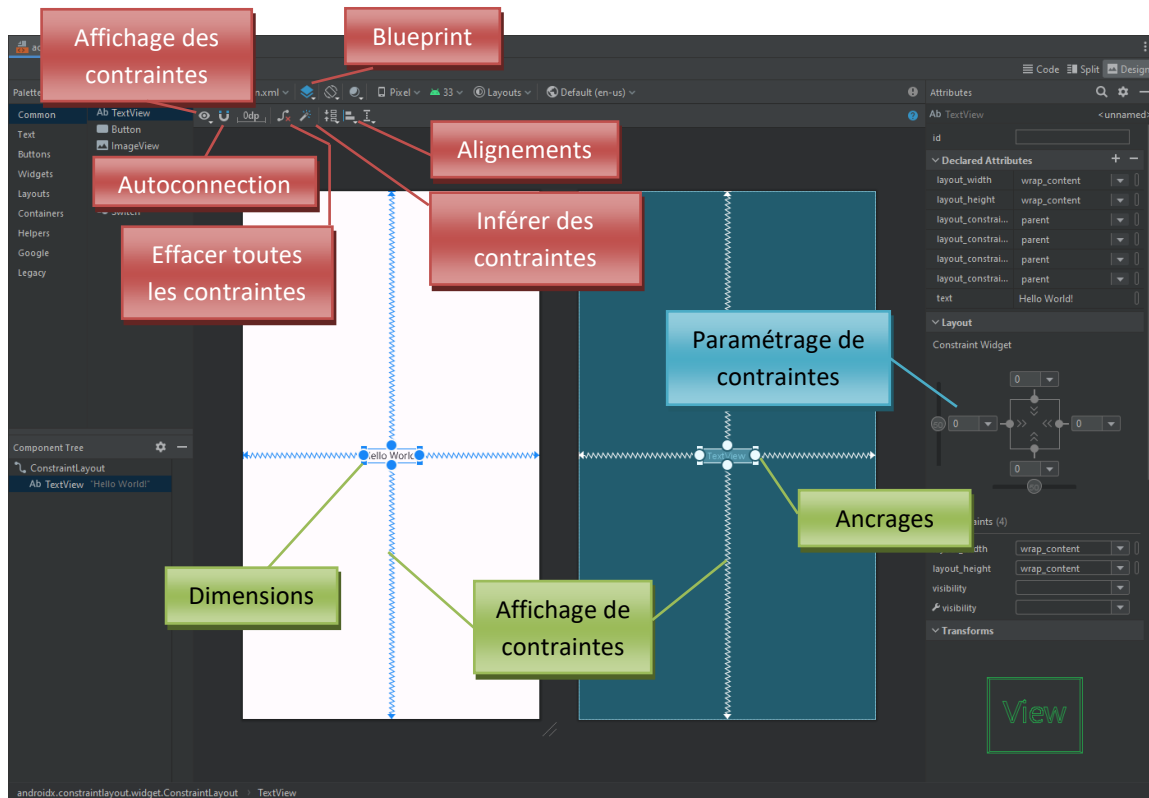
2. Expliquez l'importance d'une bonne gestion du *layout* d'une application, et les multiples difficultés rencontrées par le programmeur afin d'assurer cette gestion dans son projet *Android*
  - a. Expliquez que vu la complexité du sujet, nous n'allons brosser qu'un aperçu de l'intégration des *layouts*
3. Indiquez aux étudiants qu'ils doivent effectuer les mêmes opérations que l'instructeur
4. Démarrez *Android Studio* et sélectionnez « New Project ». Si un projet antérieur est déjà chargé au démarrage de l'application, fermez celui-ci au préalable via File » Close Project
5. Dans la fenêtre de gabarits, sélectionnez « Empty Views Activity » puis appuyez sur « Next »
6. Nommez le projet « *Layouts* », entrez le nom de domaine du collège et sélectionnez un répertoire où créer le projet. Choisissez « Kotlin » comme langage, et sélectionnez « API 26: Android 8.0 (Oreo) » comme SDK minimal (ne pas activer les autres options). Appuyez ensuite le bouton « Finish »
7. Assurez-vous que le *layout* `activity_main.xml` est chargé dans l'éditeur Design

8. Supprimez le widget « *Hello World!* » du fichier XML via l'éditeur Text, puis affichez l'éditeur Design (ou son *preview*) pour montrer que le `TextView` est bien disparu
9. Dans l'éditeur Text, soulignez que la `View` de base de l'activité est un `ConstraintLayout`

### 4.3. Aperçu du widget `ConstraintLayout`

10. Depuis *Android Studio 2*, il existe un nouveau type de conteneur *layout* visant à combiner les forces de tous les autres : `ConstraintLayout`
  - a. Expliquez que les développeurs de *Google* ont travaillé fort pour unifier tous les conteneur *layout* en un seul conteneur
11. La version 2 de *Android Studio* a été complètement réécrite pour y intégrer la définition et configuration du widget `ConstraintLayout`
12. Le fait que `ConstraintLayout` unifie les autres conteneurs *layout* a deux conséquences
  - a. Les contraintes d'un widget peuvent être difficiles à configurer lorsque le widget ne se positionne pas comme on le veut
  - b. L'interface de *Android Studio* permettant de manipuler les widgets dans un `ConstraintLayout` est assez complexe
13. Soulignez que ce tutoriel ne donne qu'un bref aperçu des fonctionnalités du `ConstraintLayout` et de *Android Studio* pour les manipuler
  - a. Nous étudierons plus attentivement les particularités de ce *layout* selon nos besoins durant le trimestre

14. Décrivez brièvement les principales fonctionnalités :





15. Afin d'explorer les fonctionnalités de `ConstraintLayout` dans l'éditeur graphique d'*Android Studio*, désactivez *Autoconnection*

16. Pour illustrer les fonctionnalités de *Android Studio* pour gérer les `ConstraintLayout`, remplacez le `TextView` par un `Button` au centre l'activité, puis déplacez celui-ci pour montrer les lignes guides affichées dans l'éditeur selon la position du widget

- a. Soulignez que ces lignes guides sont affichées à titre indicatif seulement, elles ne résultent pas en contraintes

17. Sélectionnez le `Button` et cliquez le bouton *Infer Constraints*, puis décrivez les contraintes attribuées au widget

- a. Déplacez le `Button` et soulignez le changement de biais de contraintes affichés dans les attributs du widget (paramétrage de contraintes). Expliquez ce qu'est un biais de contrainte

- b. Recentrez le `Button` via l'*Attributes Window* en déplaçant les indicateurs de biais de contraintes
- 18. Effacer les contraintes associées au `Button` via le bouton  du menu contextuel associé au widget (affiché en cliquant le bouton droit de souris sur celui-ci).
- 19. Recréez les quatre contraintes séquentiellement en exploitant les ancrages sur chaque côté du widget ().
- 20. Supprimez les contraintes du `Button`, puis déplacez-le au haut de l'activité (à 8dp du bord). Imposez une contrainte afin qu'il demeure à cette distance du haut, ainsi que des contraintes latérales afin qu'il demeure centré horizontalement
  - a. Assurez-vous que les attributs `layout_width` et `layout_height` du `Button` sont tous deux à `wrap_content`
  - b. Expliquez la différence entre une contrainte fixe (un seul côté du widget) et des contraintes opposées (des deux côtés du widget)
- 21. Déposez un deuxième `Button` sous le premier, et fixez-le à 24dp sous celui-ci. Répétez l'opération avec un troisième `Button` afin d'avoir trois boutons au haut de l'activité, espacés verticalement de 24dp l'un de l'autre
- 22. Tournez l'appareil pour démontrer que seul le premier des trois `Button` est centré horizontalement
  - a. Pour centrer les deux autres, utilisez des contraintes d'alignement au premier `Button` plutôt que des contraintes opposées
- 23. Demandez aux étudiants de faire en sorte que les trois `Button` soient positionnés au centre de l'activité, horizontalement centrés et à 16dp l'un de l'autre verticalement
- 24. Les étudiants vont probablement exploiter des contraintes opposées sur chaque `Button` pour qu'ils soient centrés horizontalement. Montrez qu'il est difficile d'attribuer plutôt des contraintes d'alignement basées sur le `Button` du milieu (c.à.d. le second `Button` verticalement) car la commande d'alignement du menu contextuel ne fait pas ce qu'on désire

25. Éditez le fichier XML de l'activité afin les contraintes d'alignement des `Button` afin que celui du haut et celui du bas soient alignés sur celui du milieu (contraintes `app:layout_constraintStart_toStartOf` et `app:layout_constraintEnd_toEndOf`)
- Expliquez que l'éditeur *Design* d'*Android Studio* a ses limites et qu'il faut souvent avoir recours au code XML d'un widget pour configurer adéquatement ses contraintes
26. Afin de donner un exemple de chaîne de contraintes, ajoutez un quatrième `Button` et expliquez qu'on désire que les quatre `Button` soient répartis verticalement sur la hauteur de l'activité
- Effacez toutes les contraintes existantes de l'activité
  - Sélectionnez les quatre `Button`, puis la commande **Chain » Create Vertical Chain** du menu contextuel accessible via le bouton droit de la souris
  - Tournez l'appareil pour souligner la distribution verticale des `Button`
27. Déplacez horizontalement un des `Button` pour démontrer que la chaîne ne gère que leur espacement, mais non leur alignement
- Pour aligner verticalement les `Button`, sélectionnez ceux-ci, puis la commande **Align » Horizontal Centers** du menu contextuel
  - Si les boutons ne sont pas centrés horizontalement, identifié via XML le bouton à la base de l'alignement horizontal, et appliquez deux contraintes horizontales opposées à celui-ci
28. Soulignez aux étudiants que, s'ils se trompent, ils peuvent toujours faire **Edit » Undo...** (**Cmd+Z**) pour annuler les dernières opérations

29. Pour démontrer l'alignement de contenu, déposez un `TextView` à côté du premier `Button`, puis imposez-lui des contraintes latérales afin qu'il demeure centré horizontalement entre la marge gauche de l'activité et le premier `Button`

- a. Tournez l'appareil pour montrer que le `TextView` ne demeure pas aligné à côté de son `Button`

30. Plutôt qu'imposer une contrainte d'alignement vertical au `TextView`, imposez-lui une contrainte d'alignement de contenu (activée via le bouton à cet effet)

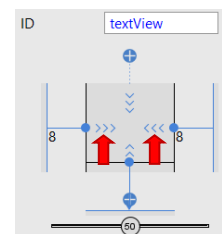
- a. Dans le menu contextuel du `TextView`, sélectionnez la commande `Show Baseline`



- b. Réorientez l'appareil pour démontrer que le `TextView` demeure maintenant aligné avec le `Button`
- c. Expliquez en détails la distinction entre les contraintes d'alignement de widget et celles d'alignement de contenu

31. Pour démontrer les contraintes de dimensionnement de widget, sélectionnez le `TextView` et modifiez son attribut `text` en soulignant que le widget se redimensionne automatiquement selon son contenu

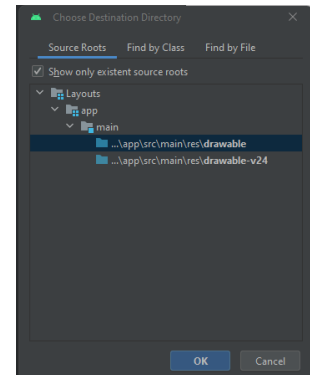
- a. En utilisant le panneau à cet effet dans l'*Attributes Window*, changez successivement les contraintes de dimensionnement à **60dp**, puis **match\_constraint** et, enfin, de retour à **wrap\_content**, changeant l'attribut `text` à chaque fois pour montrer l'impact de ces contraintes sur la taille du widget



32. Indiquez que d'autres concepts de `ConstraintLayout` seront présentés lorsque requis durant le trimestre, telles que les lignes guides et les barrières

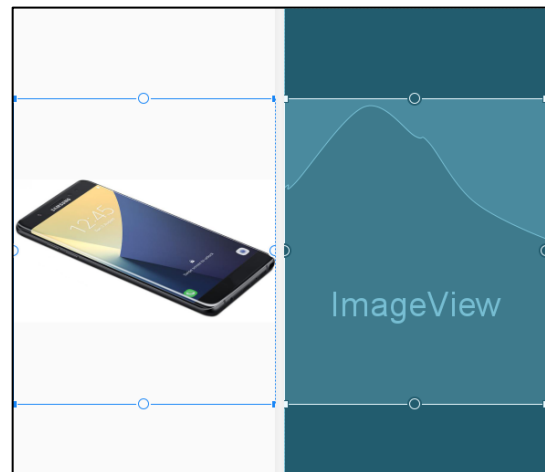
#### 4.4. Exemple d'utilisation de `ConstraintLayout`

33. Expliquez qu'on va développer l'application présentée au chapitre 28 du manuel de référence. Pour cela, supprimez tous les widgets de l'activité développée à date (plutôt que de recommencer avec un nouveau projet)
34. Demandez aux étudiants de désactiver *Autoconnection* puisque c'est un tutoriel portant sur la manipulation de `ConstraintLayout`
35. Demandez aux étudiants de télécharger de *eCité* l'image requise pour ce tutoriel : `galaxy.jpg`
36. Par copier-coller, déposez cette image dans le répertoire `app`
  - » `res` » `drawable` en conservant le même nom de fichier
  - a. Expliquez aux étudiants que des images sont intégrées dans un projet *Android Studio* par *copier-coller*, *non pas par glisser-déposer*
  - b. Soulignez qu'ils doivent coller le fichier dans le répertoire `drawable`, *non pas dans ~~drawable-v24~~*
37. Dans l'activité vierge, déposez un **`ImageView`** au centre de celle-ci, en sélectionnant `galaxy.jpg` comme contenu



38. Au besoin, redimensionnez le `ImageView` afin qu'il ressemble à ça :

  - a. Investiguez les différentes valeurs de l'attribut `scaleType` du `ImageView`, puis attribuez-lui la valeur `fitCenter`

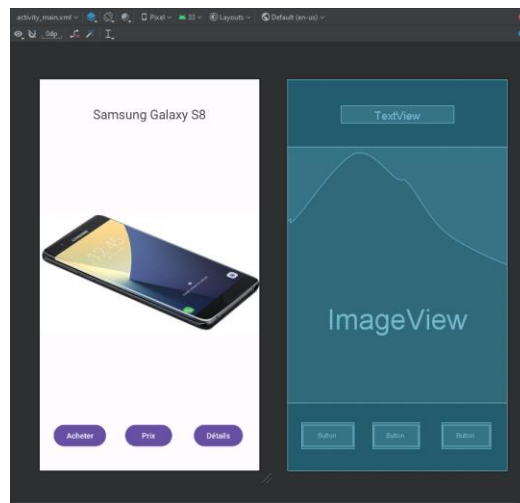


39. Déposez un `TextView` centré horizontalement au-dessus du `ImageView`, et changez les attributs suivants de celui-ci

- `textAlignment` = **center**
- `textSize` = **24sp**
- `text` = **Samsung Galaxy S8**

40. Ajoutez trois `Button` horizontalement au bas de l'activité, titrés respectivement « Acheter », « Prix » et « Détails »

41. L'activité devrait ressembler à ce qui suit



42. Démontrez, en pivotant l'appareil, que l'absence de contraintes rend l'activité inutilisable

43. Attribuez des contraintes au `TextView`

- Deux contraintes latérales au `TextView` (bias de 50%)
- Une contrainte supérieure à 24dp de la marge

44. Attribuez des contraintes au `ImageView`

- Deux contraintes verticales en le connectant au `TextView` supérieur ainsi qu'au `Button` central inférieur (50% de biais)



- b. Deux contraintes latérales aux marges gauche et droite de l'activité (50% de biais)

45. Enfin, attribuez des contraintes aux trois `Button`

- a. Sélectionnez les trois `Button`, puis la commande `Chain » Create Horizontal Chain` du menu contextuel
- b. Expliquez qu'une chaîne relie les `Button` par des *contraintes réciproques* (c.à.d. des contraintes bidirectionnelles)
- c. Les trois `Button` toujours sélectionnés, appliquez-leur la commande `Align » Vertical Centers` du menu contextuel
- d. Affichez l'activité en mode `Text` afin de déterminer quel `Button` agit comme point de référence de positionnement vertical pour les trois boutons (ça devrait être le bouton « Acheter »). Attribuez une contrainte verticale au `Button` « Acheter » de 8dp à la marge inférieure de l'activité

46. Affichez l'activité dans les deux orientations pour démontrez le bon fonctionnement des contraintes

- a. Si le `ImageView` empiète sur le `TextView` et/ou les `Button`, changez son attribut `layout_height` à `match_constraint`; expliquez qu'une taille de 0dp et `match_constraint` ont la même signification

47. Démontrez les trois types de chaînes

- a. Afficher le code XML de l'activité avec son *Preview*
- b. Soulignez les attributs `app:layout_constraintEnd_toStartOf` et `app:layout_constraintStart_toEndOf` des boutons « Acheter » et « Prix » de la chaîne. Ces deux boutons se contraignent mutuellement.

- c. Cliquez **Chains** » **Horizontal Chain Style** du menu contextuel de n'importe lequel des trois `Button`, ce qui occasionne l'ajout de l'attribut `app:layout_constraintHorizontal_chainStyle= "spread"` à celui-ci, et cliquez cet item du menu contextuel à nouveau pour faire passer successivement l'attribut de la chaîne à `spread_inside` puis `packed`
  - d. Expliquez qu'on peut aussi configurer la chaîne afin que les `Button` s'agrandissent pour combler l'espace entre ceux-ci à l'aide de poids (*weights*), mais ça ne peut être fait qu'en XML pour l'instant
48. Expliquez aux étudiants qu'on a brossé qu'un bref aperçu de `ConstraintLayout`, et qu'on n'a pas le temps d'investiguer les autres *layouts* (p.ex. `FrameLayout`, `RelativeLayout`, ...)
- a. L'auteur du manuel de référence ne les couvre pas, mais nous allons tout-de-même en voir quelques-uns durant le trimestre, selon nos besoins

**Évaluation formative 04** Indiquez aux étudiants qu'ils peuvent récupérer l'énoncé de l'exercice (en format PDF) sur le portail éducatif du collège. Ils devraient pouvoir solutionner l'exercice sans aide en se référant aux chapitres 23 à 28 du livre de référence