



# Prototypage (basse et moyenne fidélité) **et interaction**

8INF865

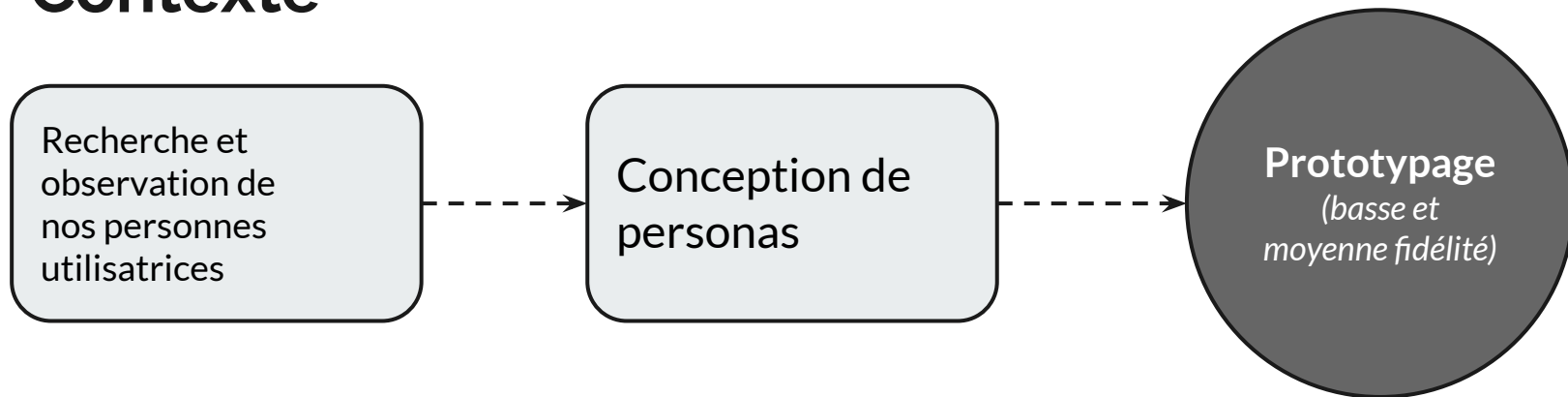
**UQAC**  
Université du Québec  
à Chicoutimi



# Ordre du jour

- Flux d'utilisation
  - Activité
- Croquis
  - Activité
- Fil de fer (maquette)
- Maquette interactive
- Concepts spéciaux

# Contexte





## Prototypage basse et moyenne fidélité



### Flux d'utilisation

Ensemble des actions et chemins possibles dans l'application



### Croquis

Remue-méninge visuel des interactions, fenêtres, configurations graphiques



### Fil de fer

Première version informatisée des interfaces et interactions



### Maquette

Version raffinée des interfaces et interactions

# Utilité (prototypage)

Identification des besoins	Conception	Développement	Tests	Après le lancement
\$1	\$10	\$100	\$1,000	\$10,000
				

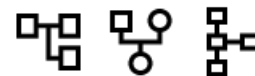
Boehm, Barry W. and Philip N. Papaccio. '**Understanding and Controlling Software Costs**'  
IEEE Transactions on Software Engineering, v. 14, no. 10, October 1988, pp. 1462-1477

---

# Flux d'utilisation

(flux utilisateur, « user flow »)

# Flux d'utilisation



Qu'est-ce que c'est ?

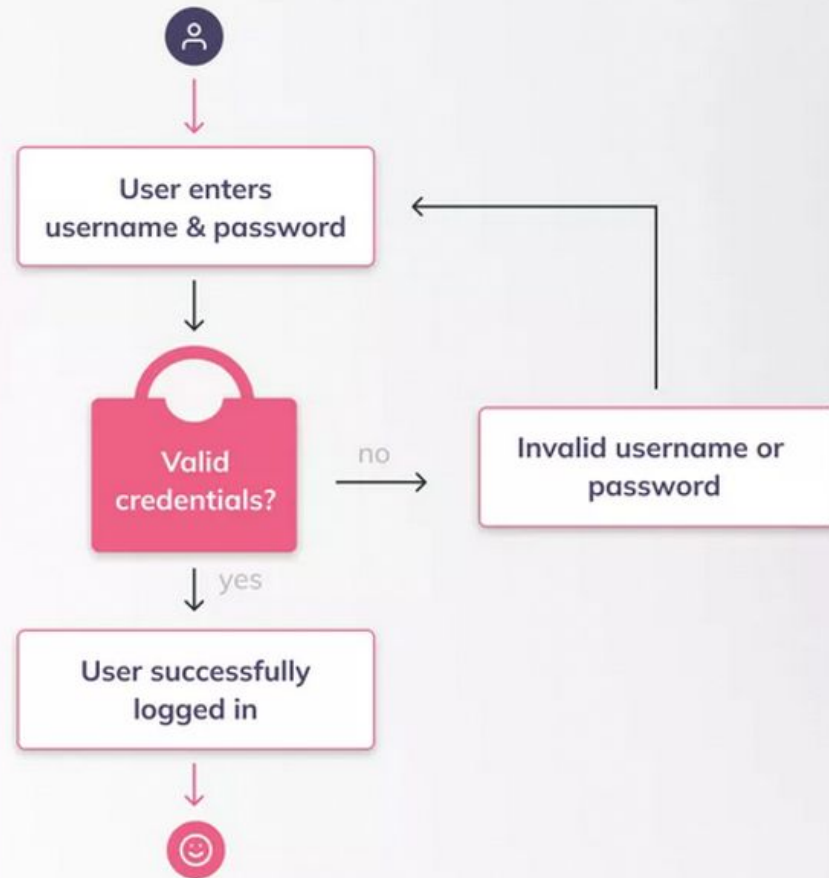
Représentation **schématique** de l'ensemble des chemins et interactions que **peut** faire une personne utilisatrice dans une application ou section d'application

→ Traditionnellement, un diagramme de flux

*N'inclut pas de détail par rapport aux fenêtres, pages ou autre*

# Flux d'interaction (exemple)

Identification





# Flux d'utilisation (exemple)

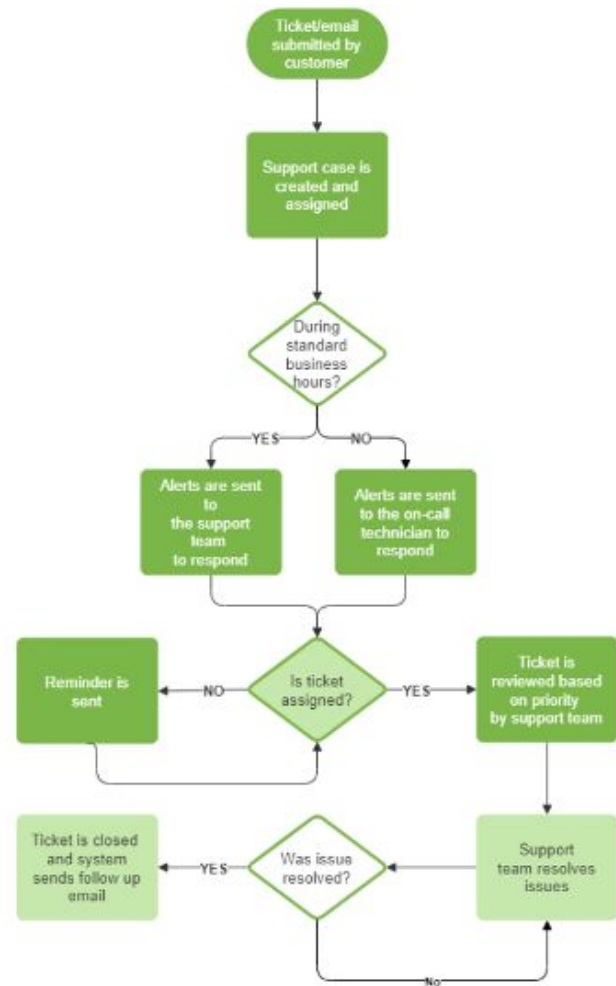
## Application de musique

<https://dribbble.com/shots/6118359-Music-player-app-design-system-user-flow/attachments/1312292>



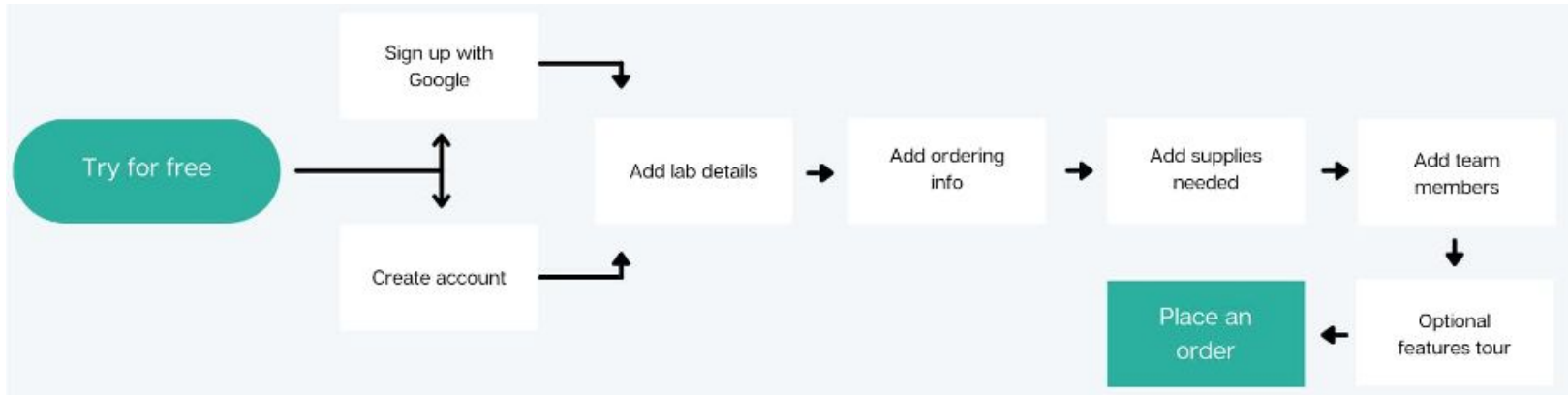
# Flux d'utilisation (exemple)

Global vs Spécifique



# Flux d'utilisation (exemple)

Accent sur le comportement, l'état et les conditions (et NON sur les fenêtres où se situe l'interaction)




<https://www.appcues.com/blog/user-flow-examples>



## Utilité (flux d'utilisation)

1. Permet d'identifier rapidement les éléments qui devront être implémentés (ou pas)
2. Permet d'identifier des points de frustration **tôt** et d'éviter d'avoir à faire des changements plus tard une fois que la conception est plus avancée
3. Facilite la création d'une interface familière et utilisable



## Chemin idéal (flux d'utilisation)

Un **chemin idéal** (« happy path ») est spécifique à une persona et sa tâche à accomplir

- Chemin **optimal** dans l'application tant pour l'utilisateur que pour l'entreprise
- Chemin dans lequel il y a le **moins de friction** pour accomplir une tâche
- Permettra plus tard d'identifier les éléments interactifs qui doivent être mis de l'avant pour faciliter l'atteinte des objectifs

# Flux d'utilisation



## Comment le créer ?

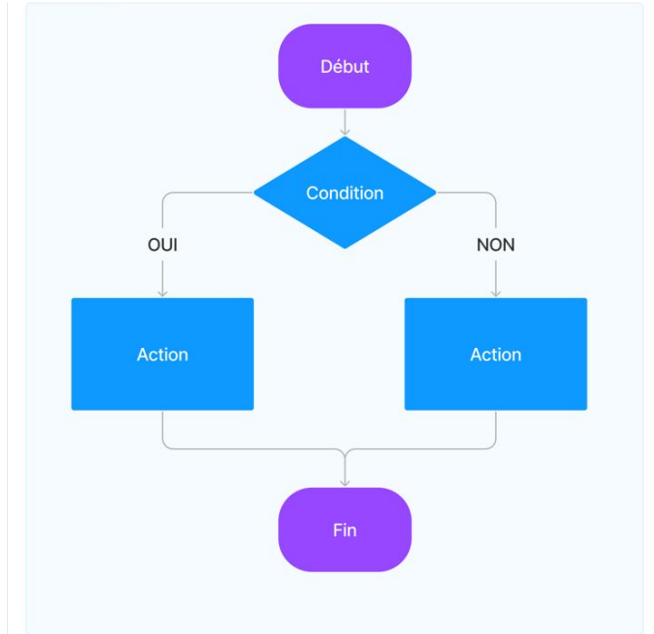
1. Déterminer les objectifs, besoins, frustrations et comportements des personnes utilisatrices
2. Établir une liste de tâches, fonctionnalités ou autre qui devraient correspondre à ces éléments
3. Identifier les meilleures pratiques du moment
4. Commencer à dessiner (à la main)
5. Mettre l'accent sur le **chemin idéal**
6. Reproduire informatiquement

# Flux d'utilisation

## Symboles les plus fréquents

(Restez simple)

- **Cercle ou rectangle arrondi**  
Début / fin du diagramme
- **Losange**  
Décision / Condition
- **Rectangle**  
Action / État





# Figma + FigJam

Outil de dessin et conception d'interface collaboratif

Entièrement gratuit pour l'éducation

<https://www.figma.com/fr/education>

- Flux d'utilisation → **FigJam**  
<https://www.youtube.com/watch?v=DNBlcBdKnQo> (tutoriel vidéo)
- Maquette fil de fer, maquette interactive → **Figma**



---

# Croquis

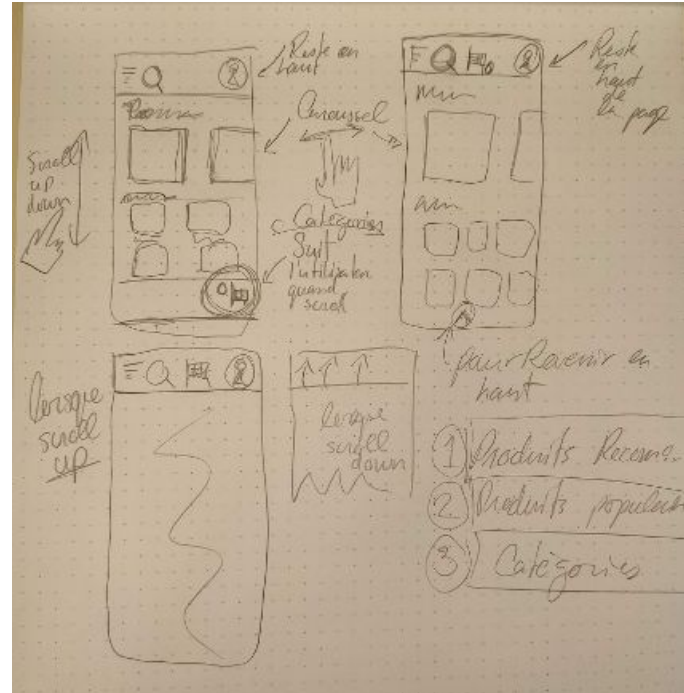
(« sketching »)

# Croquis

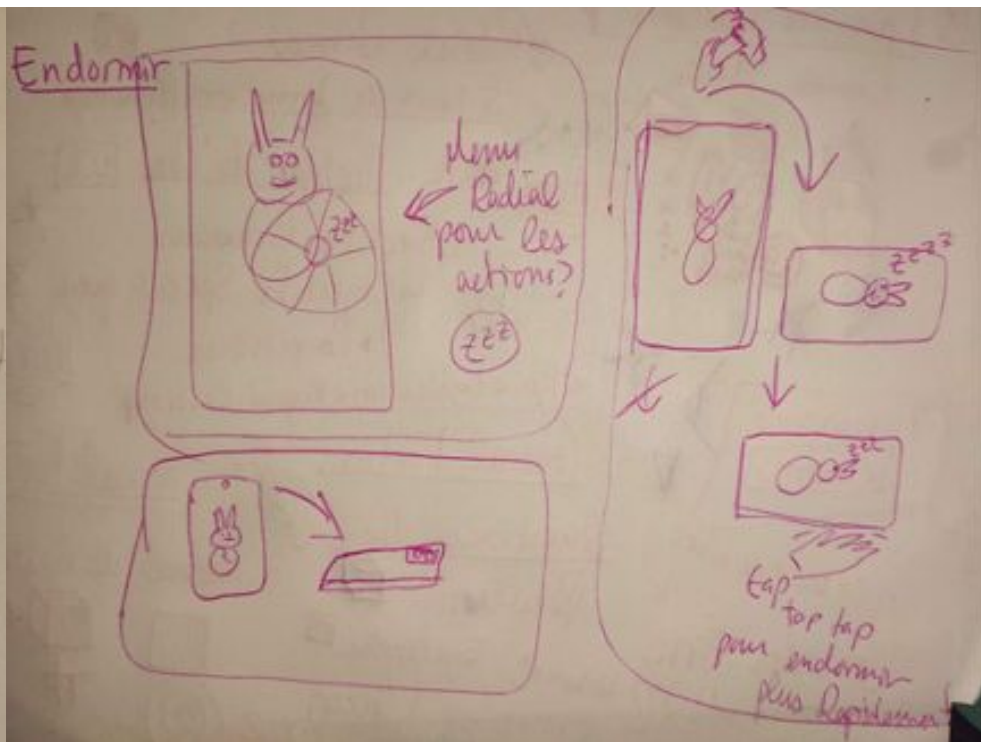
Qu'est-ce que c'est ?

**Premier jet d'une interface/technique d'interaction dessinée à main levée**

Permet d'explorer différentes manières d'interagir avec un système informatique (application mobile) ou d'en présenter l'interface



Dessins du Professeur Pascal Fortin



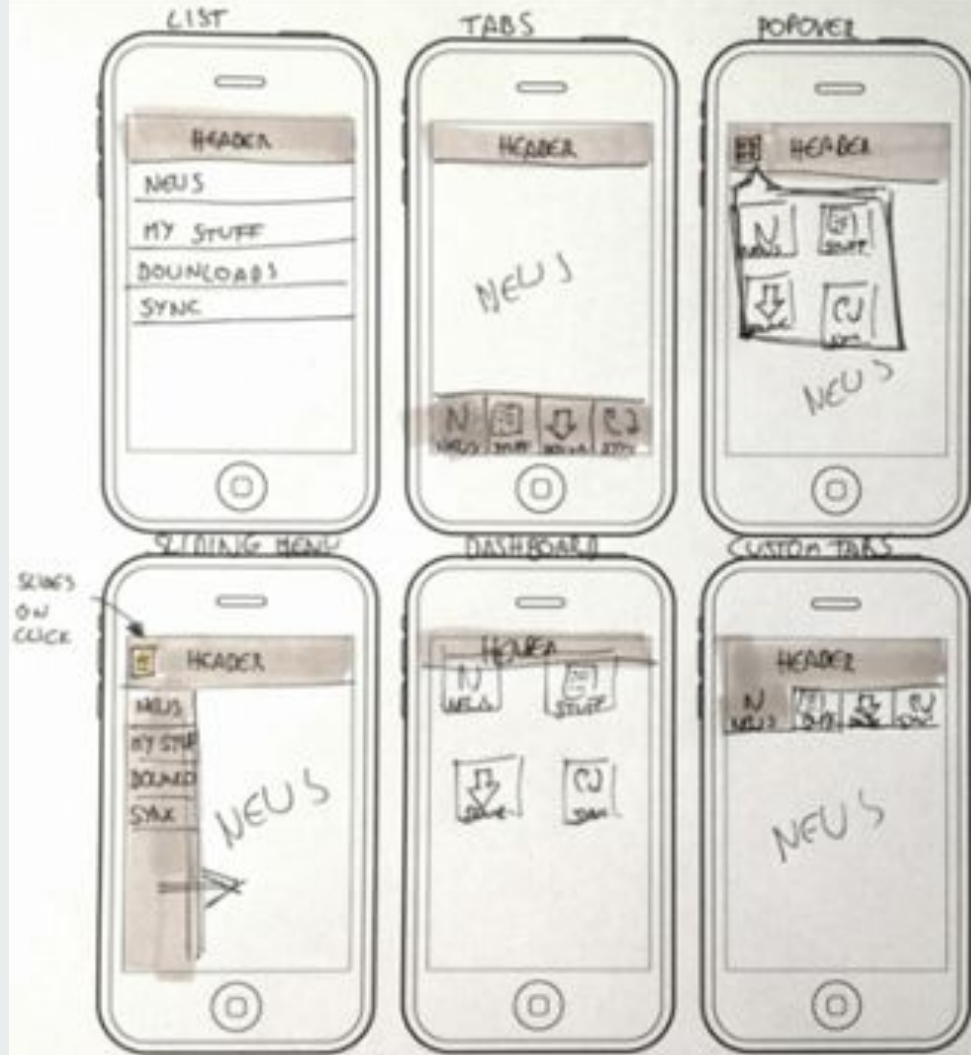
Dessins du Professeur Pascal Fortin

Croquis (exemple)

# Croquis (exemple)

6x le même menu

<https://www.smashingmagazine.com/2013/06/sketching-for-better-mobile-experiences>






## Comment cela fonctionne ? (Croquis)

1. Choisir un élément d'interaction, une partie d'interface graphique ou une fenêtre entière qui devra être réalisée
2. Produire plusieurs croquis explorant différentes manières d'interagir avec l'élément ou de positionner les constituants de l'interface
3. Discuter des options en groupe, réfléchir à comment elles répondent aux besoins, attentes, frustrations de nos personnes utilisatrices
4. Choisir une des options ou combiner divers aspects de plusieurs croquis afin de produire un croquis final

→ Approche  
**divergente-convergente**

Répéter les étapes 1 à 4,  
pour chaque **fenêtre ou  
fonctionnalité critique**



## Utilité (croquis)



*Réflexion sur le problème*

*Expérimentation*

*Exploration*



*Relation entre les éléments*

*Sans condition*



*Très rapide*



# Croquis



## Important

- Rapide
- Jetable
- Nombreux
- Minimaliste



## Inutile/interdit

- Détails
- Esthétique

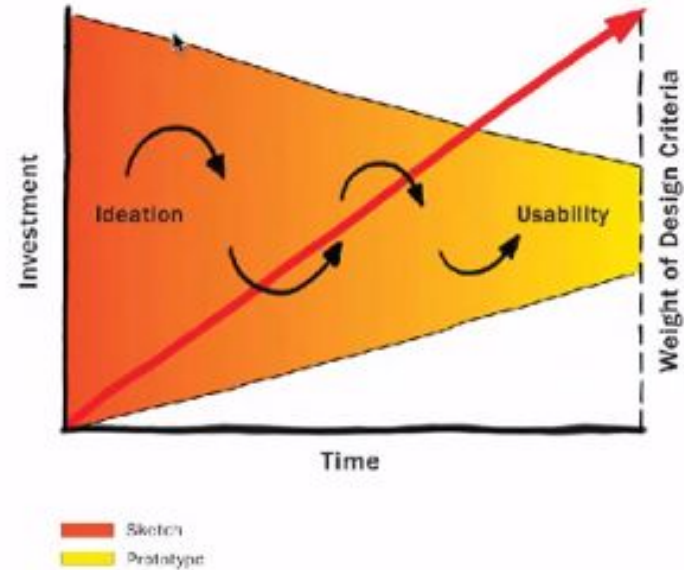
**“Your sketches don’t need to look good; they only have to convey your idea.”**

Lennart Hennigs





<u>SKETCH</u>		<u>PROTOTYPE</u>
EVOCATIVE	→	DIDACTIC
SUGGEST	→	DESCRIBE
EXPLORE	→	REFINE
QUESTION	→	ANSWER
PROPOSE	→	TEST
PROVOKE	→	RESOLVE
TENTATIVE	→	SPECIFIC
NONCOMMITTAL	→	DEPICTION



Buxton, 2007 – Sketching User Experiences

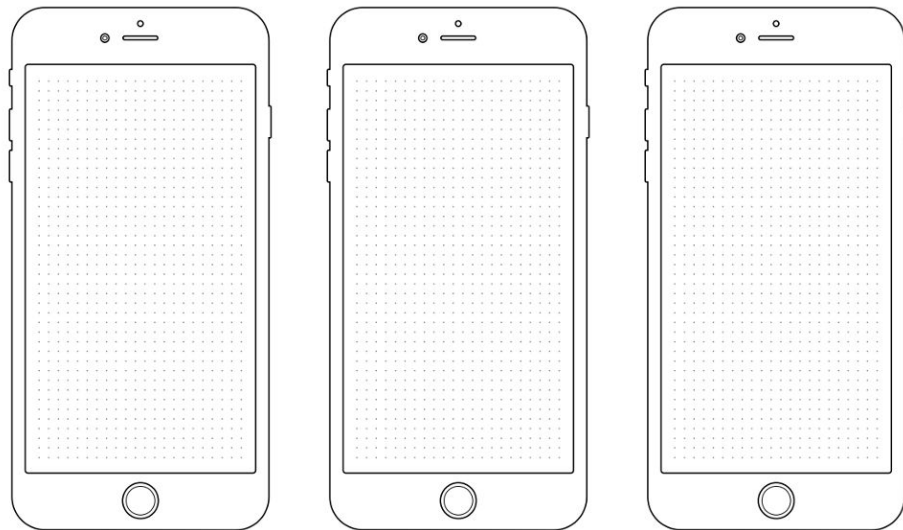
Croquis vs Prototype (haute fidélité)

# Croquis

Gabarits de téléphones  
intelligents ou tablettes

(ex., pour vos projets)

<https://www.sketchize.com>



---

# Fil de fer (maquette)

(« wireframe »)



## **Qu'est-ce que c'est ?** (maquette fil de fer)

Un schéma ou un autre rendu de basse fidélité destiné à démontrer :

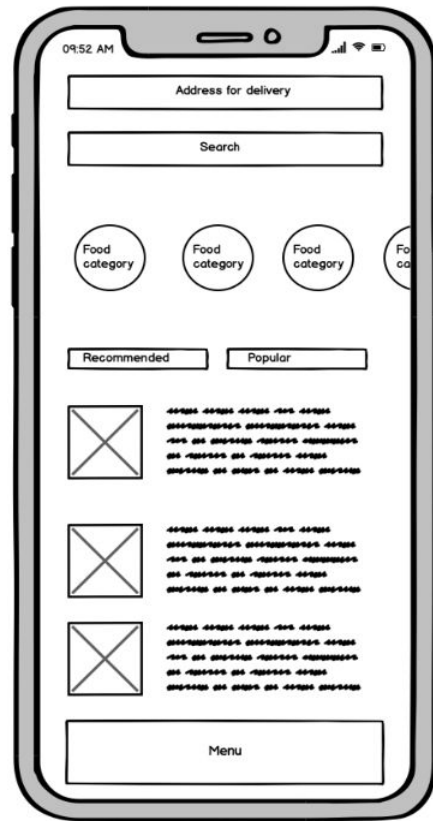
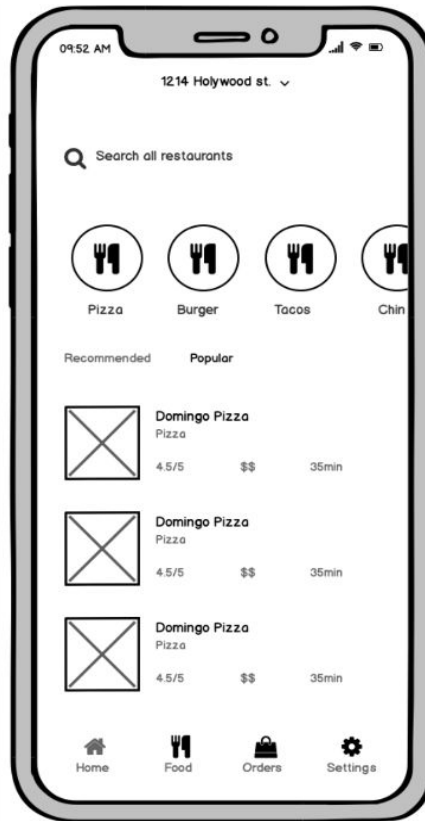
- la fonctionnalité
- les caractéristiques
- le contenu
  - hiérarchie et/ou structure de l'information
- le flux d'utilisation (utilisateur)

# Fil de fer (maquette)

## Exemple

<https://balsamiq.com/learn/articles/wireframing-mobile-applications>

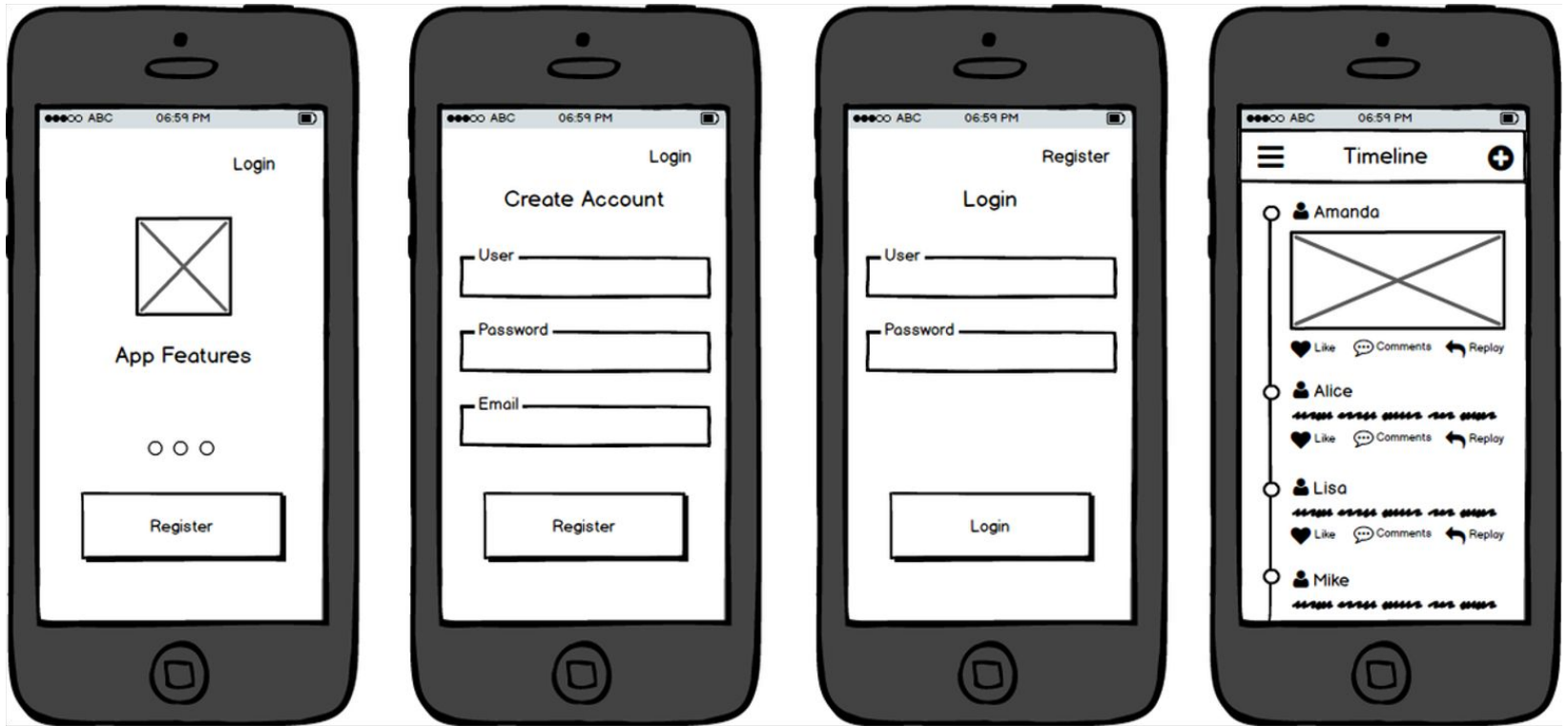
Plus ou moins détaillé





<https://www.mockplus.com/blog/post/mobile-app-wireframing-guide>

Exemple (maquette fil de fer)



<https://soulless.medium.com/the-5-best-free-wireframe-tools-for-mobile-apps-you-cant-miss-out-394bf115a027>

Exemple (maquette fil de fer)



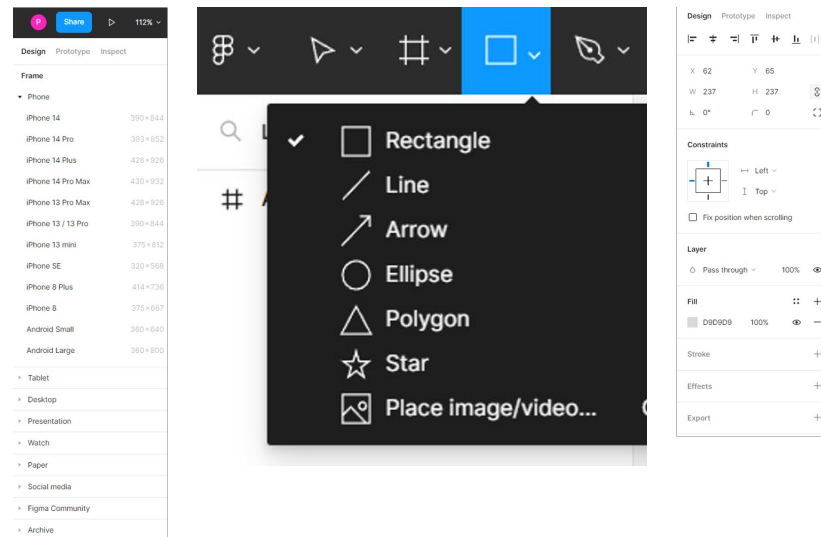
## **Particularités** (maquette fil de fer)

- Surtout visuel, avec peu de texte et des annotations
- Utilisation de blanc/gris/noir presque exclusivement
- Utilisation d'espaces réservés (« placeholders »)
- Fidélité basse (moyenne)
- Pas ou peu d'interactivité



# Figma

1. Créer un Frame sur lequel on pourra dessiner, une fenêtre = un Frame
2. Ajouter des éléments à notre fenêtre
3. Écrire du texte
4. Dessiner des formes
5. ...





## Figma (Ressource)

Chaîne (vidéo) de tutoriels d'Aliena Cai (5x12 min)

[https://www.youtube.com/playlist?list=PLKId0A0XCibUYx3c\\_NYn13W9Z\\_kkliA2m](https://www.youtube.com/playlist?list=PLKId0A0XCibUYx3c_NYn13W9Z_kkliA2m)

Butter Academy (1 h)


[https://www.youtube.com/watch?v=6t\\_dYhXyYjI](https://www.youtube.com/watch?v=6t_dYhXyYjI)

Documentation et aide officiel

<https://www.figma.com/resource-library/design-basics>

---

# Maquette interactive



## Qu'est-ce que c'est ? (Maquette interactive)

Une maquette avec des zones dans lesquelles on peut cliquer pour simuler la navigation et les interactions avec une application (**sans code**)



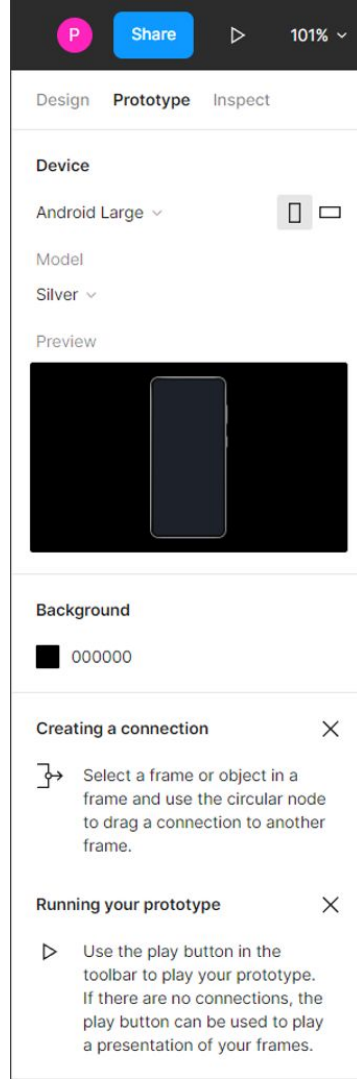
## Pourquoi ? (Maquette interactive)

1. Expérimenter avec la navigation de l'application avant d'avoir à faire le développement
2. Faire tester la navigation à une ou plusieurs personnes afin d'en tester l'utilisabilité, par exemple :
  - a. Est-ce que cliquer à un endroit a le résultat attendu ?
  - b. Est-ce que les éléments sont là où l'utilisatrice ou l'utilisateur s'attend à les retrouver ?
  - c. Est-ce qu'il y a de la confusion pour réaliser le chemin idéal ?
3. Concrétiser comment les interactions seront implémentées

# Figma

## (Maquette interactive)

1. Cliquez sur l'onglet Prototype pour passer en mode prototypage
2. Cliquez sur un élément
3. Cliquez sur le + et gardez le bouton enfoncé jusqu'à la destination de l'interaction

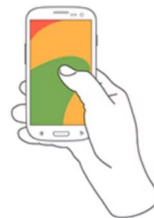


---

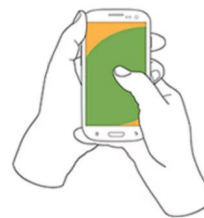
# Caractéristiques spéciales

# Concevoir pour les pouces

- <https://bootcamp.uxdesign.cc/utilizing-the-thumb-zone-for-dropdowns-e579f0f9185a>
- <https://www.peerbits.com/blog/mobile-app-designing-rules.html>



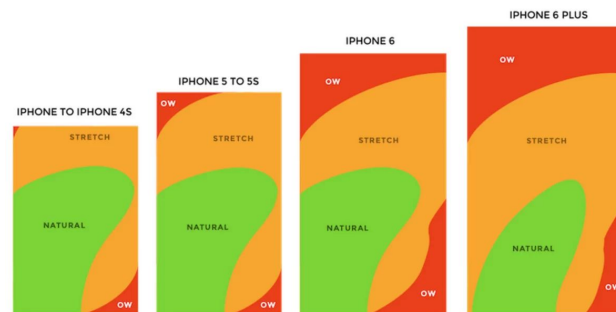
**49%**  
one handed



**36%**  
cradled



**15%**  
two handed



Thumb  
8mm target  
2mm boundary

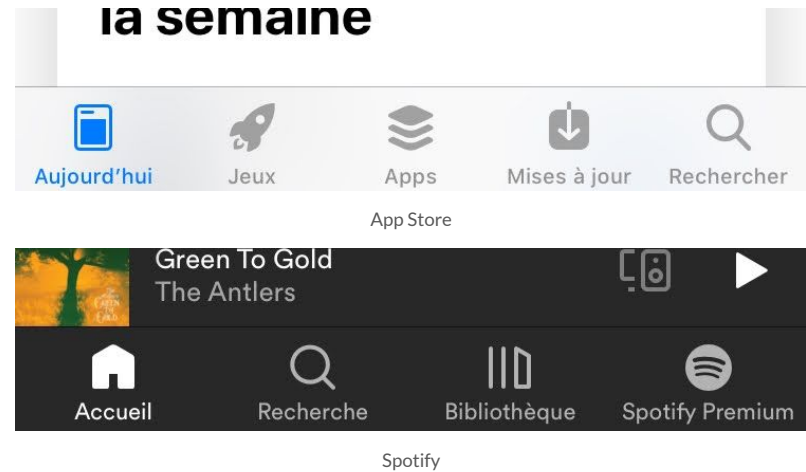


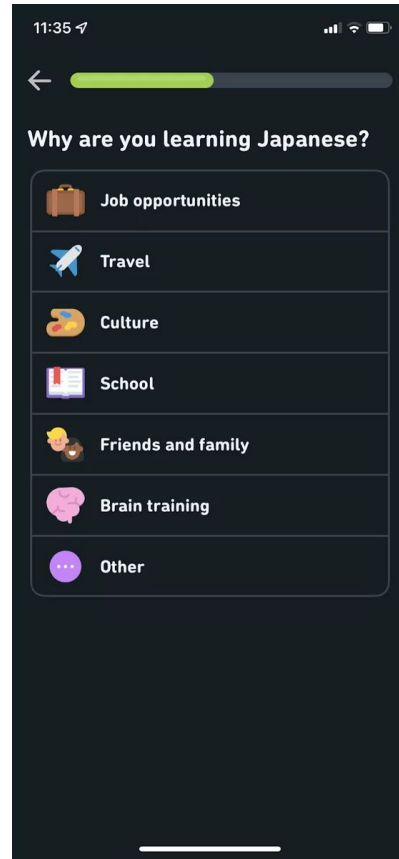
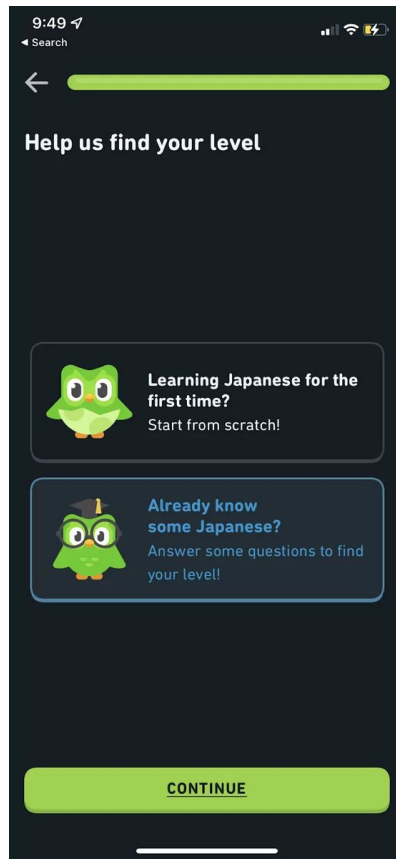
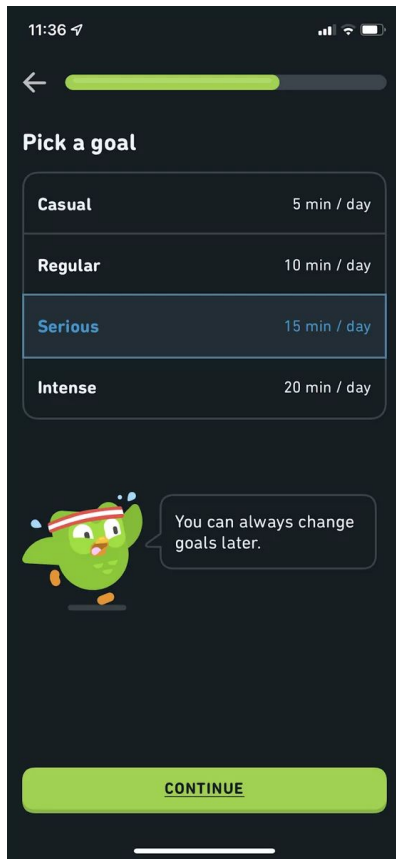
Finger  
7mm target  
1mm boundary



# Navigation par le bas

- Permet de naviguer avec les pouces
- Accès direct aux sections spécifiques
- Requiert moins d'effort

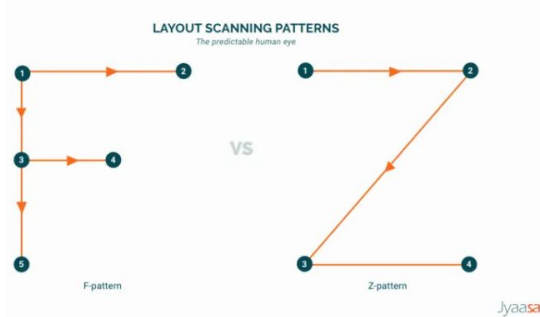




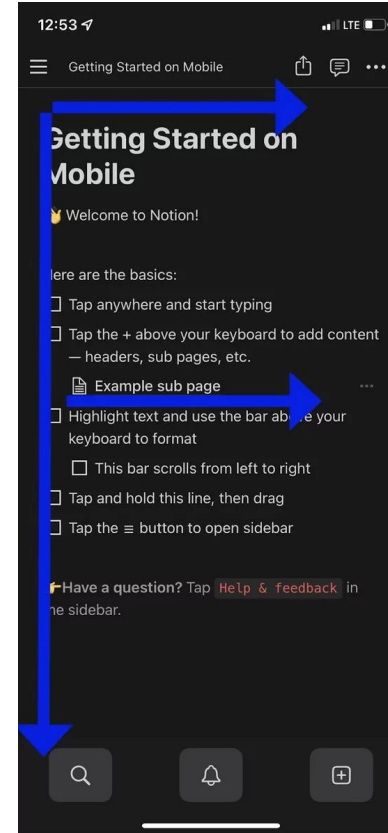
Application  
Duolingo

Séparer (l'interface) et utiliser plusieurs fenêtres simples

# Modèles de visualisation F & Z



<https://lineindesign.medium.com/be-a-designer-who-can-also-help-with-writing-copy-2f4ea02a5646>



Notion (application)



Netflix (application)



## À éviter

- Incohérence de l'interface
- Navigation confuse dans l'application
- Manque de clarté entre les graphiques et le texte
- Fonctionnalités cachées, inconnues et impossibles à trouver par les personnes utilisatrices
- Incohérence linguistique et fautes d'orthographe
- Omission de la politique de confidentialité
- Manque d'aide disponible pour l'application
- L'application plante sans avertissement
- Distorsion avec l'orientation de l'écran
- Absence de notification concernant la connectivité réseau
- Informations techniques sur les messages d'erreur et difficulté à réparer (reprendre après) les erreurs

**If you want a golden rule that will fit everybody, this is it: Have nothing in your ~~houses~~ [mobile app] that you do not know to be useful, or believe to be beautiful.**

- William Morris "The Beauty of Life," 1880



---

# Projet (préparation)

(~ 5 minutes)



# Préparation (obligatoire)

1. Créez un dépôt GitHub (**UN SEUL PAR ÉQUIPE**)
  - Public (important pour avoir accès aux pages Wiki)
  - Ajouter un fichier README
  - Ajouter un fichier .gitignore pour Android
2. Ajouter **vos prénoms et noms** (ceux l'équipe) dans le **README** puis commit/push
3. Envoyer par courriel le lien du dépôt à [dbrun@ugac.ca](mailto:dbrun@ugac.ca)

---

**Wiki** (de GitHub)



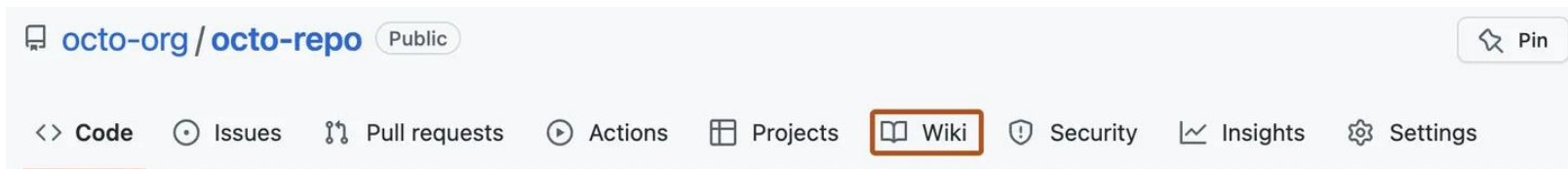
# Wiki

<https://docs.github.com/en/communities/documenting-your-project-with-wikis/about-wikis>

Idéal pour la documentation

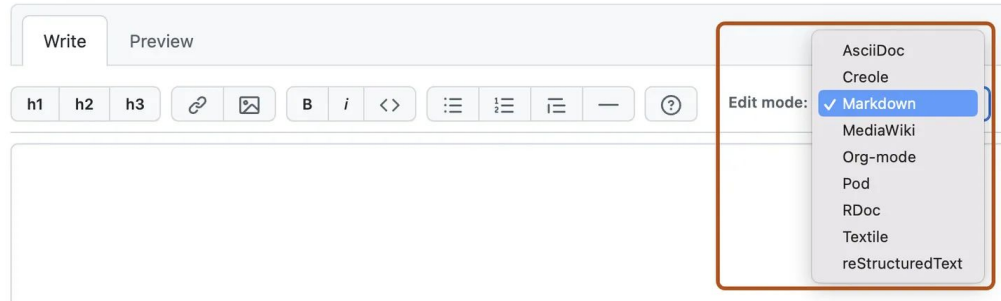
→ *disponible gratuitement sur GitHub avec les dépôts ouvert publiquement*

- Possibilité de personnaliser la barre latérale et le pied de page
- Historique des modifications



# Markdown

Langage de balisage simple et facile à utiliser pour formater pratiquement n'importe quel document.



<https://www.markdownguide.org>

# Markdown

## Cheat Sheet

### Syntaxe basique

<u>Heading</u>	# H1 ## H2 ### H3
<u>Bold</u>	<b>**bold text**</b>
<u>Italic</u>	<i>*italicized text*</i>
<u>Blockquote</u>	> blockquote
<u>Ordered List</u>	1. First item 2. Second item 3. Third item
<u>Unordered List</u>	- First item - Second item - Third item
<u>Code</u>	<code>`code`</code>
<u>Horizontal Rule</u>	---
<u>Link</u>	[title](https://www.example.com)
<u>Image</u>	![alt text](image.jpg)

# Markdown Cheat Sheet

## Syntaxe étendue

<u>Table</u>	<pre>  Syntax   Description     -----   -----     Header   Title     Paragraph   Text  </pre>
<u>Fenced Code Block</u>	<pre>``` {   "firstName": "John",   "lastName": "Smith",   "age": 25 } ```</pre>
<u>Footnote</u>	<pre>Here's a sentence with a footnote. [^1]  [^1]: This is the footnote.</pre>
<u>Heading ID</u>	<pre>### My Great Heading {#custom-id}</pre>
<u>Definition List</u>	<pre>term : definition</pre>
<u>Strikethrough</u>	<pre>~~The world is flat.~~</pre>
<u>Task List</u>	<pre>- [x] Write the press release - [ ] Update the website - [ ] Contact the media</pre>
<u>Emoji</u>	<pre>That is so funny! :joy:</pre>
<u>Highlight</u>	<pre>I need to highlight these ==very important words==.</pre>
<u>Subscript</u>	<pre>H~2~0</pre>
<u>Superscript</u>	<pre>X^2^</pre>

# Démonstration (~ 2 minutes)

<https://github.com>

---

---

# Kotlin

## Condition (if, else)

```
val guests = 30
if (guests == 0) {
    println("No guests")
} else if (guests < 20) {
    println("Small group of people")
} else {
    println("Large group of people!")
}
```

```
if( condition 1 ) {
    body 1
} else if( condition 2 ) {
    body 2
} else {
    body 3
}
```

# Condition (when, in, is)

```
when (x) {  
  0 -> println("x is just 0.")  
  2, 3, 5, 7 -> println("x is a prime number between 1 and 10." )  
  in 1..10 -> println("x is a number between 1 and 10, but not a prime number." )  
  is Int -> println("x is an integer number, but not between 1 and 10." )  
  else -> println("x isn't an integer number." )  
}
```

```
when ( parameter ) {  
  condition 1 , condition 2 -> body 1 & 2  
  condition 3 -> body 3  
}
```

```
when ( parameter ) {  
  in range start . range end -> body 1  
  condition 2 -> body 2  
}
```

```
when ( parameter ) {  
  is type -> body 1  
  condition 2 -> body 2  
}
```



## Condition (if, else, when) en affectation

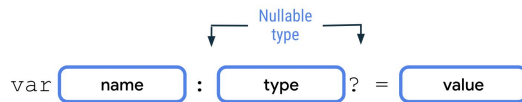
```
val lightColor = "Black"
val message =
    if (lightColor == "Red") "Stop"
    else if (lightColor == "Yellow") "Slow"
    else if (lightColor == "Green") "Go"
    else "Invalid light color"
println(message)
```

```
val lightColor = "Amber"
val message = when(lightColor) {
    "Red" -> "Stop"
    "Yellow", "Amber" -> "Slow"
    "Green" -> "Go"
    else -> "Invalid light color"
}
println(message)
```

# Variable « Nullable » et Null safety

Par défaut, avec Kotlin les variables ne peuvent pas être null

- ? déclare la variable comme « nullable »  
`var numberOfBooks: Int? = 6`
- Opérateur « sécurisé » ?. pour accéder à la propriété de la variable  
`n = numberOfBooks?.dec()`
- Opérateur d'assertion non null !! pour affirmer que la valeur de la variable n'est pas null (attention, peut générer `NullPointerException`)  
`n = numberOfBooks!!.dec()`
- Opérateur Elvis ?: pour ajouter une valeur par défaut si null  
`n = numberOfBooks?.dec() ?: 0`



<https://kotlinlang.org/docs/null-safety.html>

# Classe

```
class SmartDevice {  
    val name = "Android TV"  
    val category = "Entertainment"  
    var deviceStatus = "online"  
  
    fun turnOn() {  
        println("Smart device is turned on.")  
    }  
    fun turnOff() {  
        println("Smart device is turned off.")  
    }  
}  
...  
val smartTvDevice = SmartDevice()  
smartTvDevice.turnOn()
```

```
class name {  
    body  
}
```

```
val name = ClassName ()
```

```
classObject . methodName ( [Optional] Arguments )
```

## Getter/Setter (des propriétés d'une classe) 1/2

```
class SmartDevice {  
    var speakerVolume = 2  
    get() = field  
    set(value) {  
        if (value in 0..100) {  
            field = value  
        }  
    }  
    ...  
}
```

```
var name : data type = initial value  
  
get() {  
    body  
    return statement  
}  
  
set(value) {  
    body  
}
```

## Getter/Setter (des propriétés d'une classe) 2/2

Par défaut

```
class Person(var name: String)
```

```
fun main() {  
    val person = Person("Alex")  
    println(person.name)  
    person.name = "Joey"  
    println(person.name)  
}
```

Access with `.<property name>`

Set with `.<property name>`

# Héritage

- Par défaut, les classes Kotlin sont *final*
  - elles ne peuvent pas être héritées.
- Pour rendre une classe héritable, marquez-la avec le mot-clé `open`

```
open class Shape {  
    open fun draw() { /*...*/ }  
    fun fill() { /*...*/ }  
}  
  
class Circle() : Shape() {  
    override fun draw() { /*...*/ }  
}
```



# Interface

```
interface Fax {  
    fun call(number: String) = println("Calling $number")  
    fun print(doc: String) = println("Fax:Printing $doc")  
    fun answer()  
}  
  
class MultiFunction : Fax {  
    override fun answer () { // implémenter answer  
    }  
}
```

`override` permet de  
redéfinir la méthode  
d'une interface (ou  
d'une classe mère)



# Visibilité

**Public** – Visible en dehors de la classe. Tout est public par **défaut**, incluant les variables et méthodes d'une classe

**Private** – Visible seulement dans la classe

**Protected** – Visible seulement dans la classe et les subclasses

**Internal**

- Alternative au package-private de Java
- Signifie que les déclarations sont visibles à l'intérieur d'un module

*Un module étant un ensemble de fichiers Kotlin compilés ensemble*



# Fonction comme type de donnée

```
fun main() {  
    val trickFunction = ::trick  
}  
fun trick() {  
    println("No treats!")  
}
```

::

function name

Utilisation de l'opérateur de référence ::

# Fonction (avec une expression) lambda

```
fun main() {  
    val trickFunction = trick  
}  
val trick = {  
    println("No treats!")  
}
```

```
val treat: () -> Unit = {  
    println("Have a treat!")  
}
```

```
val variable name = {  
    function body  
}
```

```
( parameters (optional) ) -> return type
```

# Renvoyez une fonction

```
val trick = {  
    println("No treats!")  
}  
val treat = {  
    println("Have a treat!")  
}
```

```
fun trickOrTreat(isTrick: Boolean): () -> Unit {  
    if (isTrick) {  
        return trick  
    } else {  
        return treat  
    }  
}
```

```
fun function name () : function type {  
    // code  
    return Name of another function  
}
```

# Transmettre une fonction en tant qu'argument

```
val trick = {  
    println("No treats!")  
}  
val treat = {  
    println("Have a treat!")  
}
```

```
fun trickOrTreat(isTrick: Boolean, extraTreat: (Int) -> String): () -> Unit {  
    if (isTrick) {  
        return trick  
    } else {  
        println(extraTreat(5))  
        return treat  
    }  
}
```

Diagram illustrating the structure of a function definition:

```
val function name = { parameter 1 , parameter 2 ->  
    function body  
}
```

## Expression lambda abrégée (1/3)

- Omettre le nom du paramètre

```
val coins: (Int) -> String = { quantity ->  
    "$quantity quarters"  
}
```



```
val coins: (Int) -> String = {  
    "$it quarters"  
}
```

## Expression lambda abrégée (2/3)

- Transmettre une expression lambda directement à une fonction

```
var coins = Lambda expression
```

```
trickOrTreat(false, coins)
```



```
trickOrTreat(false, Lambda expression )
```

## Expression lambda abrégée (3/3)

- Utiliser la syntaxe lambda de fin

`trickOrTreat(false, Lambda expression )`



`trickOrTreat(false) Lambda expression`

---

# Bases de Compose

Annotations

Mise en page

Modificateurs





# Jetpack Compose

- Kit d'outils permettant de créer des interfaces Android native
- Fonctions déclaratives
- Fonctions modulables
  - Text, Image, Button, Surface, Spacer...

```
// ...  
import androidx.compose.runtime.Composable  
  
class MainActivity : ComponentActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContent {  
            MessageCard("Android")  
        }  
    }  
}  
  
@Composable  
fun MessageCard(name: String) {  
    Text(text = "Hello $name!")  
}
```

# Annotations

Prefix character: @

Annotation

@Composable

fun Greeting(name: String, modifier: Modifier) {}

Function declaration

```
44 ✱ @Preview(showBackground = true)
45 @Composable
46 fun GreetingPreview() {
47     HappyBirthdayTheme {
48         Greeting(name: "Android")
49     }
50 }
```

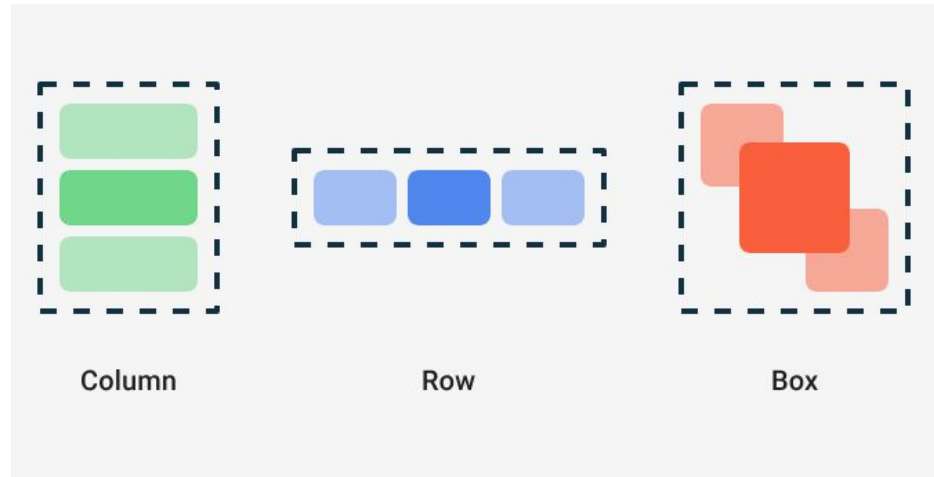
GreetingPreview

Hello Android!

- Appliquée en ajoutant le caractère @
- @Composable pour signaler (annoter) une fonction Compose (pour définir l'interface)
- @Preview pour prévisualiser une fonction Compose directement dans Android Studio

# Mise en page (layout)

- 3 éléments standards de mise en page
- Imbricables
- Arguments de positionnements (ex., `verticalAlignement`, `horizontalAlignement`, `verticalArrangement`, `horizontalArrangement`)





# Column (mise en page)

Positionnement vertical

```
@Composable
fun ArtistCardColumn() {
    Column {
        Text("Alfred Sisley")
        Text("3 minutes ago")
    }
}
```

**Alfred Sisley**  
3 minutes ago

# Row (mise en page)

## Positionnement horizontal

```
@Composable
fun ArtistCardRow(artist: Artist) {
    Row(verticalAlignment = Alignment.CenterVertically) {
        Image(bitmap = artist.image, contentDescription = "Artist image")
        Column {
            Text(artist.name)
            Text(artist.lastSeenOnline)
        }
    }
}
```



**Alfred Sisley**

3 minutes ago

# Box (mise en page)

## Positionnement horizontal

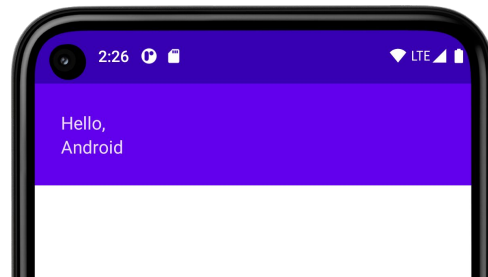
```
@Composable
fun ArtistAvatar(artist: Artist) {
    Box {
        Image(bitmap = artist.image, contentDescription = "Artist image")
        Icon(Icons.Filled.Check, contentDescription = "Check mark")
    }
}
```



# Modificateurs

Apporte des éléments décoratifs ou améliore un composable (ex., pour modifier une taille, pour ajouter des informations telles que des libellés...)

```
@Composable
private fun Greeting(name: String) {
    Column(modifier = Modifier
        .padding(24.dp)
        .fillMaxWidth()
    ) {
        Text(text = "Hello, ")
        Text(text = name)
    }
}
```





# Modificateurs

- Attention, l'ordre des modificateurs a de l'importance
- Modificateurs (de positionnement) « communs »
  - `size`, `requiredSize`, `fillMaxHeight`, `fillMaxWidth`
  - `padding`, `paddingFromBaseline`, `matchParentSize`, `offset`, `align`
- Liste des modificateurs de Jetpack Compose

<https://developer.android.com/develop/ui/compose/modifiers-list>



---

# Interactions (et états)

*En autonomie (exercices notés)*

*À commencer à la fin de la séance*



# Exercices

(tutoriels officiels)

Comprendre les interactions et  
les états avec Jetpack  
Compose

1. Créez une branche à **partir de votre branche Main** et nommez-la **TipTime**
2. Effectuer les exercices dans l'ordre, le second commence à la suite du premier
  - a. <https://developer.android.com/codelabs/basic-android-kotlin-compose-using-state?hl=fr#0>
    - Essayez de commit/push après les étapes 3, 6, 8, 10
  - b. <https://developer.android.com/codelabs/basic-android-kotlin-compose-calculate-tip?hl=fr#0>
    - Essayez de commit/push après les étapes 3, 4, 5, 6, 8

**EN AUCUN CAS, commit/push juste au début et à la fin**

# Pour la prochaine séance

- **Exercices** (tutoriels officiels) à **finir**
  - **Page de présentation des équipes** (sur le Wiki de votre dépôt du projet - voir critères sur Moodle, *bientôt*)
  - **Réfléchir en équipe à des idées de projet**
-



# Idées d'activités d'enrichissement personnel

- Tenter de recréer un flux d'utilisation associé à une application que vous aimez particulièrement
- Utiliser Figma pour recréer les maquettes fil de fer de quelques fenêtres d'une application que vous utilisez couramment (incluez des éléments interactifs)
- Choisissez un élément dans une application que vous utilisez et essayez de réaliser au moins 5 croquis représentant différentes manières de faire ou présenter cet élément