
Centre Universitaire Abdelhafid Boussouf - Mila
Master STIC
Développement avancé des applications mobiles
Développement Avancé
des Applications Mobiles
Dr. MEGUEHOUT Hamza

Contactez-moi

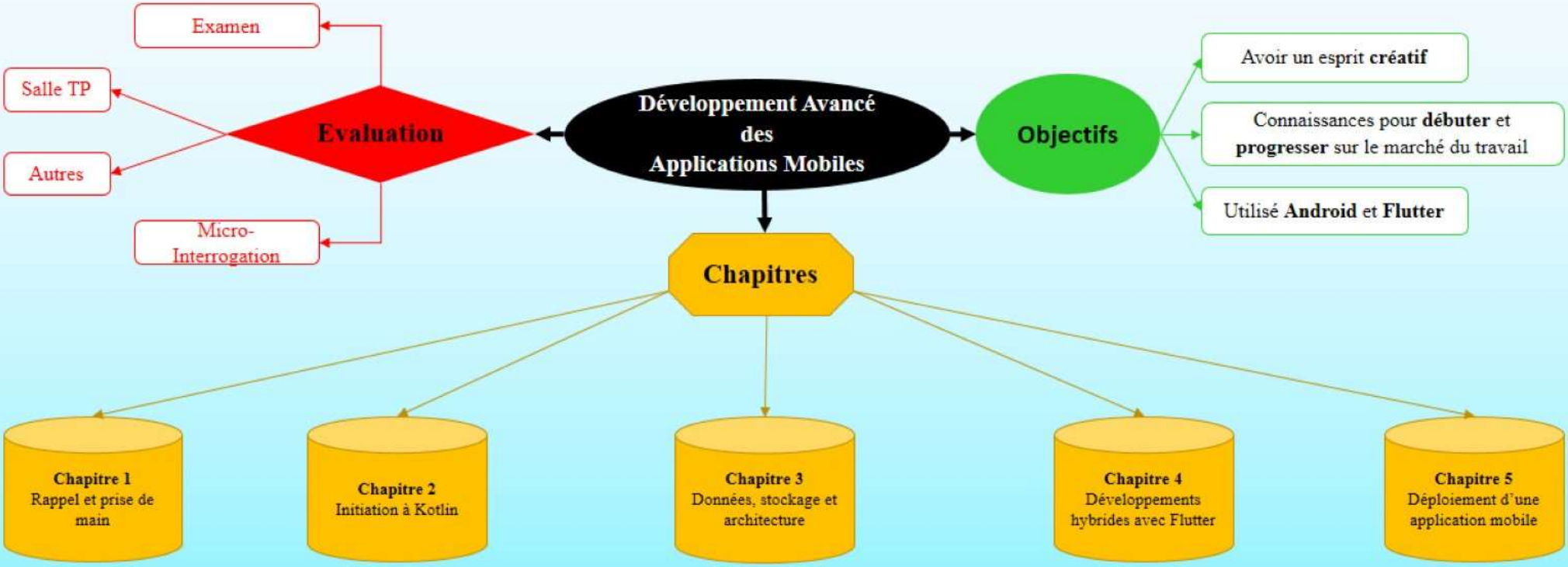
Nom : Meguehout
Prénom : Hamza
E-mail : meguehout.h@centre-univ-mila.dz
meguehout.hamza@gmail.com
Bureau : 29 Tranche 03

Collecte des E-mails

Information Matière

Unité d'enseignement fondamentale : UEM1
Crédits : 4
Coefficient : 2

Mind Map


Examen
Salle TP
Evaluation
Autres
Micro-Interrogation
Développement Avancé des Applications Mobiles
Objectifs
Avoir un esprit **créatif**
Connaissances pour **débuter** et **progresser** sur le marché du travail
Utilisé **Android** et **Flutter**
Chapitres
Chapitre 1 Rappel et prise de main
Chapitre 2 Initiation à Kotlin
Chapitre 3 Données, stockage et architecture
Chapitre 4 Développements hybrides avec Flutter
Chapitre 5 Déploiement d'une application mobile

CANEVAS

Chapitre 1 : Rappel et prise de main

- ✓ Installation d'Android Studio et Flutter
- ✓ Les environnements de travail
- ✓ Vue générale sur UX/UI, d'Adobe XD et les maquettes
- ✓ Lancement d'une application modèle
- ✓ Cycle de vie d'une application

Chapitre 2 : Initiation à Kotlin

- ✓ Variables et types
- ✓ Les collections natives (tableaux, listes, etc.)
- ✓ Structures de contrôle
- ✓ Fonctions et classes
- ✓ AsyncTask
- ✓ Requête réseau et RecyclerView

Chapitre 3 : Données, stockage et architecture

- ✓ Stockage dans un fichier et avec base de données SQLite
- ✓ L'architecture components
- ✓ Structure d'une application

Chapitre 4 : Développements hybrides avec Flutter

- ✓ L'environnement Flutter
- ✓ Initiation au langage DART

Chapitre 5 : Déploiement d'une application mobile

- ✓ Publication du projet sur GitHub
- ✓ Publication d'une application sur le Store

Installation des outils

Installation des outils

- Installation Android Ladybug
- Suivre les étapes d'installation depuis le site officiel de Flutter
- Installation VSCode
- Installation de Flutter et Dart dans VSCode

Installation des outils

User eXperience UX

Ne pas prendre en compte UX = Pertes

- Les programmeurs passent 50 % de leur temps à effectuer des retouches évitables.
- Le coût de correction des erreurs après le développement est 100 fois plus élevé.
- La formation interne pour des applications non intuitives peut augmenter les coûts.

User eXperience UX

ROI = Retour sur investissement

- 68 % des utilisateurs quittent les sites à cause de la qualité du design.
- 85 % des problèmes d'expérience utilisateur peuvent être résolus avec des tests sur seulement 5 personnes.
- Chaque dollar investi dans l'expérience utilisateur rapporte un gain de 100 dollars.

User eXperience UX

Utile
Souhaitable
Conviens aux personnes ayant des besoins spéciaux
Digne de confiance
Trouvable
Utilisable

User eXperience UX

Ce terme renvoie à l'expérience vécue par un utilisateur lorsqu'il interagit avec un "système", bien souvent dans un contexte d'interaction homme-machine

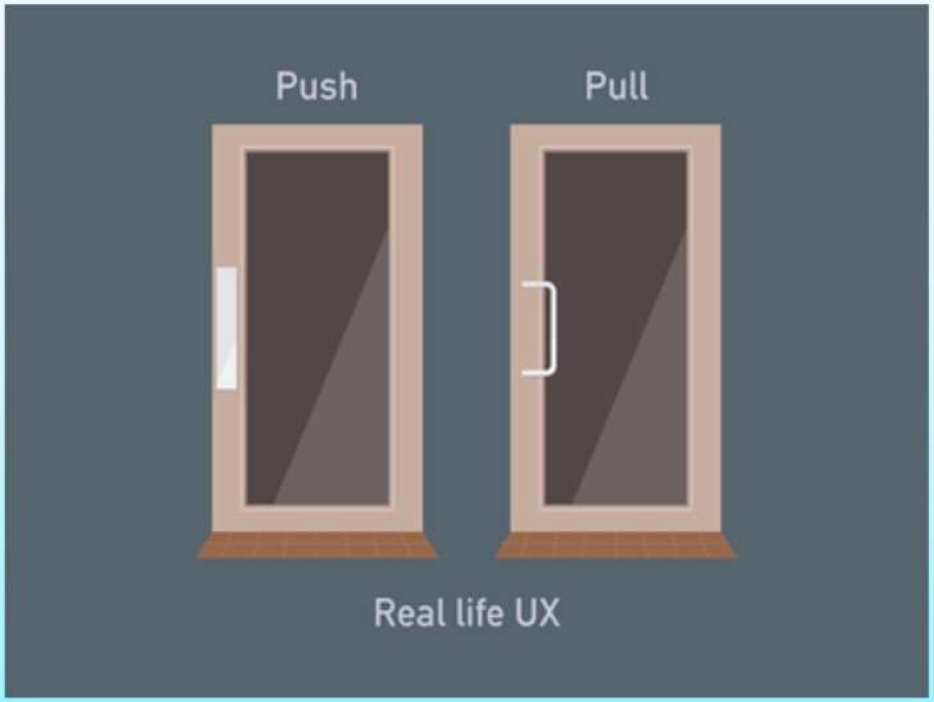
En Marketing Clients, l'Expérience Utilisateur désigne la facilité avec laquelle un individu va utiliser le système

Désigne la qualité de l'expérience vécue par l'utilisateur dans toute situation d'interaction. L'UX qualifie l'expérience globale ressentie par l'utilisateur lors de l'utilisation d'une interface, d'un appareil digital ou plus largement en interaction avec tout dispositif ou service.

User eXperience UX

User eXperience UX

User eXperience UX

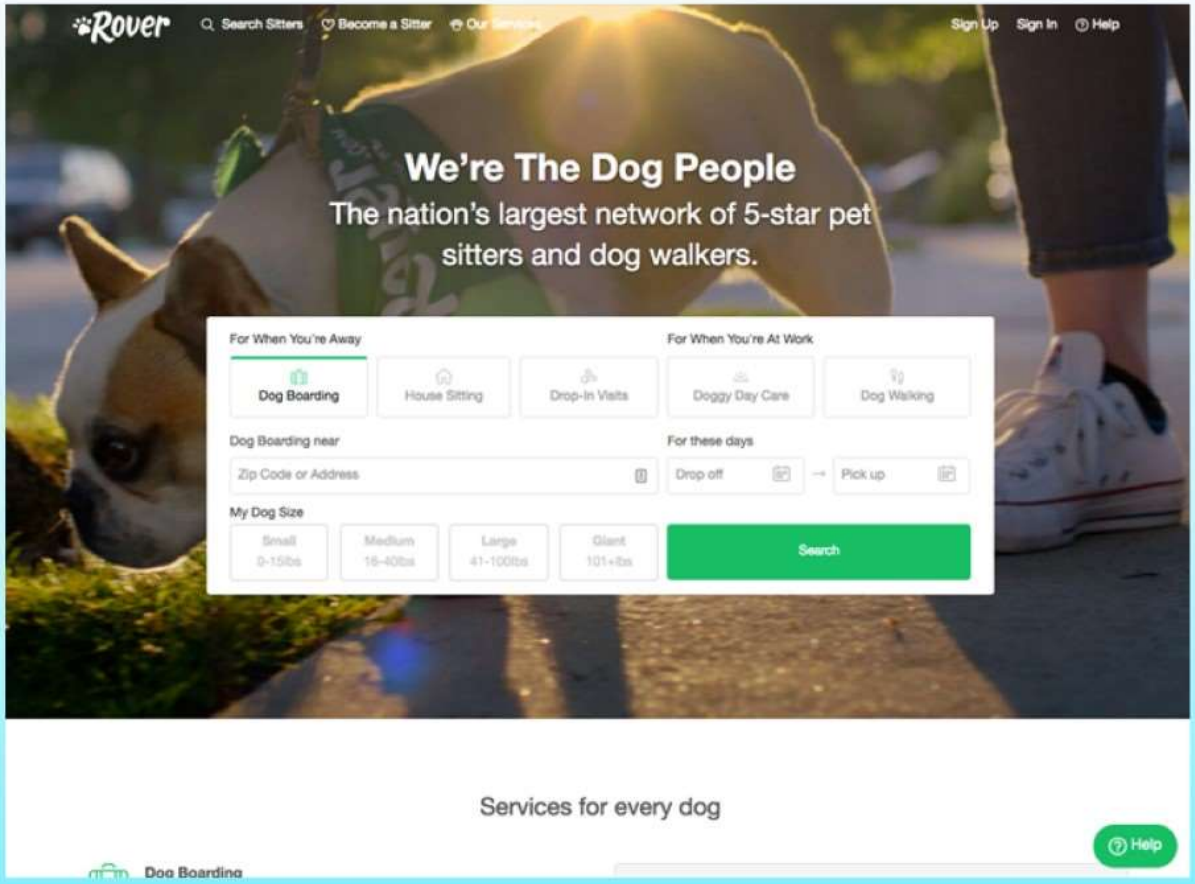


User eXperience UX

User eXperience UX

User eXperience UX

User eXperience UX



User eXperience UX

User eXperience UX

Apprendre à identifier les problèmes d'UX et à proposer des solutions de conception

1- Choisir une application existante

2- Analyse de l'UX actuelle

- ✓ Points positifs
- ✓ Points négatifs
- ✓ Expérience générale

3- Poser des questions sur l'utilisation de l'application, les frustrations rencontrées et les fonctionnalités qu'ils aimeraient voir

User eXperience UX

Exemple

Points positifs

- ✓ Large choix de produits et catégories variées.
- ✓ Offres promotionnelles fréquentes et système de réduction attractif.
- ✓ Interface utilisateur bien organisée, permettant une navigation fluide entre les produits.
- ✓ Suivi des commandes avec des notifications de mise à jour.

User eXperience UX

Exemple

Points négatifs

- ✓ Parfois, les délais de livraison peuvent être plus longs que prévu.
- ✓ Difficulté à contacter le service client ou obtenir des réponses rapides en cas de problème.
- ✓ Options de filtrage et de tri limitées, surtout lorsqu'on cherche des produits spécifiques avec des critères précis.
- ✓ Beaucoup d'information et d'animation.
- ✓ Produits chers par rapport au marché.

User eXperience UX

Exemple

Expérience générale

L'expérience utilisateur est généralement positive, avec des offres intéressantes et une navigation fluide.

Cependant, les utilisateurs peuvent être déçus par les retards de livraison et l'assistance client parfois difficile à joindre.

User eXperience UX

Exemple

Questions posées aux utilisateurs

- ✓ Comment évaluez-vous la fiabilité des délais de livraison ?
- ✓ Avez-vous rencontré des difficultés pour suivre une commande ou contacter le service client ?
- ✓ Quelles améliorations aimeriez-vous voir dans la navigation ou les filtres de recherche ?
- ✓ Avez-vous des suggestions pour rendre l'expérience d'achat plus agréable ?

User eXperience UX

Exemple

Solutions de conception proposées

- ✓ Optimiser les options de filtrage et de tri pour aider les utilisateurs à trouver des produits plus facilement (par exemple : filtres par prix, évaluation, délai de livraison estimé).
- ✓ Améliorer le suivi des commandes avec des mises à jour plus fréquentes et plus précises pour réduire l'incertitude.
- ✓ Renforcer le service client avec une assistance en direct ou un chatbot pour des réponses rapides.
- ✓ Ajouter un système de retour et d'évaluation des produits pour aider les utilisateurs à faire des choix éclairés en se basant sur les avis d'autres acheteurs.

User eXperience UX

Étapes de conception de l'expérience utilisateur

- ✓ **Stratégie** : définir les étapes selon l'entreprise, atteindre le segment cible, établir le budget, etc.
- ✓ **Recherche et exploration** : étudier les utilisateurs, les concurrents et les objectifs du projet.
- ✓ **Analyse** : analyser les informations de la recherche (création du persona, carte de l'expérience utilisateur, etc.) pour définir les idées de base du produit.
- ✓ **Conception** : transformer les idées en croquis.
- ✓ **Production** : programmer le prototype.
- ✓ **Lancement en version bêta**.
- ✓ **Évaluation** pour améliorer le produit.
- ✓ **Lancement final**.

User eXperience UX

Étude de l'utilisateur

Via les entretiens

- ✓ Sélection d'un groupe appartenant à la catégorie cible.
- ✓ Comprendre l'environnement des utilisateurs et leurs besoins.
- ✓ Toutes les questions sont ouvertes, sans réponses par oui ou non.
- ✓ L'utilisateur est celui qui parle le plus.

Via le questionnaire

User eXperience UX

Étude de l'utilisateur

Google Analytics et environnement de l'utilisateur

- ✓ Lors de l'activation de cette fonctionnalité sur notre site, cela nous aide grandement à comprendre le comportement de l'utilisateur.
- ✓ Observer les employés et voir comment ils interagissent avec le système.

User eXperience UX

Étude des objectifs du projet - BMC (Business Model Canvas)

Étude des concurrents sur le marché

User eXperience UX

Analyse, conception et construction des hypothèses

Structuration de l'information et création de la carte du site

Card Sorting, nous préparons les sections du site, puis nous demandons aux utilisateurs de trier ces cartes en fonction de l'endroit où ils s'attendent à les trouver.

Création du persona de l'utilisateur potentiel

Informations de base sur l'utilisateur (âge, statut marital, profession, défis rencontrés, préoccupations, comment le programme répond aux besoins de cet utilisateur, etc.). Au lieu de concevoir pour un inconnu, nous concevons pour une personne spécifique. En général, nous concevons environ trois personas pour couvrir le plus grand nombre de segments d'utilisateurs du produit.

User eXperience UX

Analyse, conception et construction des hypothèses

Création de la carte de l'expérience utilisateur

Cette carte décrit toutes les étapes par lesquelles l'utilisateur passe au cours de son expérience avec le produit (avant, pendant et après l'utilisation).

Analyse des tâches et séquencement des étapes de travail

Nous analysons toutes les étapes nécessaires pour accomplir une tâche donnée.

Création de cas d'utilisation et scénarios

Nous imaginons ce que fait un utilisateur du produit.

User eXperience UX

Analyse, conception et construction des hypothèses

Réalisation de croquis et de maquettes

- ✓ Nous dessinons la conception du produit sur papier (Sketching).
- ✓ Le schéma de base du site, de la page, du produit, etc., est appelé Wireframing.
- ✓ L'avantage est que nous ne faisons pas de changements majeurs après la construction du produit.
- ✓ Nous faisons tester le produit par des utilisateurs et recueillons leurs avis avant de construire le produit final.

User eXperience UX

Outils

Adobe XD

- Conception d'Interfaces
- Prototypage Interactif
- Collaboration et Partage
- Intégration avec d'autres outils Adobe

Figma

User eXperience UX

Analyse, conception et construction des hypothèses

Création et conception de l'interface utilisateur

- ✓ Il s'agit de la dernière étape.
- ✓ C'est la conception de l'interface utilisateur (UI).
- ✓ Nous appliquons l'identité visuelle sur le produit.
- ✓ Nous construisons l'interface utilisateur.
- ✓ Il est essentiel d'apprendre certaines bases à ce sujet.

User eXperience UX

Test de l'utilisabilité

Préparation et sélection des échantillons de recherche

- ✓ Il s'agit d'une expérience pour vérifier si les utilisateurs font ce que nous attendions ou non.
- ✓ On teste une personne à la fois, en utilisant une version préliminaire du produit.
- ✓ Il y a trois rôles : l'utilisateur, le modérateur et l'observateur.
- ✓ 3 à 5 personnes
- ✓ Autres

User eXperience UX

Test de l'utilisabilité

Comment réaliser le test

- ✓ Nous rédigeons un script de test afin d'uniformiser l'expérience pour tous les utilisateurs.
- ✓ Nous préparons l'environnement de test.
- ✓ Le test se déroule dans un laboratoire d'utilisabilité.
- ✓ Il est préférable d'enregistrer tout, en audio et en vidéo (avec l'accord des participants)..

User eXperience UX

Test de l'utilisabilité

Analyse des résultats

L'analyse doit être scientifique, basée sur des statistiques et des théories.

Reskin

Le reskin d'une application Android consiste à modifier l'apparence visuelle ou l'interface utilisateur (UI) d'une application existante sans changer ses fonctionnalités de base. C'est une méthode couramment utilisée pour créer de nouvelles versions d'une application ou adapter une application existante à une nouvelle marque ou à un public spécifique.

Reskin

Pourquoi faire un reskin d'une application ?

Réduire le temps de développement

En utilisant une base existante, on évite de repartir de zéro.

Adapter à une nouvelle marque

Changer les couleurs, les logos et les styles pour refléter une nouvelle identité visuelle.

Cibler différents marchés

Créer plusieurs versions d'une application adaptée aux préférences culturelles, linguistiques ou régionales.

Augmenter les revenus

Proposer une nouvelle version d'une application populaire, souvent dans le cadre de modèles commerciaux comme la monétisation par publicité ou achats intégrés.

Reskin

Étapes pour faire un reskin d'une application Android

- ✓ Acheter une licence pour un code source prêt à l'emploi (par exemple, sur des plateformes comme CodeCanyon).
- ✓ Définir une charte graphique : couleurs, polices, icônes, et images.
- ✓ Créer un wireframe ou une maquette du nouvel aspect de l'application.
- ✓ Modifier les ressources graphiques.
- ✓ Tester l'application relookée
- ✓ Mettre à jour les détails sur le Google Play Store (nouveau nom package, nouveau nom, description, captures d'écran, etc.).

Reskin

Exemple

Créer plusieurs versions d'un même jeu en modifiant les personnages, les arrière-plans ou les thèmes.

Reskin

Avantages et inconvénients du reskin

Avantages

- ✓ Gain de temps et réduction des coûts de développement.
- ✓ Permets de tester de nouvelles idées ou audiences rapidement.
- ✓ Facilité de personnalisation pour divers marchés ou marques.

Inconvénients

- ✓ Le succès peut être limité si seules les apparences sont modifiées sans innovation fonctionnelle.
- ✓ Nécessité de respecter les licences des codes sources achetés.
- ✓ La concurrence peut avoir des versions similaires.

Reskin

Plateformes de codes sources

CodeCanyon codecanyon.net

SellMyApp sellmyapp.com

Cycle de vie

Phases principales du cycle de vie

Le cycle de vie est géré par une série de méthodes de rappel (callbacks) définies dans la classe Activity ou Fragment. Voici les étapes pour une Activity :

Cycle de vie

Phases principales du cycle de vie

1. Création de l'activité onCreate()

- Première méthode appelée lors de la création de l'activité.

Utilisée pour :

- Initialiser les ressources.
- Définir l'interface utilisateur avec setContentView().
- Récupérer les données sauvegardées si l'activité est recréée.

Cycle de vie

Phases principales du cycle de vie

Démarrage de l'activité onStart()

- L'activité devient visible, mais n'est pas encore en interaction avec l'utilisateur.
- Prépare les ressources visibles ou effectue des opérations légères

Cycle de vie

Phases principales du cycle de vie

Reprise de l'activité onResume()

- L'activité est maintenant au premier plan et peut interagir avec l'utilisateur.
- Idéal pour reprendre des animations, flux de données ou interactions.

Cycle de vie

Phases principales du cycle de vie

Pause de l'activité onPause()

- L'activité perd le focus (par exemple, si une autre activité est partiellement visible).
- Utilisée pour suspendre les opérations gourmandes en ressources ou sauvegarder les données.

Cycle de vie

Phases principales du cycle de vie

Arrêt de l'activité onStop()

- L'activité n'est plus visible (remplacée par une autre ou minimisée).
- Sert à libérer les ressources non essentielles, telles que les connexions réseau ou les threads.

Cycle de vie

Phases principales du cycle de vie

Destruction de l'activité onDestroy()

- Appelée lorsque l'activité est définitivement terminée (par exemple, l'utilisateur l'a fermée ou le système doit libérer de la mémoire).
- Nettoyage des ressources ou fermeture des bases de données.

Cycle de vie

Exemple Jumia

1. Ouverture de l'application

Méthode appelée : onCreate()

- L'application charge la page d'accueil avec les offres.
- Initialise les bases de données et les API.

onStart() et onResume()

- L'interface devient visible et interactive (l'utilisateur peut naviguer entre les catégories).

Cycle de vie

Exemple Jumia

2 L'utilisateur consulte un produit

- L'application reste dans l'état actif.

Si l'utilisateur clique sur un produit :

- Une nouvelle activité s'ouvre (détails du produit).
- Méthodes appelées pour la nouvelle activité : onCreate(), onStart(), onResume().

Cycle de vie

Exemple Jumia

L'utilisateur passe à une autre application

Méthode appelée : onPause() pour la page de détails.

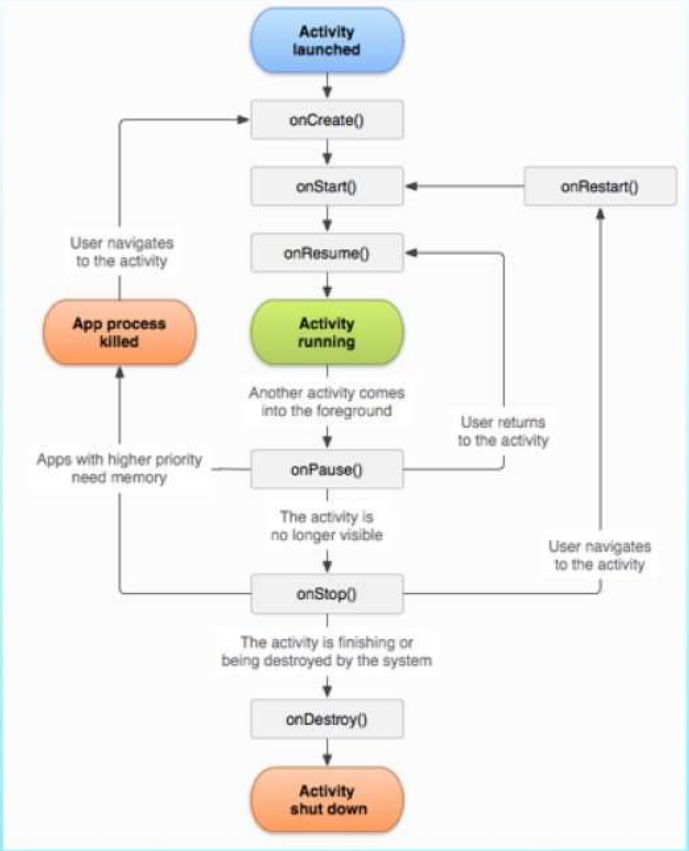
- L'application suspend les animations ou mises à jour.

Si l'utilisateur reste longtemps inactif, onStop() est appelé.

- Les ressources non critiques (par exemple, les requêtes réseau) sont libérées.

Cycle de vie

Shéma



Cycle de vie

Exemple de deux applications

Lancement de YouTube
et
lecteur d'une vidéos

L'utilisateur ouvre l'application YouTube

- L'activité principale (liste des vidéos) est **créée** : onCreate(), onStart(), et onResume() sont appelées.

L'utilisateur clique sur une vidéo pour la regarder

- Une **nouvelle activité** (lecteur vidéo) est **créée** et devient active.
- La méthode onCreate() est appelée pour le lecteur vidéo. La liste des vidéos passe en onPause().

Cycle de vie

Exemple de deux applications

Notification d'appel Messenger

Pendant que la vidéo est en cours, l'utilisateur reçoit un appel Messenger.

- Une **nouvelle activité** pour l'écran d'appel est **créée** (via Messenger).
- Les méthodes onCreate(), onStart(), et onResume() de l'activité d'appel sont appelées.

L'activité du lecteur vidéo de YouTube entre dans l'état onPause(), puis éventuellement onStop() (car elle n'est plus visible).

Cycle de vie

Exemple de deux applications

Pendant l'appel Messenger

L'utilisateur répond à l'appel et discute via Messenger

Pendant ce temps, l'activité de lecture vidéo reste en arrière-plan, mais la vidéo est mise en pause automatiquement ou manuellement.

Cycle de vie

Exemple de deux applications

Fin de l'appel et retour à YouTube

Lorsque l'utilisateur termine l'appel, il revient à YouTube.

- L'activité du lecteur vidéo passe de onStop() à onRestart(), puis à onStart() et onResume().

La vidéo reprend à l'endroit où elle avait été interrompue.

Cycle de vie

Exemple de deux applications

Fin de l'appel et retour à YouTube

Lorsque l'utilisateur termine l'appel, il revient à YouTube.

- L'activité du lecteur vidéo passe de onStop() à onRestart(), puis à onStart() et onResume().

La vidéo reprend à l'endroit où elle avait été interrompue.

Kotlin

Test 1

En Kotlin, on distingue les variables dont on **peut changer la valeur**, et les variables **immuables** qui ne sont pas tout à fait des constantes

- le mot-clé **val** permet de déclarer une variable immuable,
- le mot-clé **var** permet de déclarer une variable altérable.

Kotlin

Test 2

Bien que vous puissiez techniquement déclarer des variables avec des types **immuables (val)** ou **altérables (var)** pour des listes, cela peut prêter à confusion, car cela **ne change pas la nature immuable ou mutable de la liste**.

val

- Si vous déclarez une liste avec val, cela signifie que la référence à cette liste ne peut pas être modifiée. Vous ne pouvez pas réassigner une nouvelle liste à cette variable.
- Cependant, si la liste est mutable (MutableList), vous pouvez toujours modifier son contenu.

var

- Si vous déclarez une liste avec var, cela signifie que vous pouvez réassigner la variable à une nouvelle liste.
- Cependant, la nature mutable ou immuable de la liste dépend toujours du type de liste que vous utilisez (List vs MutableList).

Kotlin

Test 3

En Kotlin, tout est objet. De surcroît Kotlin dispose d'un mécanisme, permettant **d'ajouter** des méthodes aux classes existantes

Kotlin

Test 4

Le type Boolean

Kotlin

Test 5

Le type Map représente une collection de paires **clé-valeur**
Chaque clé unique est associée à une valeur correspondante

- Map : Une carte **immuable**, où les entrées ne peuvent pas être modifiées après la création.
- MutableMap : Une carte **mutable**, où vous pouvez ajouter, supprimer ou modifier des entrées après la création.
- Chaque entrée dans la carte est une paire **clé-valeur**. Les **clés sont uniques**, mais les valeurs peuvent être dupliquées.
- Vous pouvez **accéder** à une valeur en utilisant sa **clé**.
- Ajout et modification d'entrées
- Suppression d'entrées

Kotlin

Test 6

Le type String dispose des mêmes méthodes qu'en Java, mais aussi des méthodes supplémentaires

- La méthode **all**, permettant de tester si tous les éléments d'une collection remplissent un critère

Des méthodes utilisées avec elle:

- Vérifier si tous les caractères sont des lettres : **isLetter**
- Vérifier si tous les caractères sont des chiffres : **isDigit**
- Retourne true si le caractère est en minuscule : **isLowerCase()**
- Retourne true si le caractère est en majuscule : **isUpperCase**

Kotlin

Test 7

La fonction **any** est utilisée pour vérifier si au moins un élément d'une collection satisfait une condition donnée. Donc, parcourt chaque élément d'une collection et retourne **true** si au moins un élément remplit une certaine condition, sinon il retourne **false**.

Kotlin

Test 8

le type **Personne** n'est pas un type prédéfini dans la bibliothèque standard.
Cependant, vous pouvez facilement créer un tel type en définissant une classe personnalisée.

Une classe Personne pourrait être utilisée pour représenter une personne avec des propriétés comme le **nom**, **l'âge**, **l'adresse**, etc.

Vous pouvez également ajouter des méthodes à la classe Personne pour effectuer des actions spécifiques, comme afficher les informations de la personne

Kotlin

Test 9

Il est également possible d'utiliser des data classes pour des classes comme Personne qui sont principalement utilisées pour contenir des données

Les data classes génèrent automatiquement des méthodes comme **toString()**, **equals()**, **hashCode()**, et **copy()**.

Kotlin

Test 10

Les tableaux sont représentés par la classe générique Array : ainsi l'on peut déclarer par exemple, un **Array<Int>**, un **Array<String>**, voire même un **Array<Personne>**.
Évidemment, un Array ne peut contenir que des valeurs d'un **même type commun**.

On peut construire un Array de deux manières différentes :
constructeur Array
fonction arrayOf

Les tableaux Kotlin sont des objets de type Array<T>, qui ne redéfinissent pas la méthode **toString()** pour afficher les éléments de manière lisible

Stockage

SharedPreferences

API Android utilisée pour stocker des données simples sous forme de paires clé-valeur (comme des chaînes de caractères, des entiers, des booléens, etc.).
Elle permet de sauvegarder des informations persistantes de manière légère et rapide, comme les paramètres utilisateur ou les préférences de l'application.
On appelle ce type de stockage le stockage par Clé/Valeur. Les variables (et leurs clés correspondantes) seront stockées automatiquement dans un fichier XML directement sur le téléphone de l'utilisateur. Ce fichier est défini en mode PRIVATE pour n'autoriser l'accès à son contenu qu'à notre application.

Stockage

Données structurées

Android vous offre nativement la possibilité d'utiliser une base de données de type SQLite.
Ce type de stockage vous permettra d'organiser et faire persister de grosses quantités de données structurées, directement sur le téléphone de l'utilisateur, et manipulables grâce au langage SQL.

Stockage

Stockage interne et stockage externe

Afin de supporter le stockage de données sous forme de fichiers (photos, vidéos, PDF, etc.), les périphériques Android disposent de deux espaces de stockage :

- L'espace de stockage dit « interne »
- L'espace de stockage dit « externe »

Ces noms assez communs proviennent de l'époque où la plupart des périphériques Android possédaient à la fois une ***mémoire interne non volatile physique*** et une **mémoire amovible** telle qu'une carte SD par exemple.

Aujourd'hui, la mémoire amovible tend à disparaître et énormément de périphériques divisent dorénavant leur espace de stockage physique en partitions séparées : une partition pour la mémoire interne et une autre pour la mémoire externe.

Stockage

Stockage interne et stockage externe

	STOCKAGE INTERNE	STOCKAGE EXTERNE
Disponibilité	Les fichiers stockés sur cet espace seront **toujours** disponibles.	Les fichiers stockés sur cet espace ne seront **PAS TOUJOURS** disponibles. En effet, ce stockage peut être retiré à tout moment par l'utilisateur (carte SD ou clé USB).
Accessibilité	Les fichiers stockés sur cet espace ne seront accessibles **QUE** par votre application.	Les fichiers stockés sur cet espace seront accessibles par **TOUT LE MONDE**. Vous n'avez donc aucun contrôle dessus.
Durée de vie	Quand l'utilisateur désinstalle votre application, les fichiers seront **automatiquement supprimés** de cet espace de stockage.	Quand l'utilisateur désinstalle votre application, les fichiers ne seront **PAS** supprimés de cet espace de stockage (sauf si vous l'avez explicitement défini).