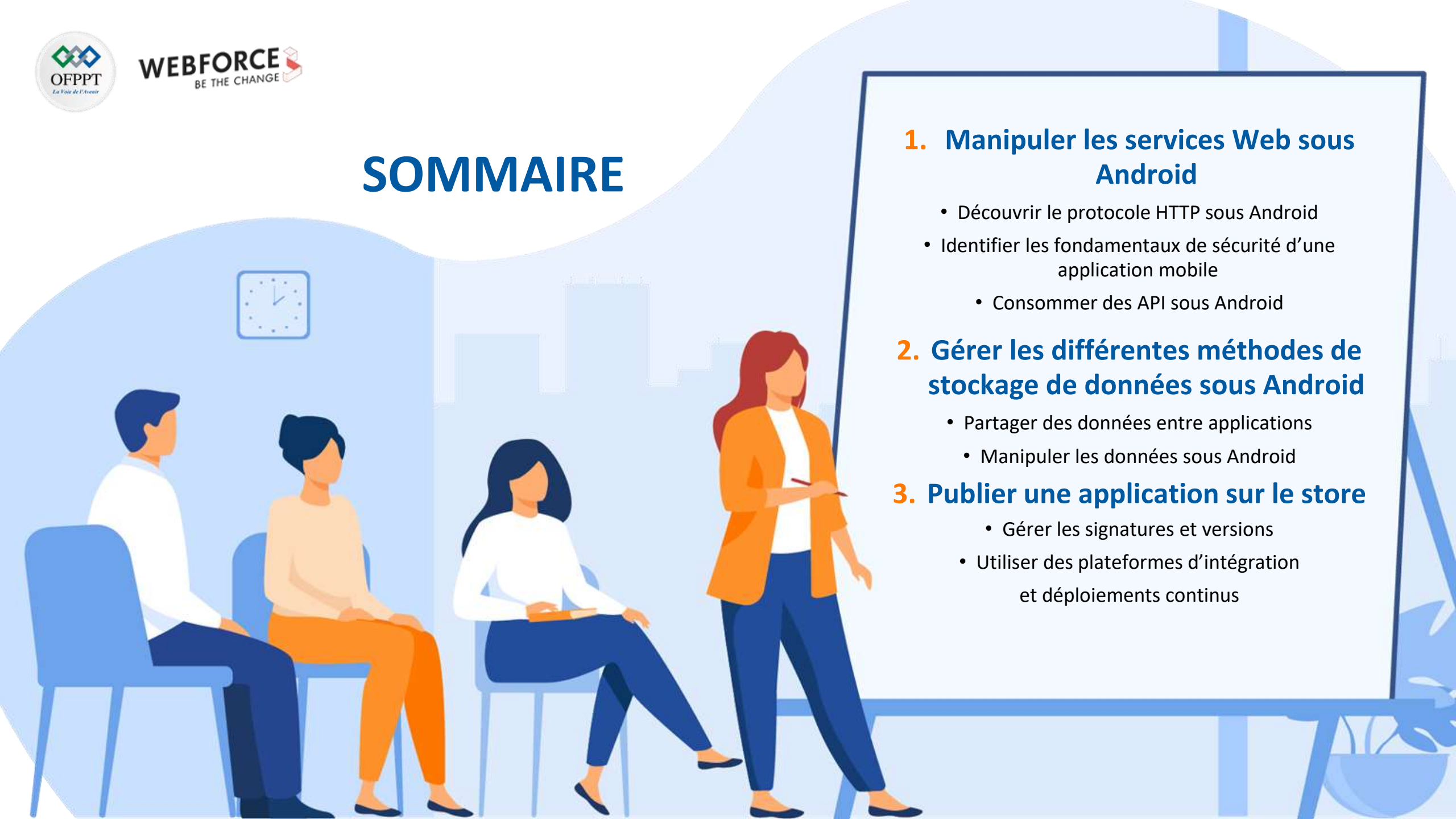


RÉSUMÉ THÉORIQUE – FILIÈRE DÉVELOPPEMENT DIGITAL
Option – Application mobiles
M206 - Élaborer une application Android sécurisée

40 heures

SOMMAIRE



1. **Manipuler les services Web sous Android**
   - Découvrir le protocole HTTP sous Android
   - Identifier les fondamentaux de sécurité d'une application mobile
   - Consommer des API sous Android
2. **Gérer les différentes méthodes de stockage de données sous Android**
   - Partager des données entre applications
   - Manipuler les données sous Android
3. **Publier une application sur le store**
   - Gérer les signatures et versions
   - Utiliser des plateformes d'intégration et déploiements continus

MODALITÉS PÉDAGOGIQUES



1

LE GUIDE DE SOUTIEN

Il contient le résumé théorique et le manuel des travaux pratiques



2

LA VERSION PDF

Une version PDF est mise en ligne sur l'espace apprenant et formateur de la plateforme WebForce Life



3

DES CONTENUS TÉLÉCHARGEABLES

Les fiches de résumés ou des exercices sont téléchargeables sur WebForce Life



4

DU CONTENU INTERACTIF

Vous disposez de contenus interactifs sous forme d'exercices et de cours à utiliser sur WebForce Life

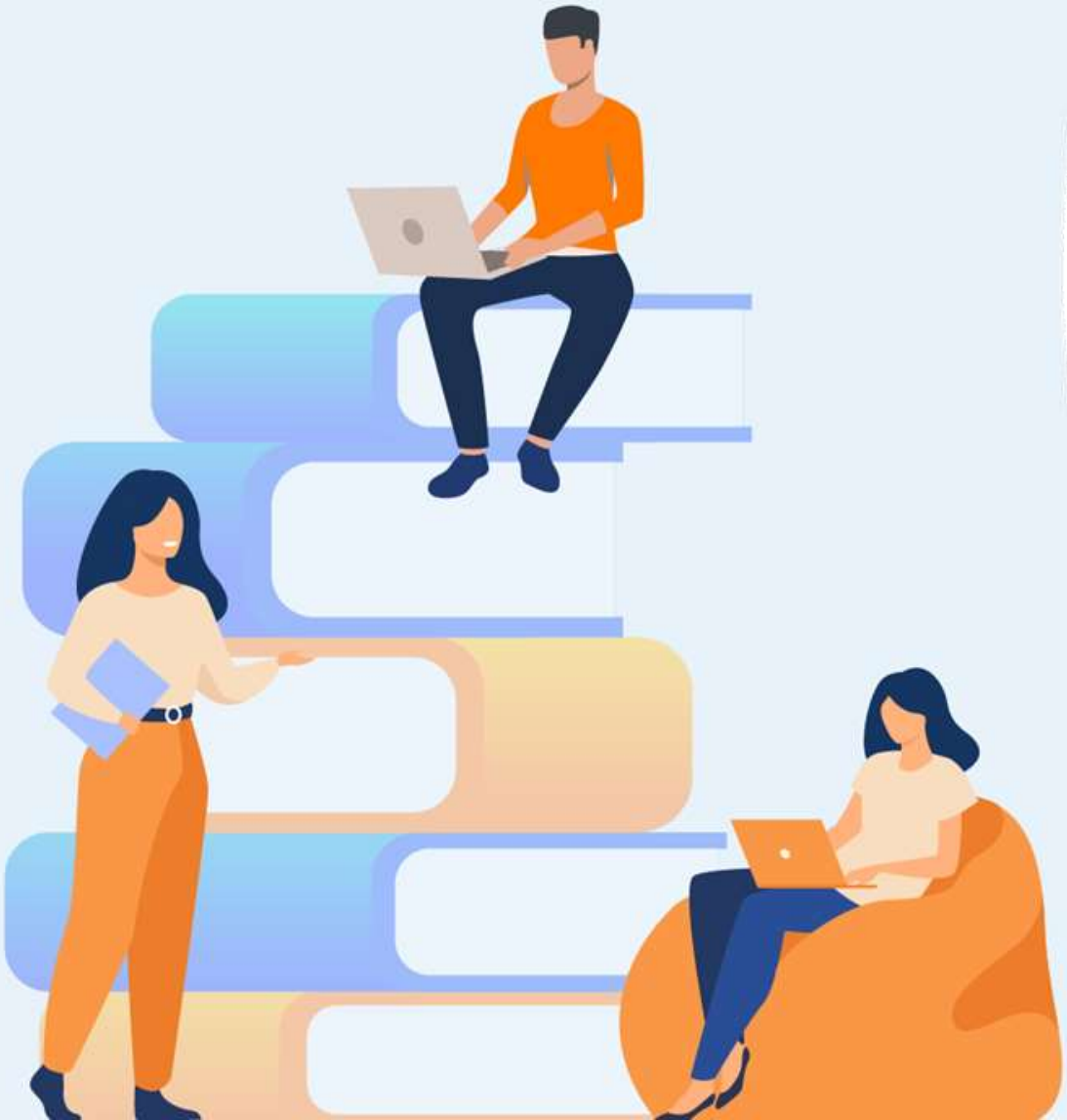


5

DES RESSOURCES EN LIGNES

Les ressources sont consultables en synchrone et en asynchrone pour s'adapter au rythme de l'apprentissage

Copyright - Tout droit réservé - OFPPT



PARTIE 1

Manipuler les services Web sous Android

Dans ce module, vous allez :

- Découvrir le protocole HTTP sous Android
- Maîtriser les fondamentaux de sécurité d'une application mobile
- Consommation des API sous Android
- Sécuriser des échanges de données entre l'application et le serveur

16 heures



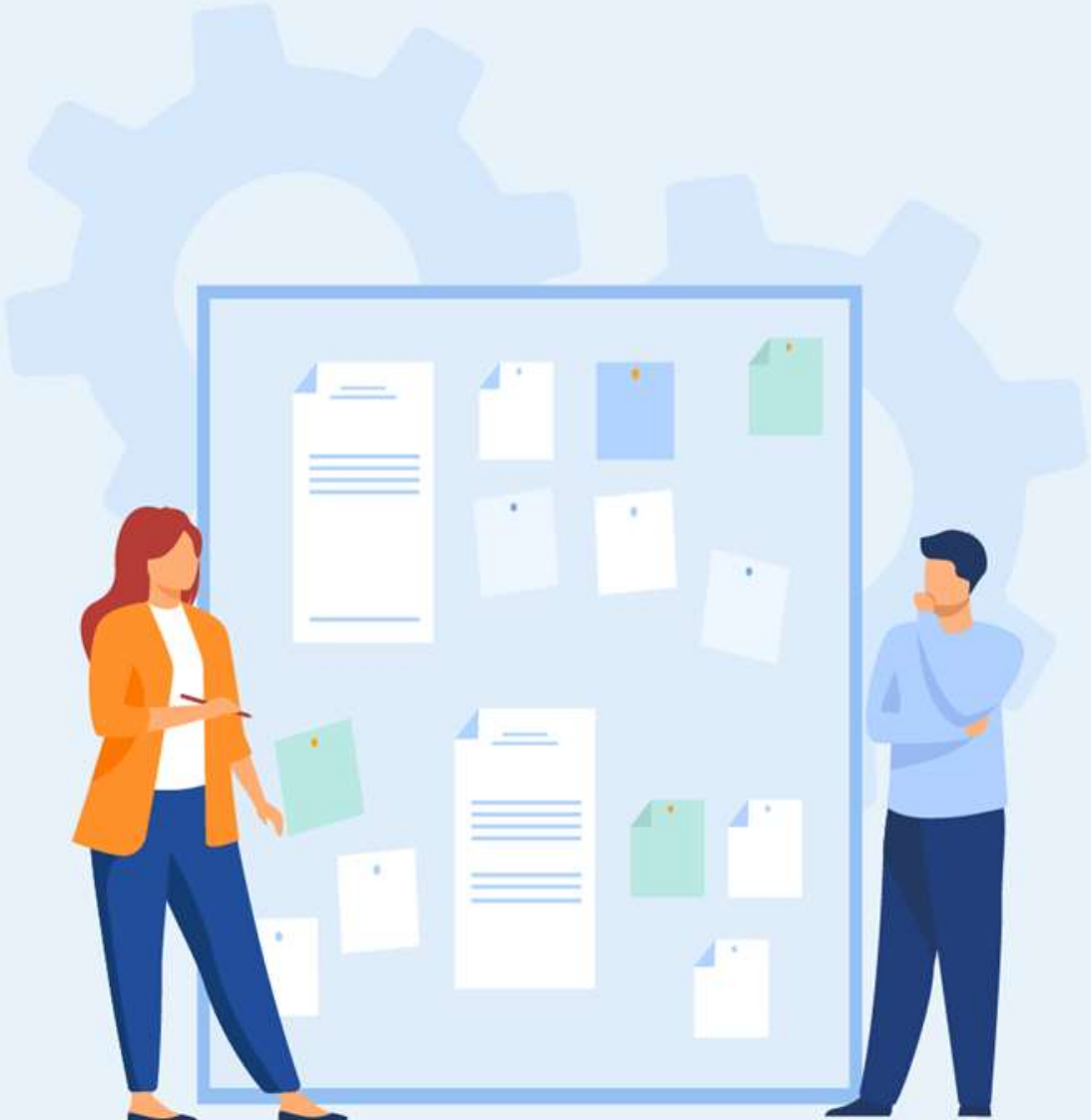
CHAPITRE 1

Découvrir le protocole HTTP sous Android

Ce que vous allez apprendre dans ce chapitre :

- Introduction aux Web services
- Présentation de protocole REST
- Découverte du protocole SOAP

04 heures



CHAPITRE 1

Découvrir le protocole HTTP sous Android

1. Introduction aux Web services

2. Présentation de protocole REST

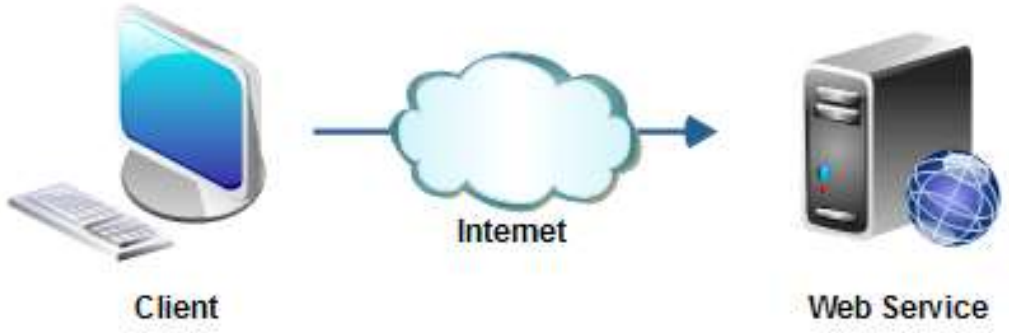
3. Découverte du protocole SOAP

Web service

Un web service est un programme s'exécutant sur un serveur accessible depuis Internet et fournissant un service.

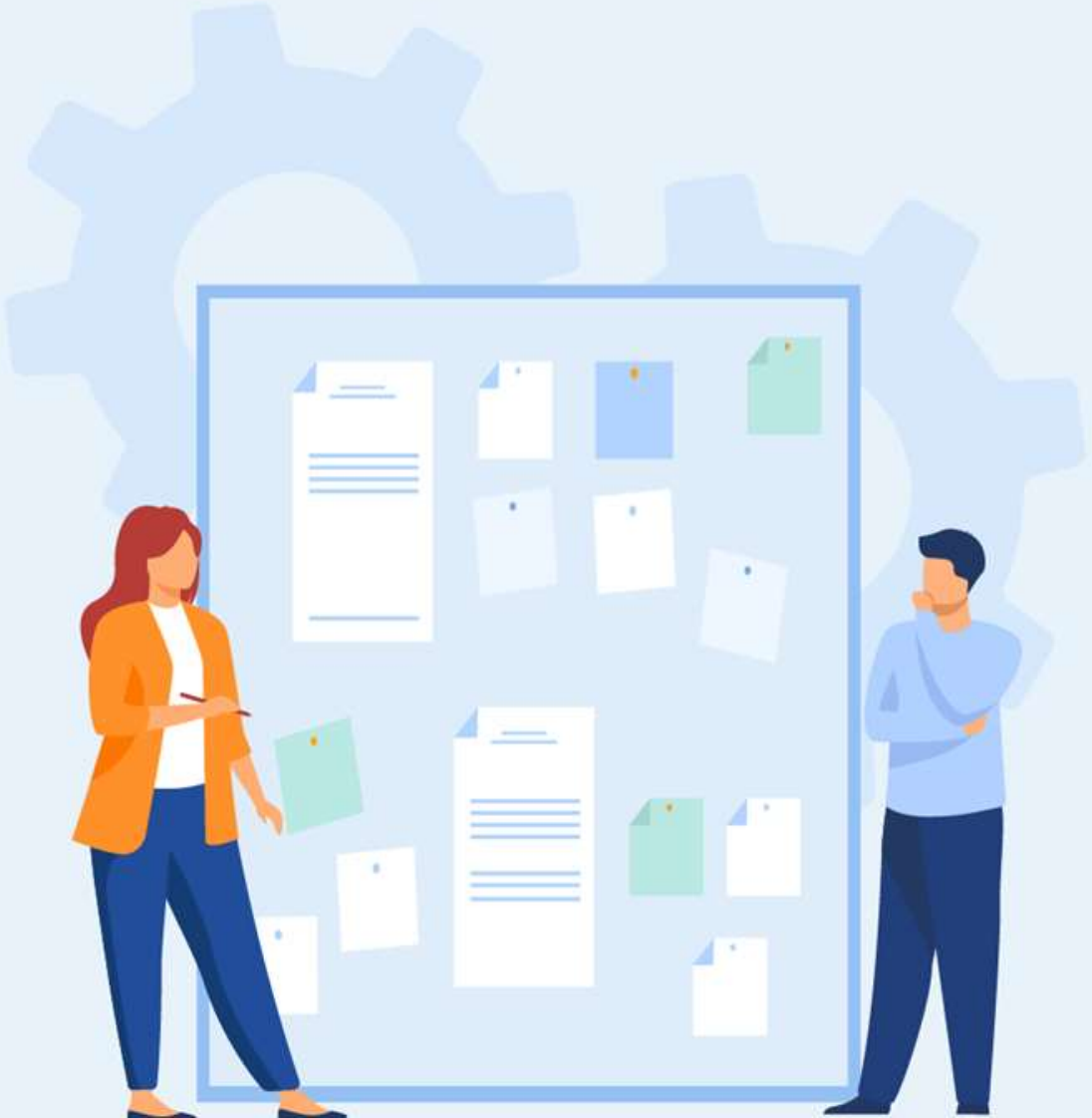
Par exemple :

- Google est un service web qui vous permet de rechercher des sites web.
- Une application météo communique avec un service web qui fournit la météo.
- Un réseau social est un service web qui permet de retrouver ses amis et de communiquer avec.



source : https://jenkov.com/

Copyright - Tout droit réservé - OFPPT



CHAPITRE 1

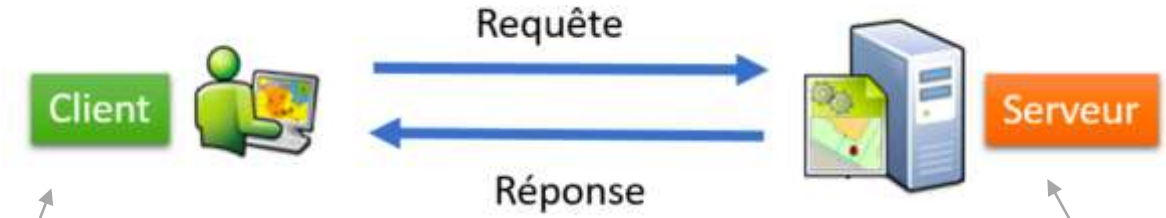
Découvrir le protocole HTTP sous Android

1. Introduction aux Web services
2. **Présentation de protocole REST**
3. Découverte du protocole SOAP

protocole REST

REST est un type d'architecture d'API qui signifie ***RE****presentational* ***S****tate* ***T****ransfer*. Il a été inventé par l'Américain **Roy Fielding** dans les années 2000, afin de mettre en place des méthodes simples pour accéder à des services web.

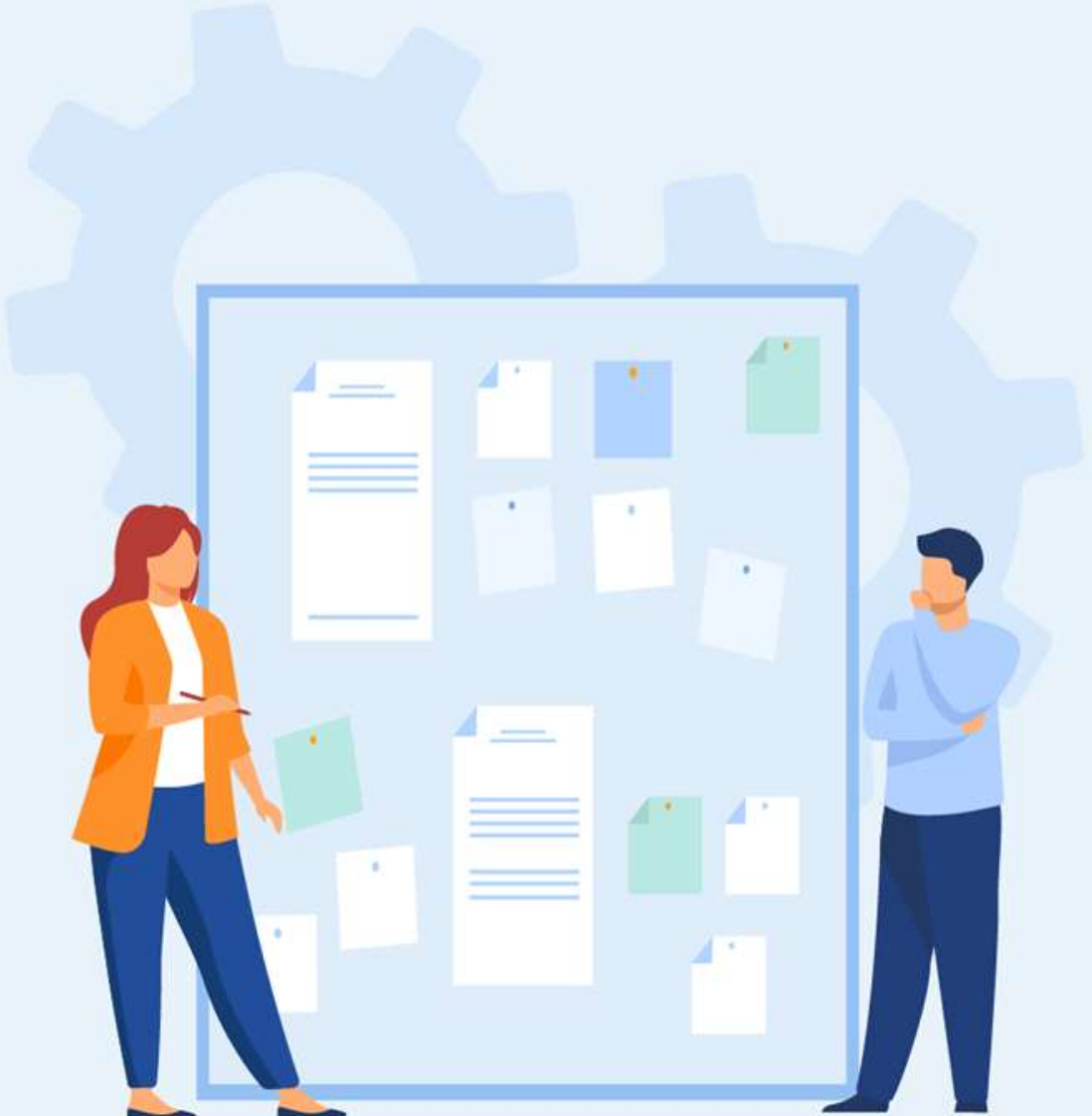
Ce type d'API permet à des logiciels qui sont incompatibles, qui ne parlent pas le même langage, de communiquer facilement. REST peut être considéré comme un langage commun à ces différents logiciels.



source : https://openclassrooms.com

- Un **client** est un programme qui échange les données avec un serveur à travers l'API. C'est l'utilisateur de l'API. On parle plus couramment d'architecture client-serveur.
- Un **serveur** est un ordinateur puissant permettant d'exécuter une API. C'est le fournisseur du service, celui qui propose au client d'accéder à ses données ou services ;

Copyright - Tout droit réservé - OFPPT



CHAPITRE 1

Découvrir le protocole HTTP sous Android

1. Introduction aux Web services
2. Présentation de protocole REST
3. **Découverte du protocole SOAP**

Protocole SOAP

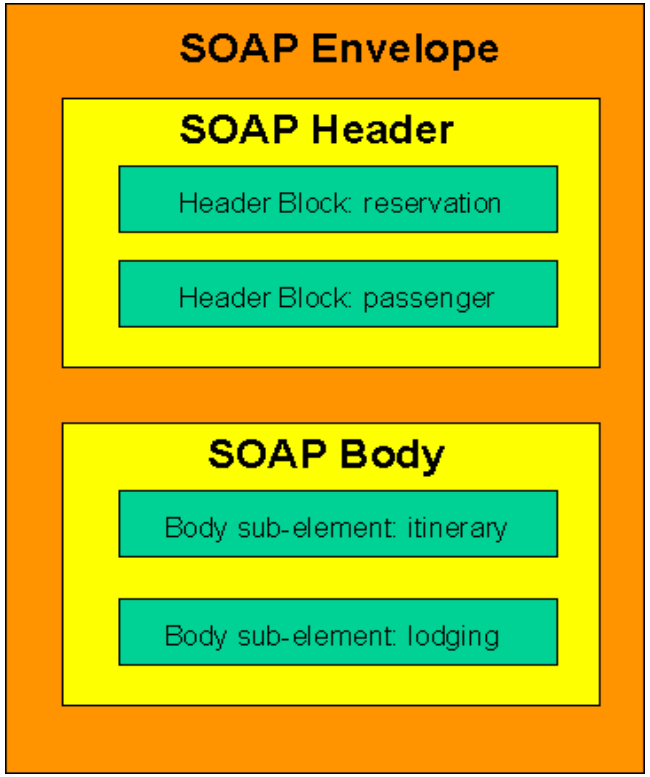
SOAP est un **protocole de communication** basé exclusivement sur **XML** pour permettre aux applications de s'échanger des informations via HTTP. C'est l'acronyme de "**S**imple **O**bject **A**ccess **P**rotocol".

Ce protocole permet ainsi l'accès aux services web et à l'interopérabilité des

applications à travers le web.

Par rapport à REST, SOAP est **plus lent** et requiert **plus de débit Internet**,

car les messages sont plus verbeux que ceux de REST.



source : https://www.w3.org/

Remarques

- XML : Extensible Markup Language vous permet de définir et de stocker des données de manière à pouvoir les partager. il prend en charge l'échange d'informations entre des systèmes informatiques tels que les sites web, les bases de données et les applications tierces.

PARTIE 1

Copyright - Tout droit réservé - OFPPT



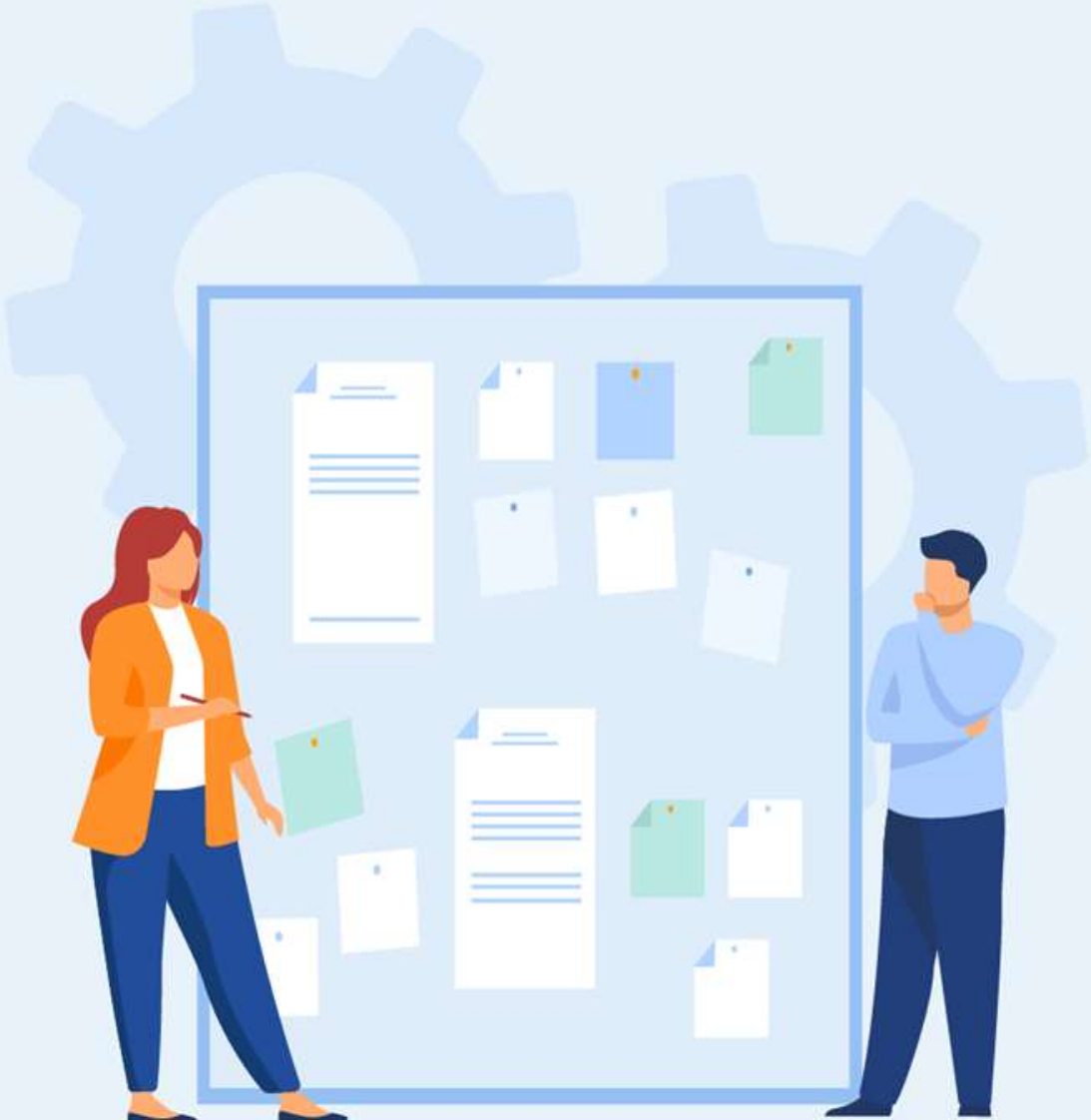
CHAPITRE 2

Identifier les fondamentaux de sécurité d'une application mobile

Ce que vous allez apprendre dans ce chapitre :

- Utilisation de OAUTH
- Intégration de SSL
- Sécurisation avec TLS
- Protection avec Proguard

06 heures



CHAPITRE 2

Identifier les fondamentaux de sécurité d’une application mobile

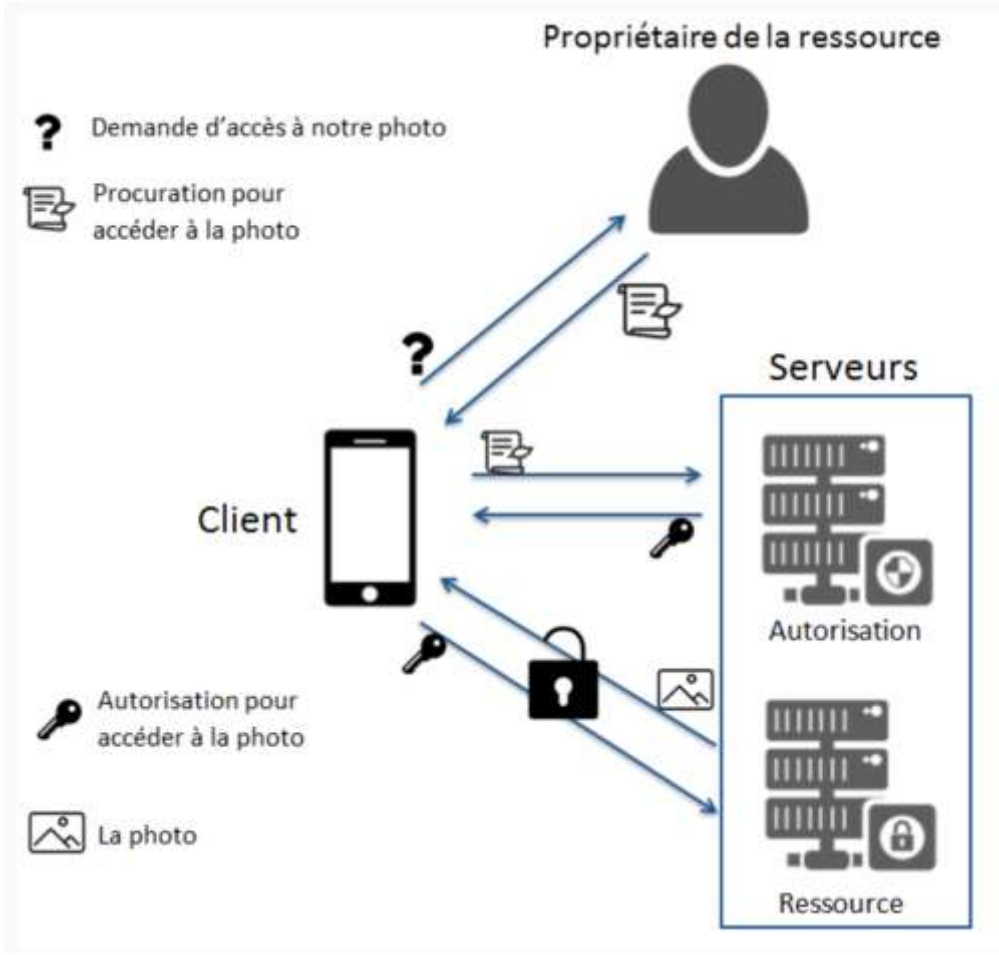
1. Utilisation de OAUTH
2. Intégration de SSL
3. Sécurisation avec TLS
4. Protection avec Proguard

Utilisation de OAUTH

OAuth (Open Authorization) est un protocole qui permet à un utilisateur d'autoriser une application tierce à accéder à ses données sans partager le mot de passe de son compte.

OAuth est utilisé dans le domaine du web avec notamment Facebook ou encore Google, OAuth est devenu incontournable.

Pour les développeurs, ils doivent utiliser un serveur fournissant un accès via OAuth ou implémenter un serveur d'autorisation pour sécuriser une API en utilisant ce protocole.



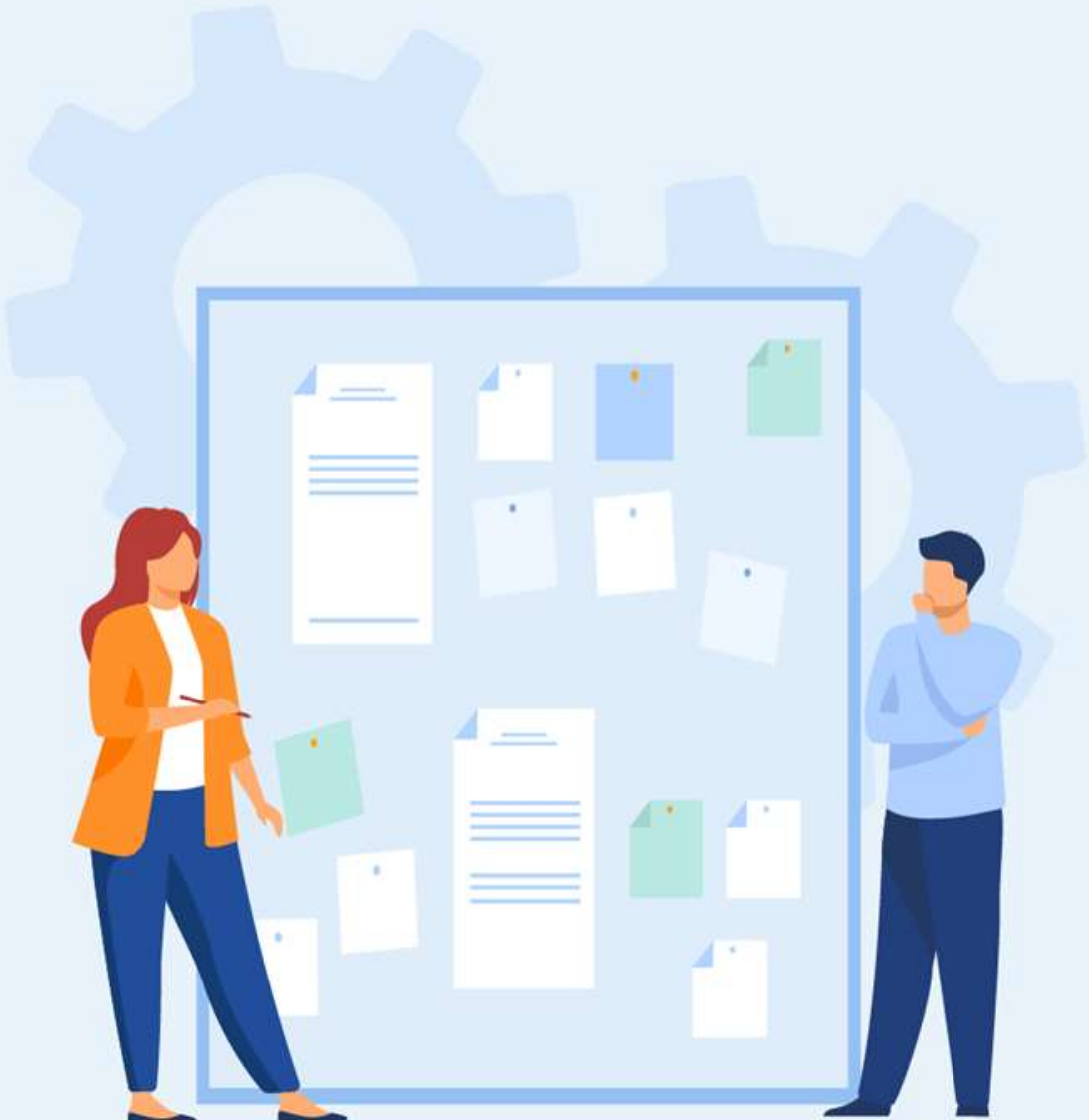
source : https://zestedesavoir.com/

Remarques

- OAuth 2.0 est une norme pour accéder aux ressources hébergées par d'autres applications Web au nom d'un utilisateur. Elle a remplacé OAuth 1.0 en 2012 et constitue désormais la norme industrielle de facto pour l'autorisation en ligne

PARTIE 1

Copyright - Tout droit réservé - OFPPT



CHAPITRE 2

Identifier les fondamentaux de sécurité d’une application mobile

1. Utilisation de OAUTH
2. **Intégration de SSL**
3. Sécurisation avec TLS
4. Protection avec Proguard

Certificat SSL

Un **certificat SSL** est un fichier de données qui lie une clé cryptographique aux informations d'une organisation.

Installé sur un serveur, le certificat active le cadenas et le protocole « **https** », afin d'assurer une connexion sécurisée entre le serveur web et le navigateur.



source : https://ebrandservices.fr/

Le **SSL** est généralement utilisé pour :

- Sécuriser les transactions bancaires
- Sécuriser la navigation sur les sites de réseaux sociaux.

Ce protocole prend en charge les principes de sécurité suivants :

- **Chiffrement** : protège les transmissions de données.
- **Authentification** : garantit que le serveur auquel vous êtes connecté est le bon serveur.
- **Intégrité des données** : garantit que les données qui sont demandées ou soumises sont bien celles qui sont fournies.

Copyright - Tout droit réservé - OFPPT

Intégration de SSL

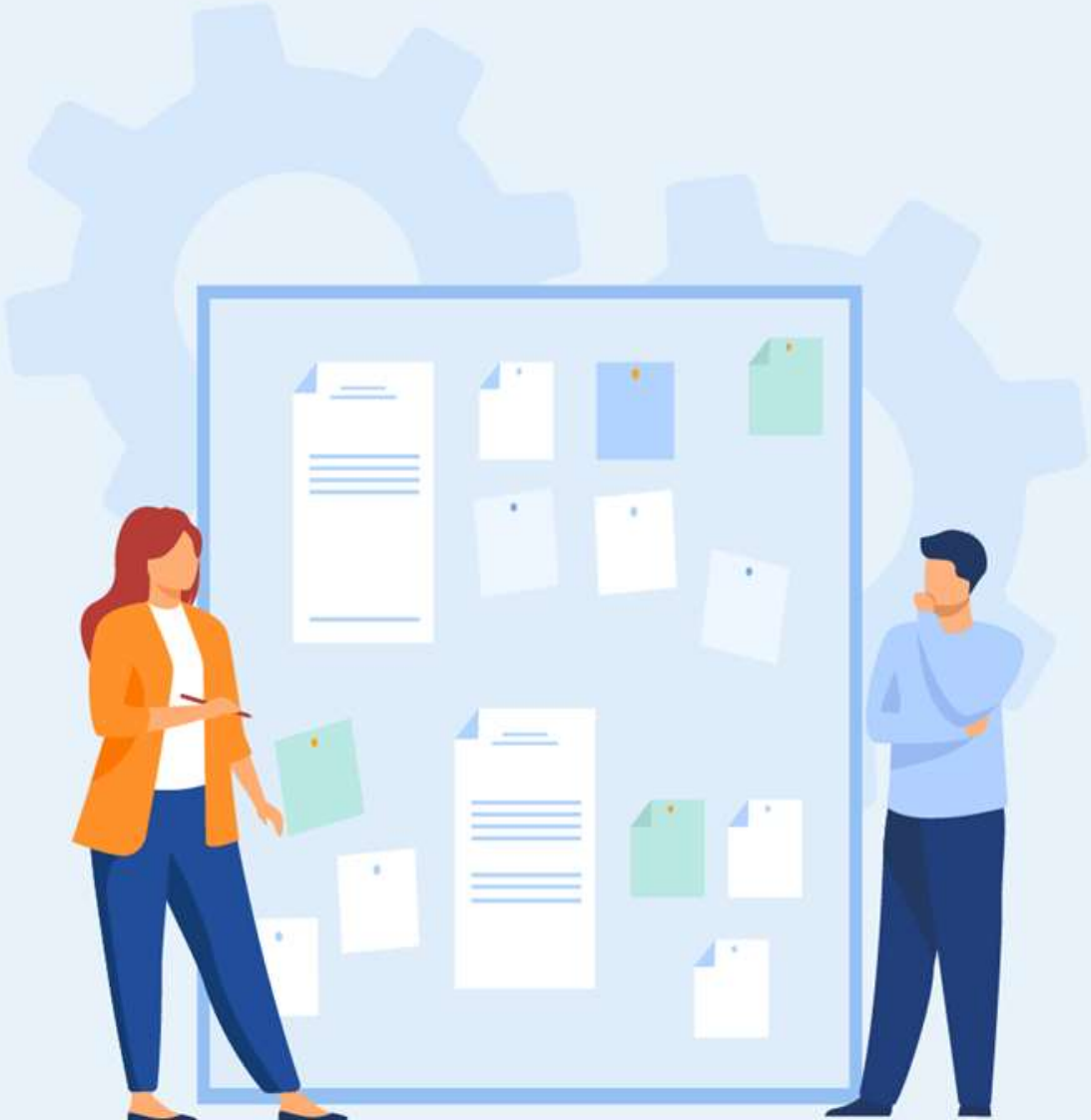
Afin d'ajouter les certificats **SSL** sur Android, il faut créer une méthode qui génère un **OkHttpClient.**

Les étapes à suivre pour intégrer les certificats SSL :

1. Ajoutez le certificat SSL dans le projet en tant que ressource brute.
2. Créez une méthode qui renvoie un fichier OkHttpClient.
3. Créez un KeyStore contenant les autorités de certification de confiance.
4. Créez une KeyManagerFactory .
5. Créez un contexte SSL contenant les autorités de certification de confiance.
6. Ajoutez l'usine de sockets au constructeur.
7. Utilisez le OkHttpClient généré comme d'habitude.

OkHttp est un client HTTP, basé sur Okio pour la lecture/écriture du flux de données. Son objectif est de prendre en charge la communication avec le serveur.

Copyright - Tout droit réservé - OFPPT



CHAPITRE 2

Identifier les fondamentaux de sécurité d’une application mobile

1. Utilisation de OAUTH
2. Intégration de SSL
3. **Sécurisation avec TLS**
4. Protection avec Proguard

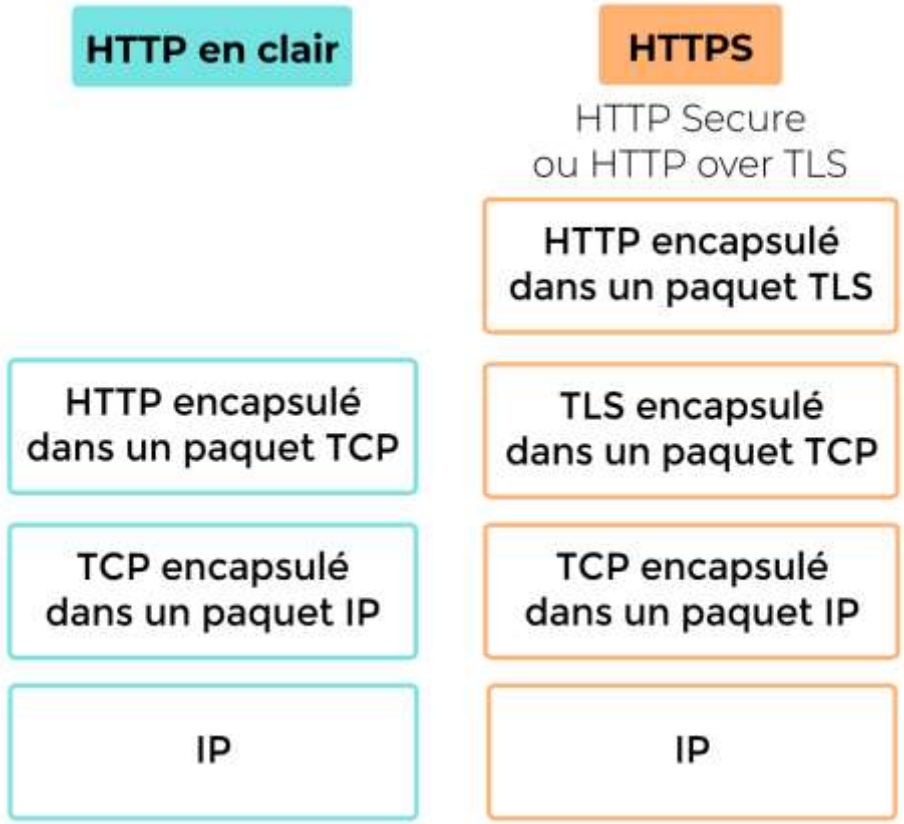
Sécurisation avec TLS

Le **protocole TLS (Transport Layer Security)** est le protocole cryptographique le plus utilisé pour sécuriser les connexions réseaux. C'est le successeur du protocole SSL.

Ce protocole permet de garantir :

- **La confidentialité** des données avec le chiffrement symétrique
- **L'intégrité** des données avec un code MAC
- **L'échange d'une clé de session** avec le chiffrement asymétrique
- **L'authentification du serveur** avec les certificats signés par une autorité de certification.

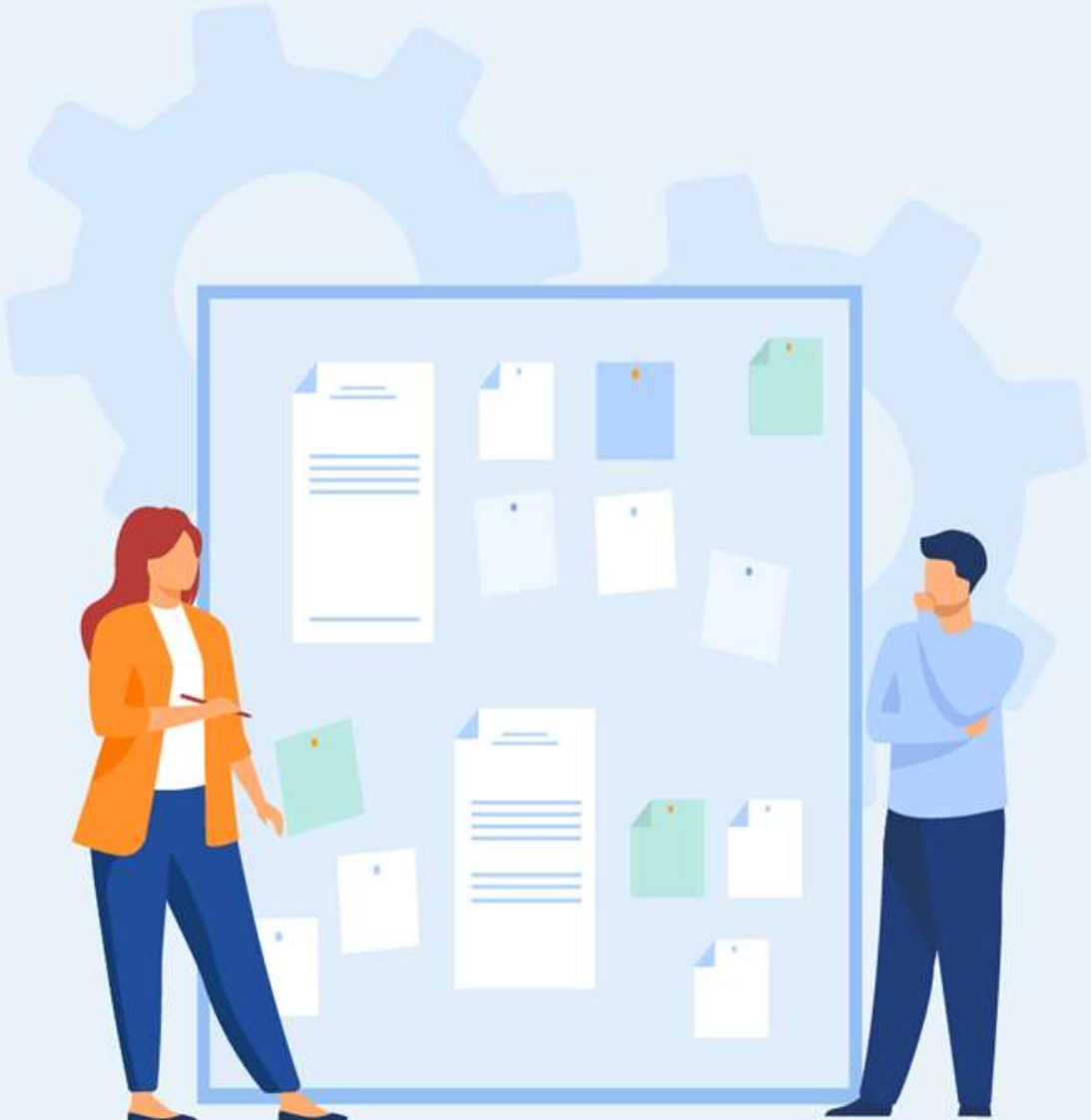
SSL est la technologie de sécurité standard pour établir un lien crypté entre un client et un serveur. Ce lien garantit que toutes les données transmises entre le serveur Web et le navigateur restent privées.



source : https://openclassrooms.com/

PARTIE 1

Copyright - Tout droit réservé - OFPPT



CHAPITRE 2

Identifier les fondamentaux de sécurité d’une application mobile

1. Utilisation de OAUTH
2. Intégration de SSL
3. Sécurisation avec TLS
4. **Protection avec Proguard**

Proguard

ProGuard est un outil de ligne de commande open source qui réduit, optimise et obscurcit le code Java. Il est capable d'optimiser le bytecode ainsi que de détecter et de supprimer les instructions inutilisées.

La principale fonction de proguard: **Obfuscation** .
Il a également deux autres fonctions importantes : **Shrinking** et **Optimization**.

```
buildTypes {
    debug {
        debuggable true
        minifyEnabled false
        shrinkResources false
    }
    release {
        minifyEnabled true
        proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
        shrinkResources true
    }
}
```

Code à ajouter au fichier build.gradle pour activer Proguard

Obfuscation : est un processus qui renomme les classes, champs et méthodes restants en utilisant des noms courts sans signification et rend votre APK difficile à désosser.

Shrinking : détecte et supprime les codes et les ressources inutilisés.

Optimization : est un processus qui rend l'APK aussi petit que possible.

Remarques

- Proguard n'empêche pas la rétro-ingénierie. Mais cela rend les choses plus difficiles, "Le code obscurci rend votre APK difficile à désosser" selon la documentation des développeurs Android.

PARTIE 1

Copyright - Tout droit réservé - OFPPT



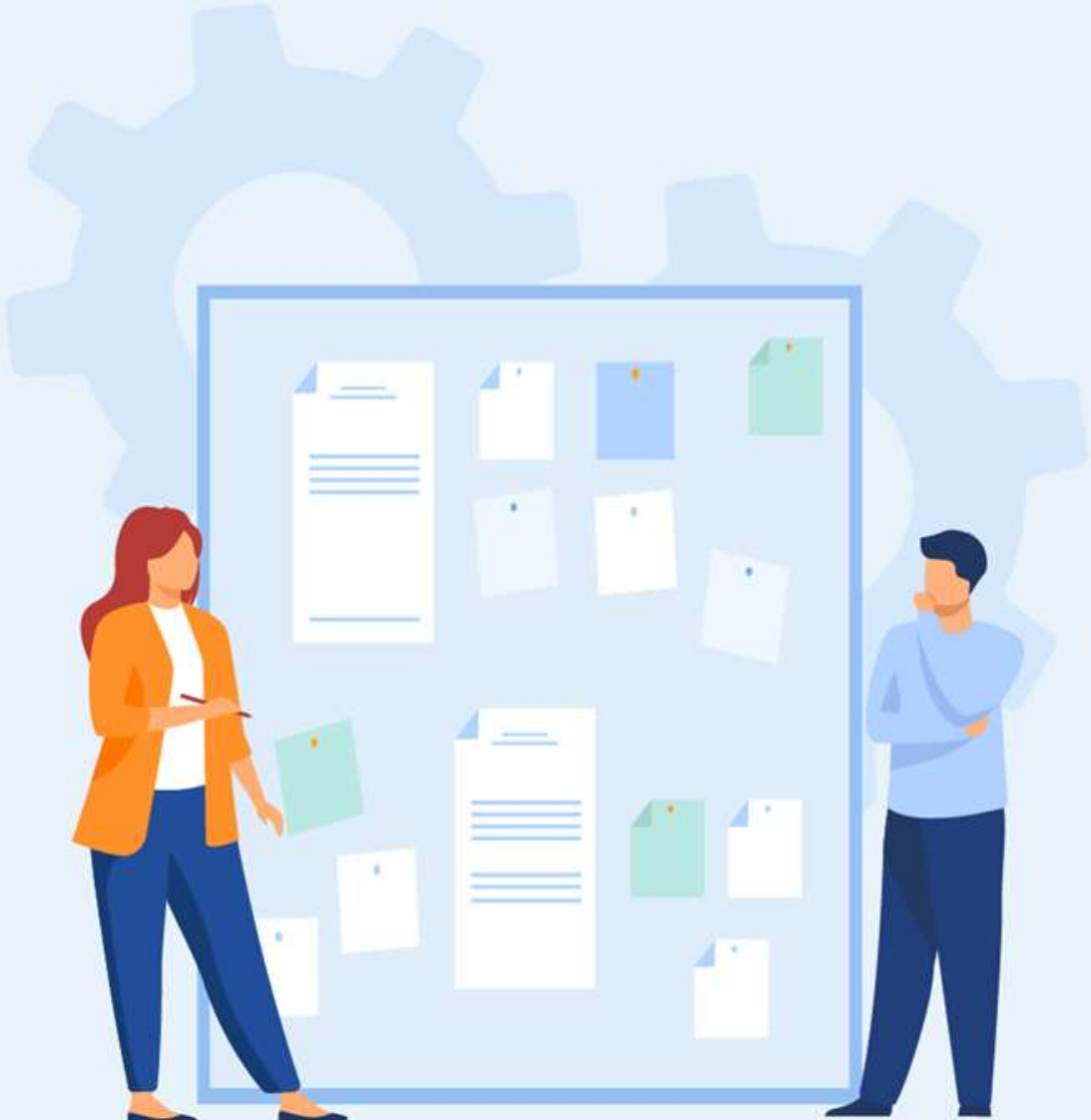
CHAPITRE 3

Consommer des API sous Android

Ce que vous allez apprendre dans ce chapitre :

- Manipulation de SOAP
- Manipulation de REST API
- Retrofit
- Présentation de Volley

06 heures



CHAPITRE 3

Consommer des API sous Android

1. Manipulation de SOAP
2. Manipulation de REST API
3. Retrofit
4. Présentation de Volley

Manipulation de SOAP

Android fournit trois types d'analyseurs XML qui sont **DOM**, **SAX** et **XMLPullParser** .
Il est recommandé d'utiliser **XMLPullParser** car il est efficace et facile.

La première étape consiste à identifier les champs des données XML qui vous intéressent.

Par exemple. Dans le XML ci-dessous, nous nous intéressons uniquement à la température.

```
 <city id="2643743" name="London">
        <temperature value="289.54" min="289.15"
max="290.15" unit="kelvin"/>
   </city>
```

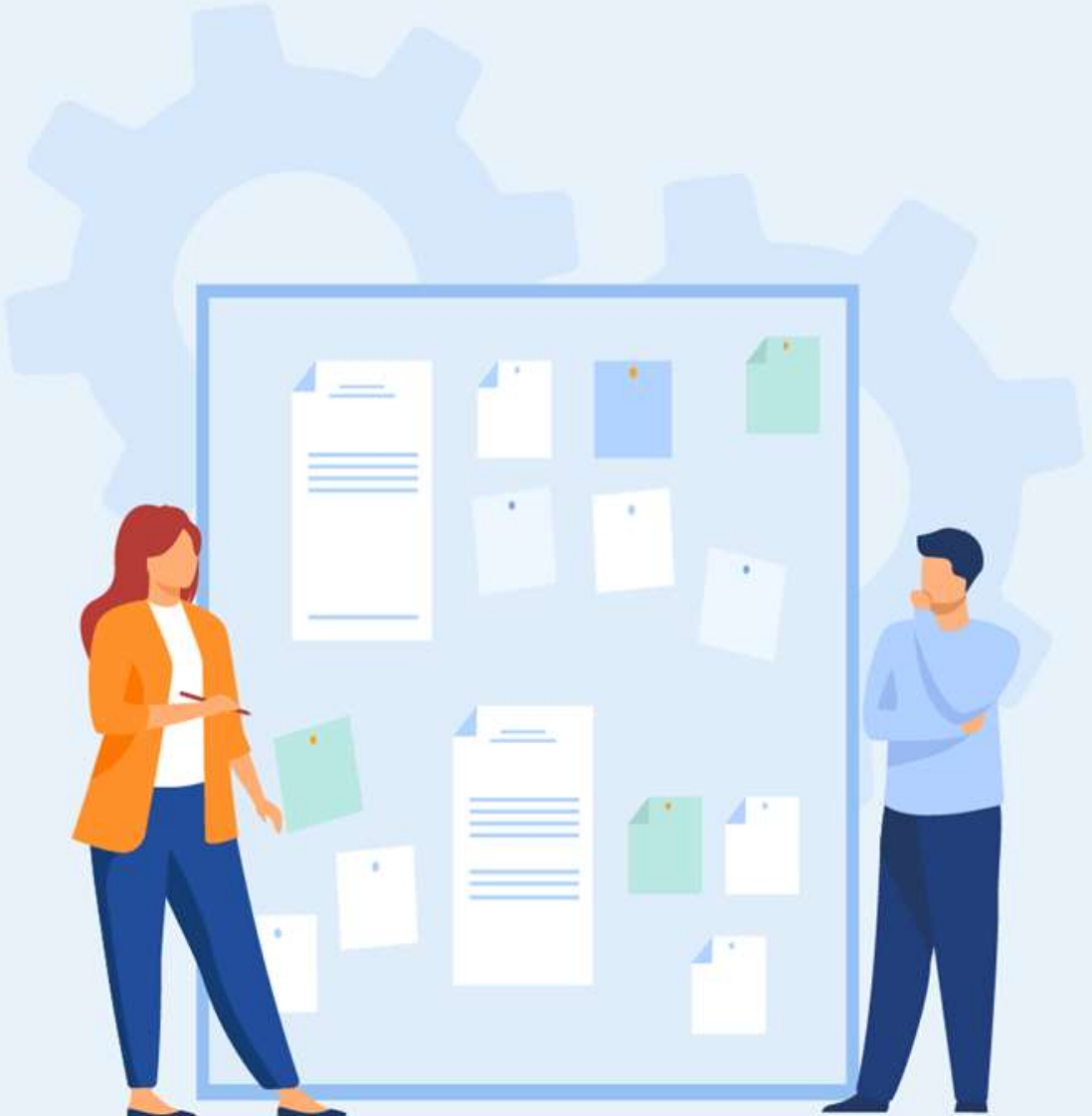
- Code à utiliser pour manipuler un flux XML

```
int event = myParser.getEventType();
while (event != XmlPullParser.END_DOCUMENT)  {
  String name=myParser.getName();
  switch (event){
    case XmlPullParser.START_TAG:
    break;

    case XmlPullParser.END_TAG:
    if(name.equals("temperature")){
      temperature = myParser.getAttributeValue(null,"value");
    }
    break;
  }
  event = myParser.next();

}
```


Copyright - Tout droit réservé - OFPPT



CHAPITRE 3

Consommer des API sous Android

1. Manipulation de SOAP
2. **Manipulation de REST API**
3. Retrofit
4. Présentation de Volley

REST API

Les informations qui se trouvent dans une requête HTTP/HTTPS :

- **La méthode**. Il s'agit de l'action que l'on souhaite faire : récupérer une ressource, la supprimer, etc…

  Voici les méthodes HTTP les plus courantes :

  - **GET** : permet de **récupérer** des ressources, comme par exemple le temps actuel sur un service de météo ;
  - **POST** : permet de **créer** ou **modifier** une ressource, comme la création d'un nouvel utilisateur sur votre application ;
  - **PUT** : permet de **modifier** une ressource, comme le nom de l'utilisateur que vous venez de créer avec *POST* ;
  - **DELETE** : Permet de **supprimer** une ressource, comme un commentaire dans un fil de discussion.

- **L'URL**. C'est l'adresse sur le service web que vous souhaitez atteindre. Un peu comme un identifiant unique afin que le web service comprenne ce que vous voulez
- **Les données**. Lorsqu'on fait une requête pour enregistrer des données (par exemple un formulaire) il faut pouvoir envoyer ces données au service web.

Copyright - Tout droit réservé - OFPPT

Manipulation de REST API

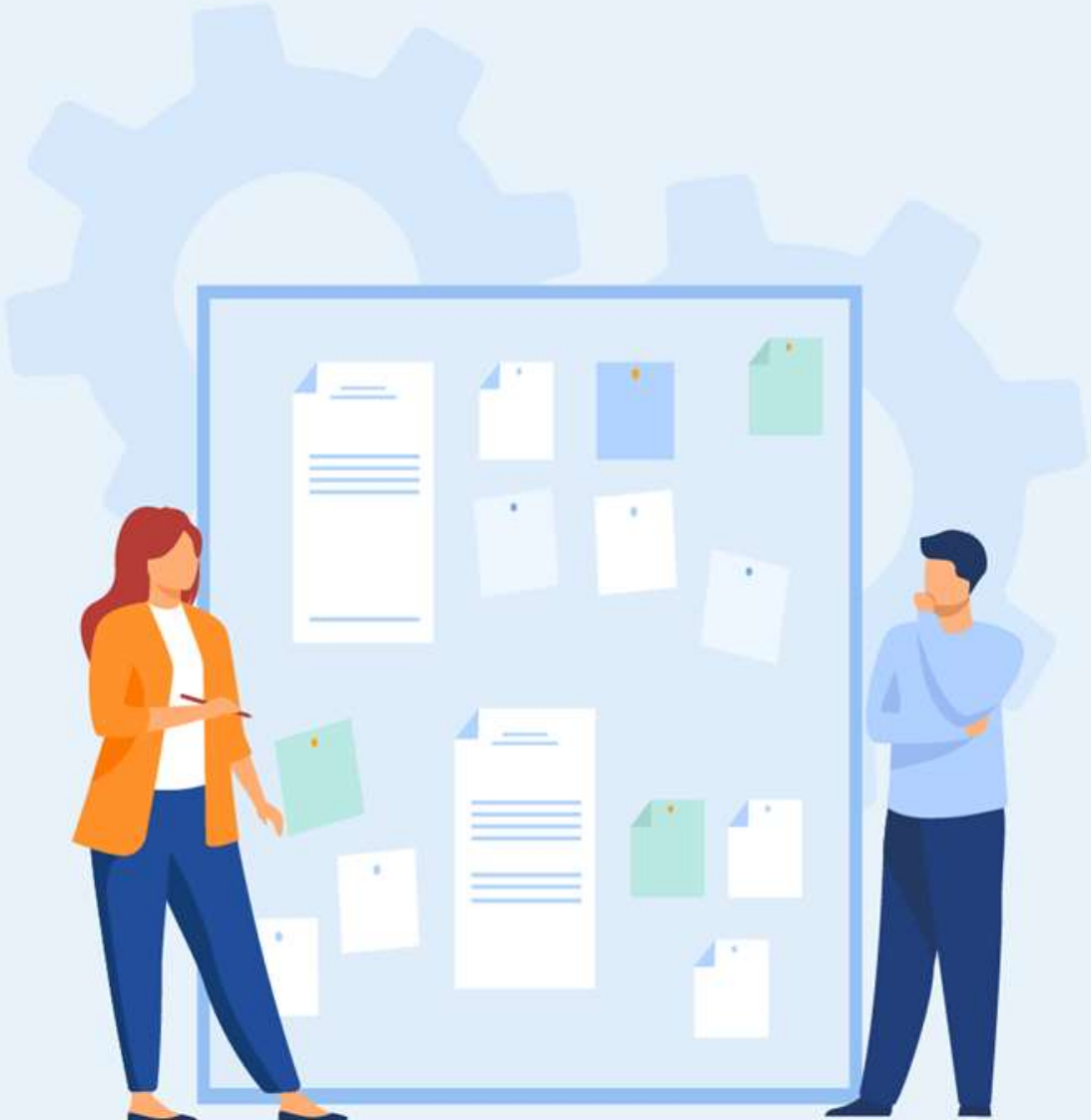
Pour le cas d'un API REST d'un restaurant, afin de gérer les commandes, nous aurons besoins des endPoints suivants :

Verbe HTTP	endPoint	Actions
GET	/commands	Lister les commandes
POST	/commands	Passer une commande
GET	/commands/123	Détails de la commande 123
PUT	/commands/123	Modifie la commande 123
DELETE	/commands/123	Annule la commande 123

EndPoint : le code qui permet à deux programmes logiciels de communiquer entre eux.

- **Le code HTTP**. Il s'agit d'un code numérique qui vous indique comment s'est déroulée la requête. Voici les plus courants :
  - **200** : indique que tout s'est bien passé
  - **400** : indique que votre requête n'est pas conforme à ce qui est attendu
  - **401** : indique que vous devez être authentifié pour faire cette requête
  - **403** : indique que vous êtes bien authentifié mais que vous n'êtes pas autorisé à faire cette requête
  - **404** : indique que la ressource demandée n'existe pas
  - **500** : indique une erreur avec le service web

Copyright - Tout droit réservé - OFPPT



CHAPITRE 3

Consommer des API sous Android

1. Manipulation de SOAP
2. Manipulation de REST API
3. **Retrofit**
4. Présentation de Volley

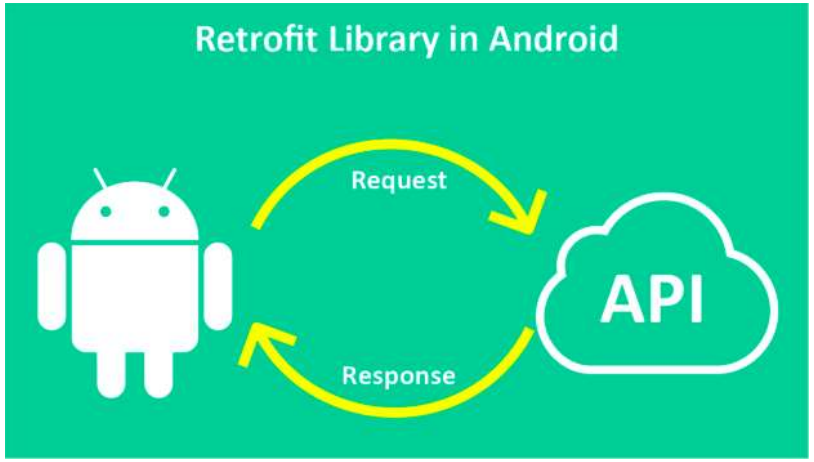
Retrofit

Retrofit est un client REST de type sécurisé pour Android et Java qui vise à faciliter l'utilisation des services Web RESTful.

https://square.github.io/retrofit/

Il existe huit annotations intégrées : HTTP, GET, POST, PUT, PATCH, DELETE, OPTIONS et HEAD.

L'URL relative de la ressource est spécifiée dans l'annotation.
par exemple : @**GET** ("utilisateurs/liste")



source : https://innovationm.co/

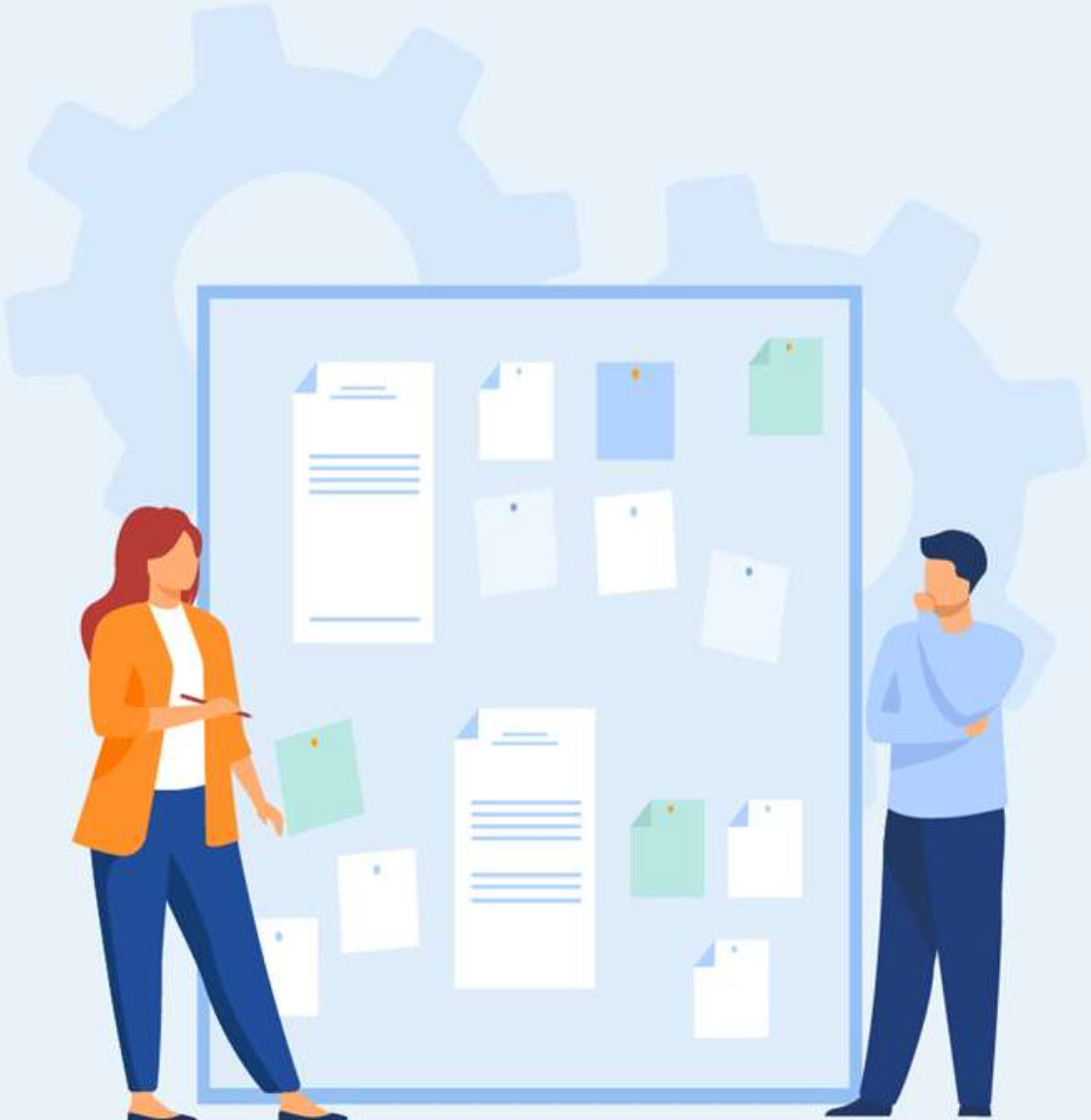
Remarques

Pour intégrer GRADLE :

```
implementation 'com.squareup.retrofit2:retrofit:(insert latest version)'
```

PARTIE 1

Copyright - Tout droit réservé - OFPPT



CHAPITRE 3

Consommer des API sous Android

1. Manipulation de SOAP
2. Manipulation de REST API
3. Retrofit
4. **Présentation de Volley**

Présentation de Volley

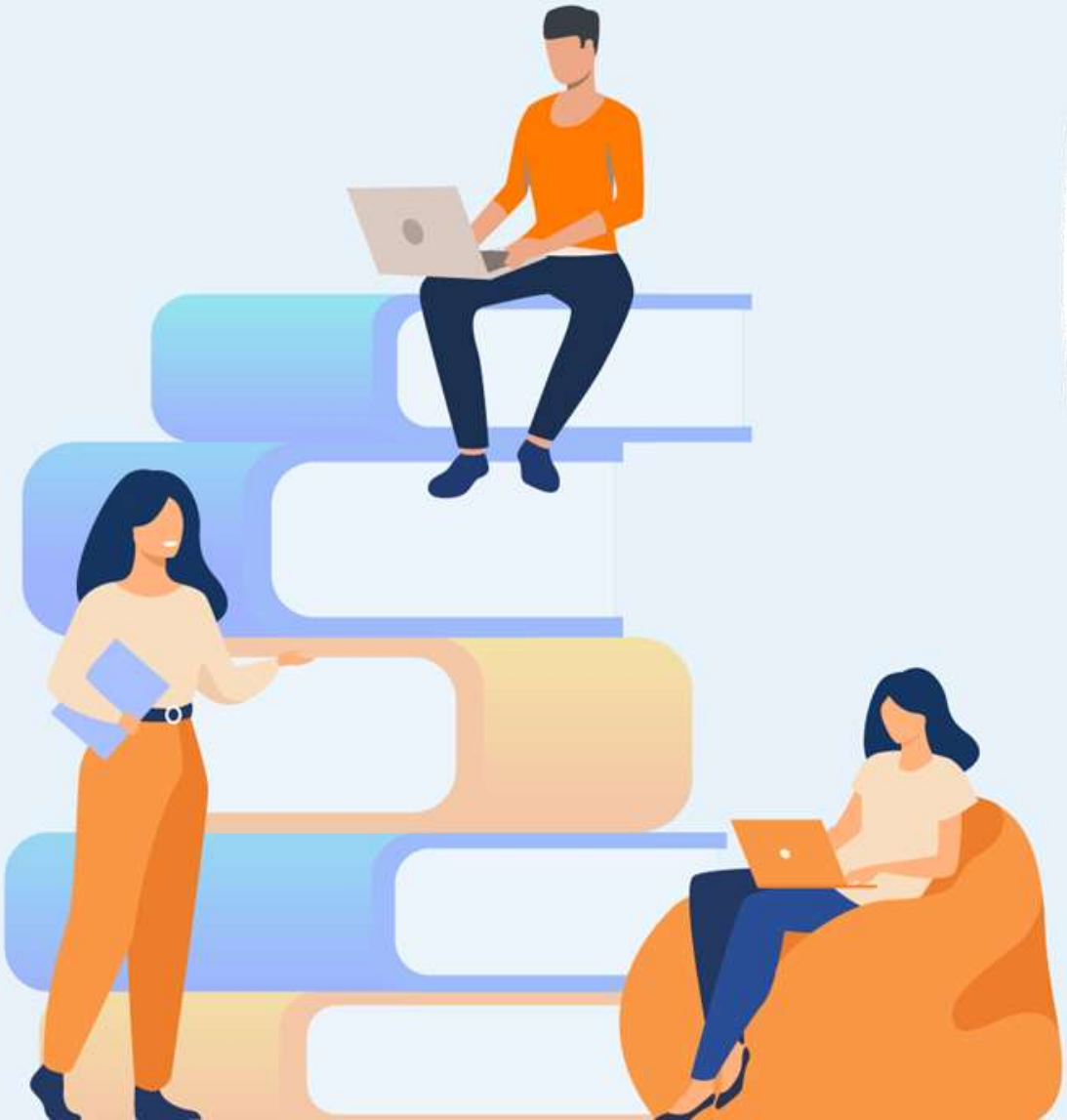
Volley est une bibliothèque HTTP qui facilite la mise en réseau des applications Android.

https://google.github.io/volley/

Les avantages de Volley :

- Planification automatique des requêtes réseau.
- Plusieurs connexions réseau simultanées.
- Mise en cache transparente des réponses disque et mémoire avec cohérence de cache HTTP standard .
- Prise en charge de la priorisation des demandes.
- API de demande d'annulation. Vous pouvez annuler une seule demande ou définir des blocs ou des étendues de demandes à annuler.
- Facilité de personnalisation, par exemple, pour les nouvelles tentatives et les interruptions.
- Un ordre solide qui facilite le remplissage correct de votre interface utilisateur avec des données extraites de manière asynchrone du réseau.
- Outils de débogage et de traçage.

Copyright - Tout droit réservé - OFPPT



PARTIE 2

Gérer les différentes méthodes de stockage de données sous Android

Dans ce module, vous allez :

- Gérer les fichiers sous Android
- Manipuler des bases de données sous Android
- Partager des données entre applications

16 heures



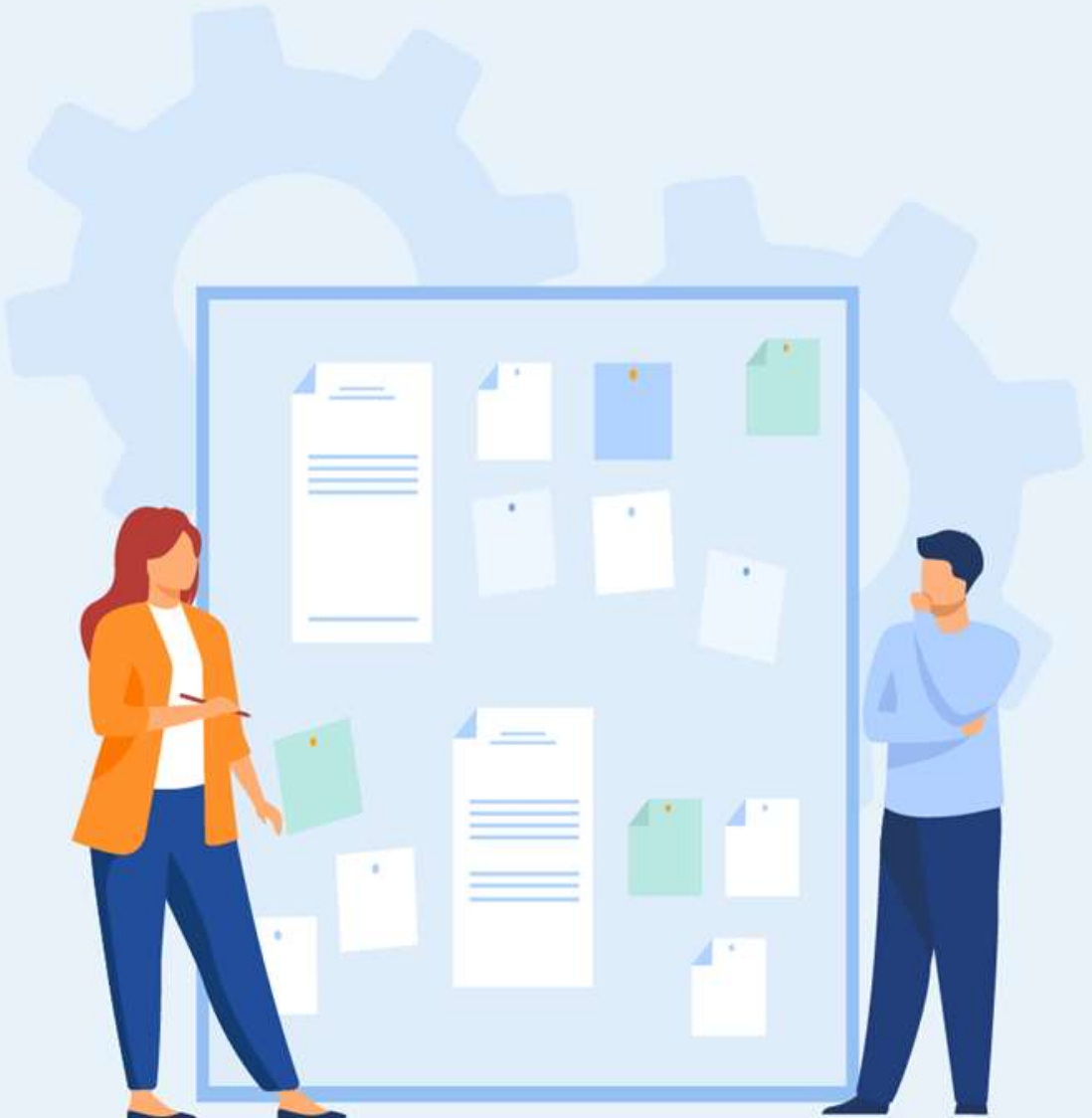
Chapitre 1

Partager des données entre applications

Ce que vous allez apprendre dans ce chapitre :

- Utilisation du ContentProvider
- Protection contre les entrées malveillantes

08 heures



Chapitre 1

Partager des données entre applications

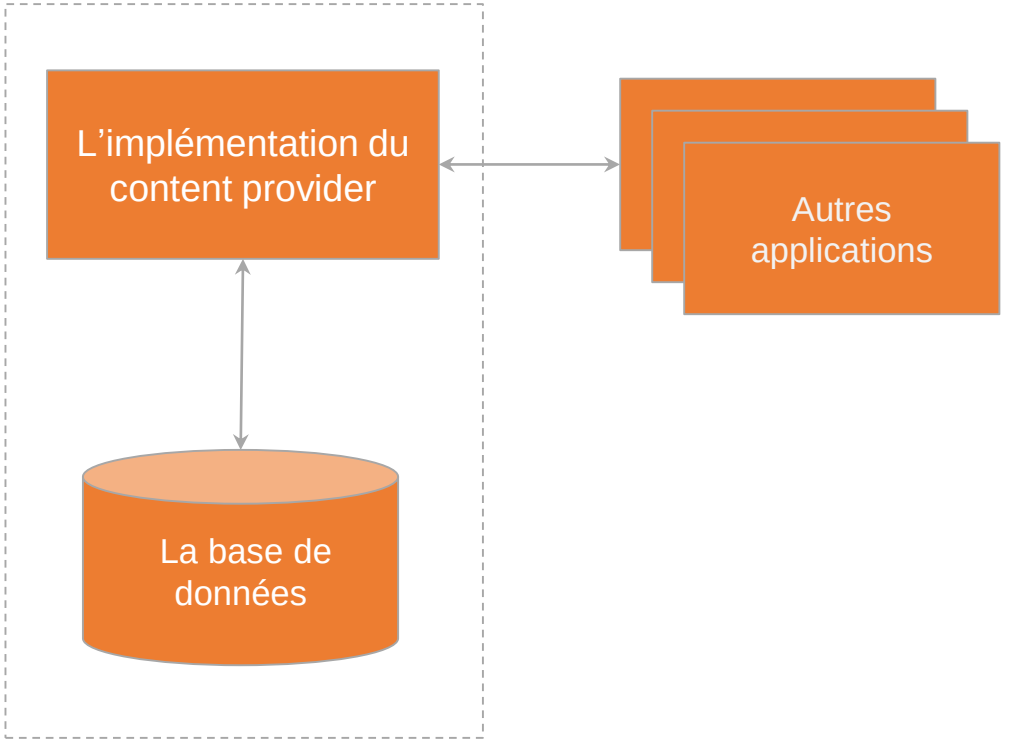
1. **Utilisation du ContentProvider**
2. Protection contre les entrées malveillantes

ContentProvider

Le **ContentProvider** permet d'exposer une base de données.

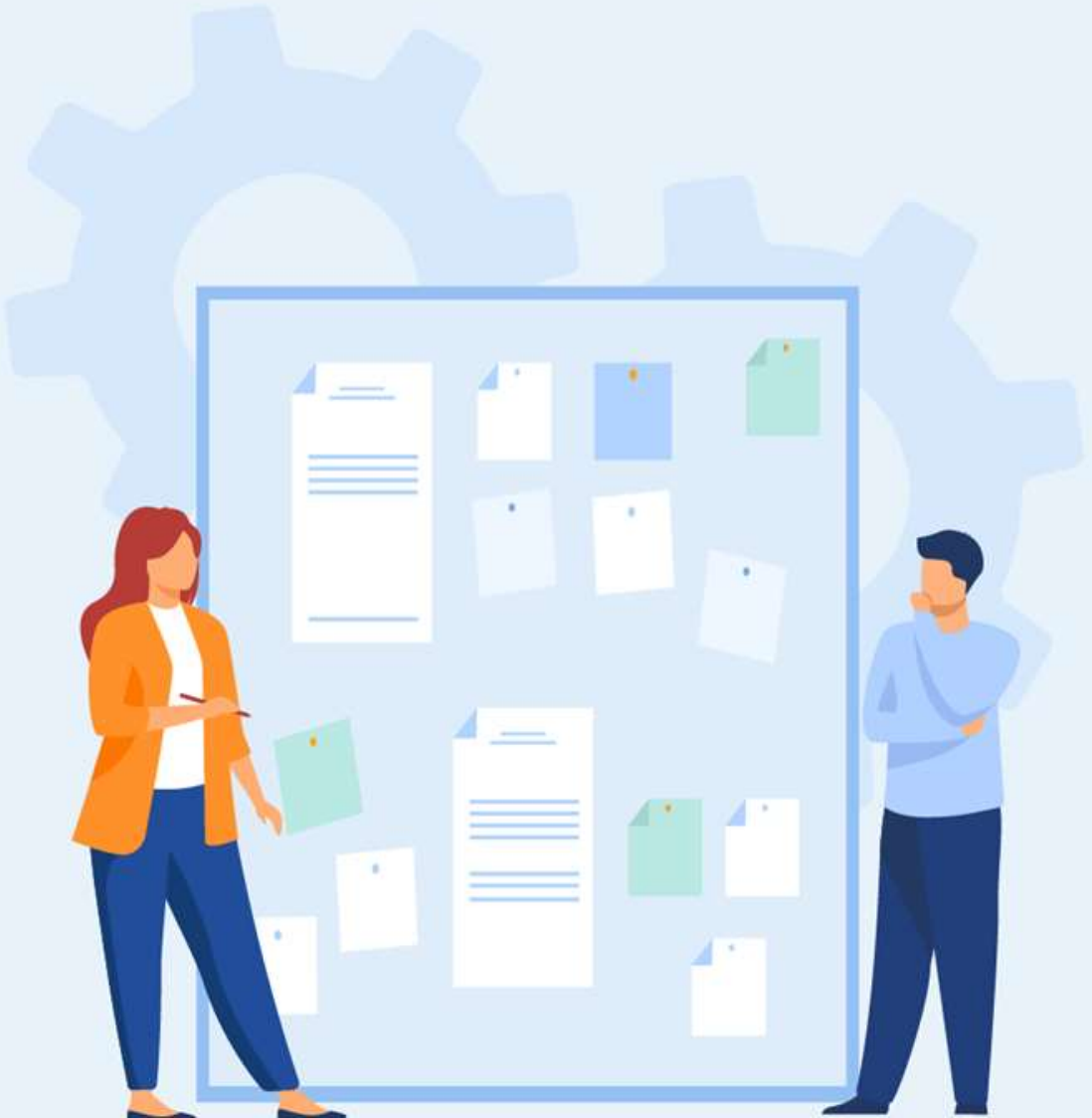
Nous pouvons ainsi récupérer, insérer, supprimer ou modifier notre base depuis l'extérieur de notre application.

Pour communiquer avec notre ContentProvider, on utilise un **ContentResolver** qui prend en paramètre une URI correspondant au provider cible et aux paramètres à envoyer pour exécuter la requête.



source : openclassrooms.com

Copyright - Tout droit réservé - OFPPT



Chapitre 1

Partager des données entre applications

1. Utilisation du ContentProvider
2. **Protection contre les entrées malveillantes**

SQL Injection locale

L'injection SQL exploite des applications vulnérables en insérant du code dans les instructions SQL pour accéder aux bases de données. L'attaque peut exposer des données privées, corrompre le contenu d'une base de données et même compromettre l'infrastructure du backend.

cette requête renvoie toutes les lignes d'une table :

SELECT * FROM purchases WHERE email='admin@app.com' OR 1=1;

Appliquer des mesures de sécurité réseau (le trafic TLS)

Si votre appli communique avec un serveur Web disposant d'un certificat émis par une autorité de certification (CA) bien connue, utilisez une requête **HTTPS.**

Copyright - Tout droit réservé - OFPPT

Utiliser les objets WebView avec soin

Les objets WebView de votre application ne doivent pas autoriser les utilisateurs à accéder à des sites que vous ne contrôlez pas. Dans la mesure du possible, utilisez une liste d'autorisation pour limiter le contenu chargé par les objets WebView de votre application.

Ajouter une configuration de sécurité réseau

Si votre application utilise de nouvelles CA ou des CA personnalisées, vous pouvez déclarer les paramètres de sécurité de votre réseau dans un fichier de configuration.

Ce processus vous permet de créer la configuration sans modifier le code de l'application.

Copyright - Tout droit réservé - OFPPT



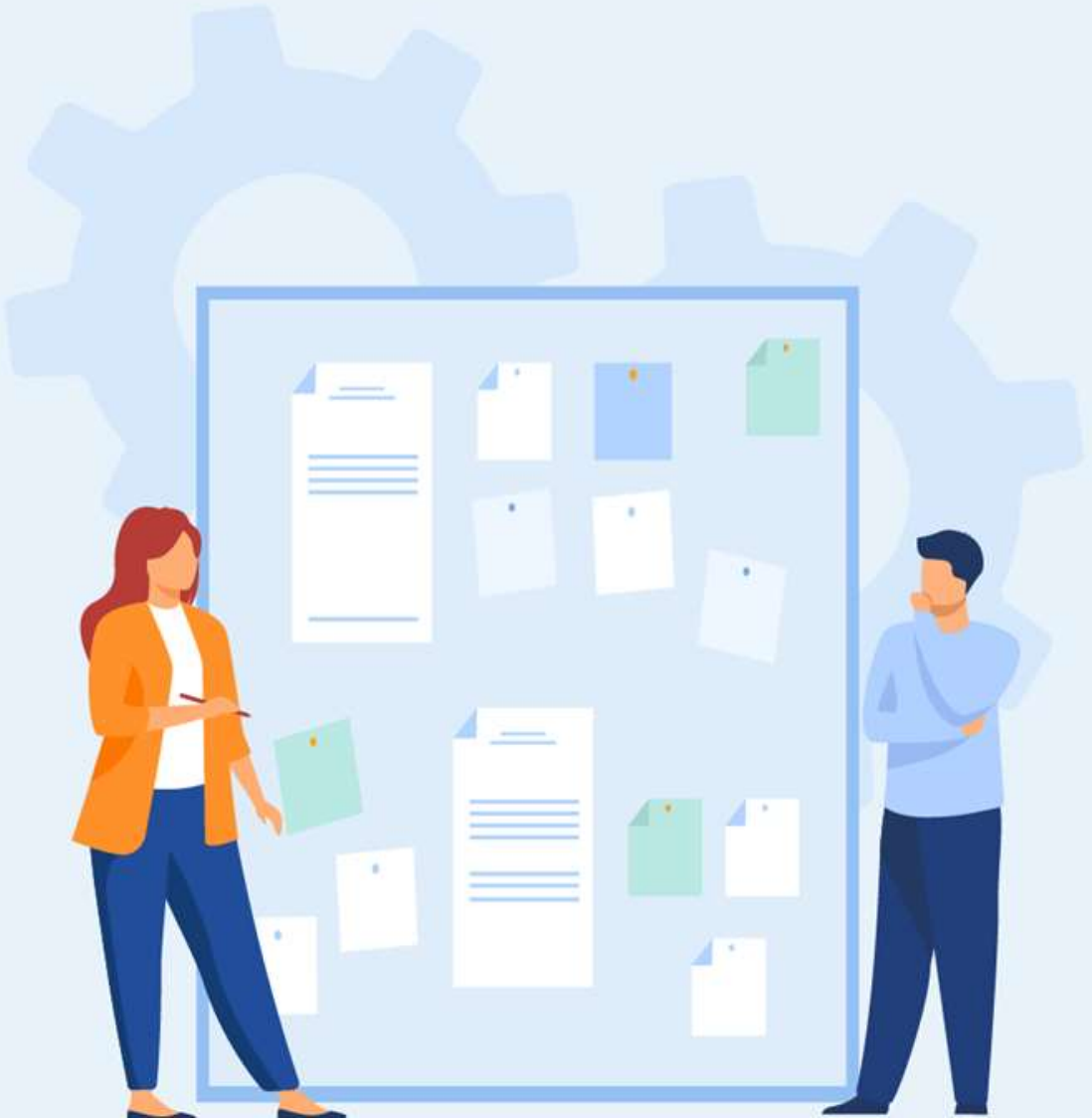
Chapitre 2

Manipuler les données sous Android

Ce que vous allez apprendre dans ce chapitre :

- Utilisation de SqlLite
- Découverte de ROOM
- Utilisation des SharePreferences
- Manipulation des fichiers

08 heures



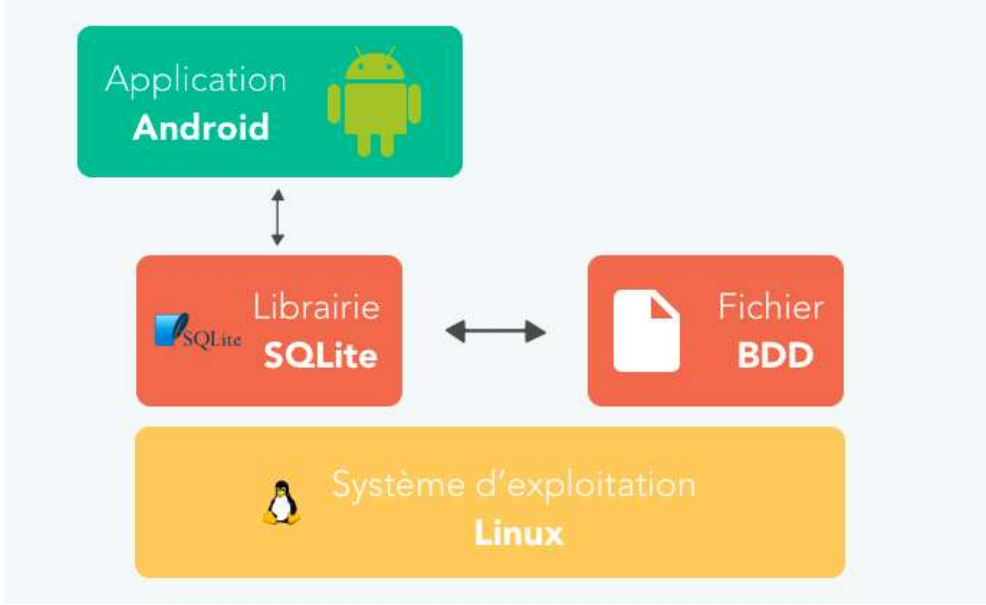
Chapitre 2

Manipuler les données sous Android

1. **Utilisation de SqlLite**
2. Découverte de ROOM
3. Utilisation des SharePreferences
4. Manipulation des fichiers

SQLite

SQLite est un **moteur de base de données relationnelle** entièrement manipulable grâce au langage SQL. Contrairement aux serveurs de base de données traditionnels comme MySQL, PostgreSQL ou encore Microsoft SQL Server, l'intégralité de la base de données est **stockée dans un fichier** (et non sur un serveur distant).



source : https://openclassrooms.com/

Copyright - Tout droit réservé - OFPPT

Utilisation de SQLite

- **Insérer des données :**

```
final SQLiteDatabase database = openHelper.getWritableDatabase();
final int id = database.insert(TABLE_NAME, null, values);
```

openHelper : Cette classe permet de différer facilement l'ouverture et la mise à niveau de la base de données.

- **Supprimer des données :**

```
database.delete(TABLE_NAME, null, null);
database.close();
```

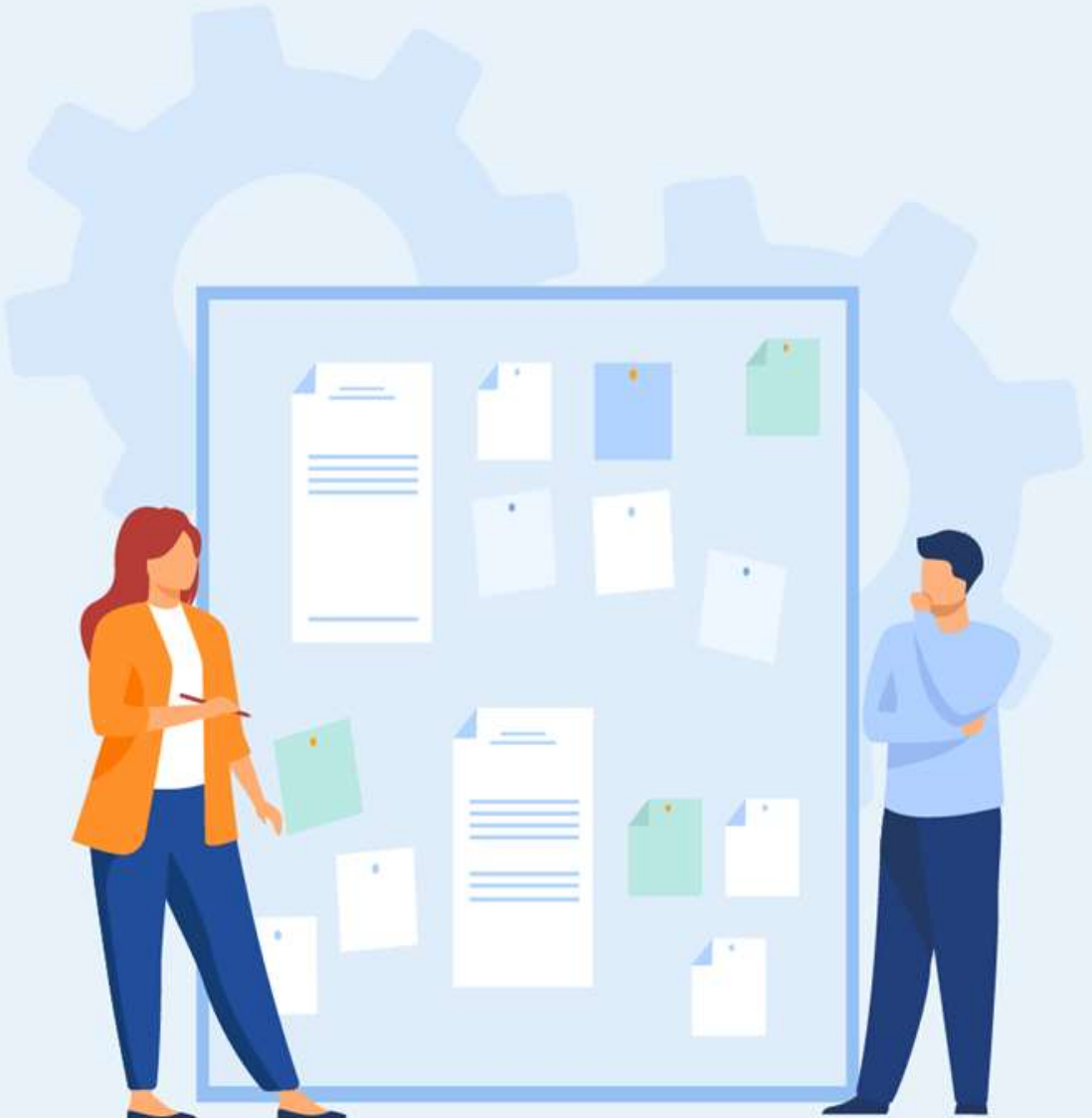
- **Mettre à jour des données :**

```
final int count = database.update(TABLE_NAME, values, COLUMN_ID + " = ?", new String[]
{String.valueOf(model.getId())});
```

- **Lire des données :**

```
String[] selectionArgs = new String[]{name};
  Cursor c = database.rawQuery("SELECT * FROM " + tabl1 +  " WHERE " + name + " = ? ", selectionArgs);
```


Copyright - Tout droit réservé - OFPPT



Chapitre 2

Manipuler les données sous Android

1. Utilisation de SqlLite
2. **Découverte de ROOM**
3. Utilisation des SharePreferences
4. Manipulation des fichiers

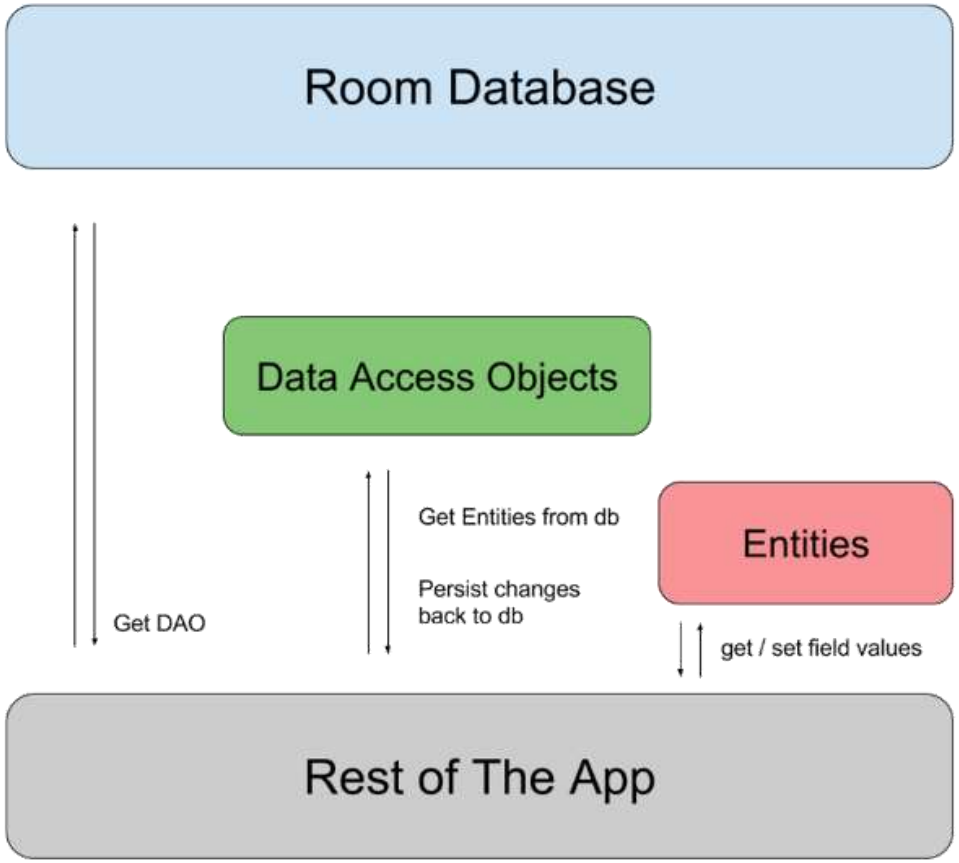
ROOM

Room est l'ORM officiel pour mettre en place une persistance de données dans une application Android.

Cette librairie brille par son efficacité, sa simplicité et son système de mise en cache.

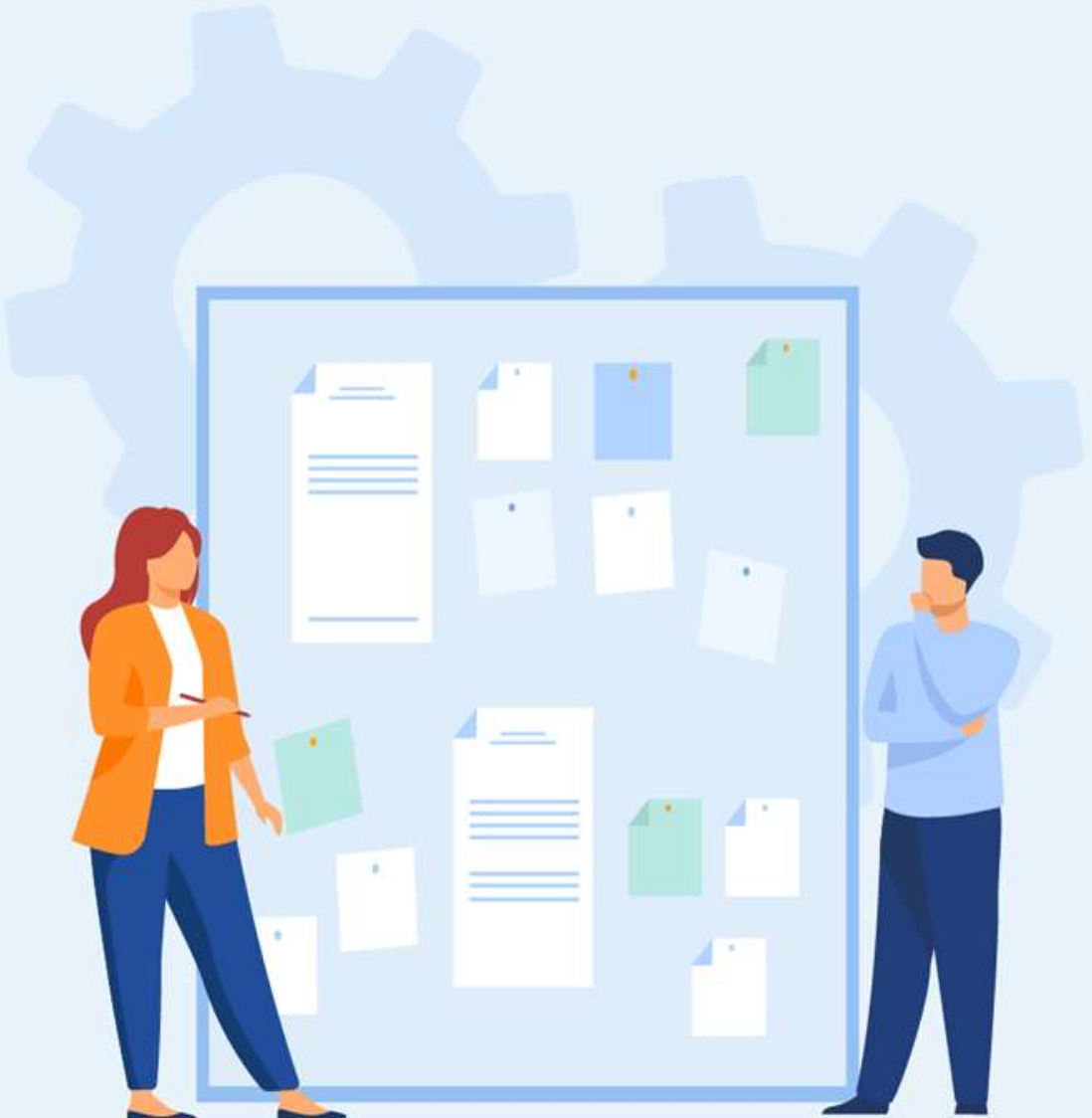
Room repose sur **trois** composants principaux :

- La classe de base de données.
- Les entités de données.
- Les objets d'accès aux données (DAO).



source : https://developer.android.com/

Copyright - Tout droit réservé - OFPPT



Chapitre 2

Manipuler les données sous Android

1. Utilisation de SqlLite
2. Découverte de ROOM
3. **Utilisation des SharedPreferences**
4. Manipulation des fichiers

SharedPreferences

L'API mise à disposition par Android pour sauvegarder des informations locales à l'application s'appelle **SharedPreferences**.

C'est une couche d'abstraction qui vous facilite la vie : elle récupère les données que vous lui passez, et les stocke dans un fichier XML.

L'API offre la possibilité d'utiliser différents fichiers pour stocker les informations. C'est très pratique dans une application importante avec des données très variées.

Editor editor = sharedpreferences.edit();

- **Ecriture :**

editor.putString("key", "value").commit();

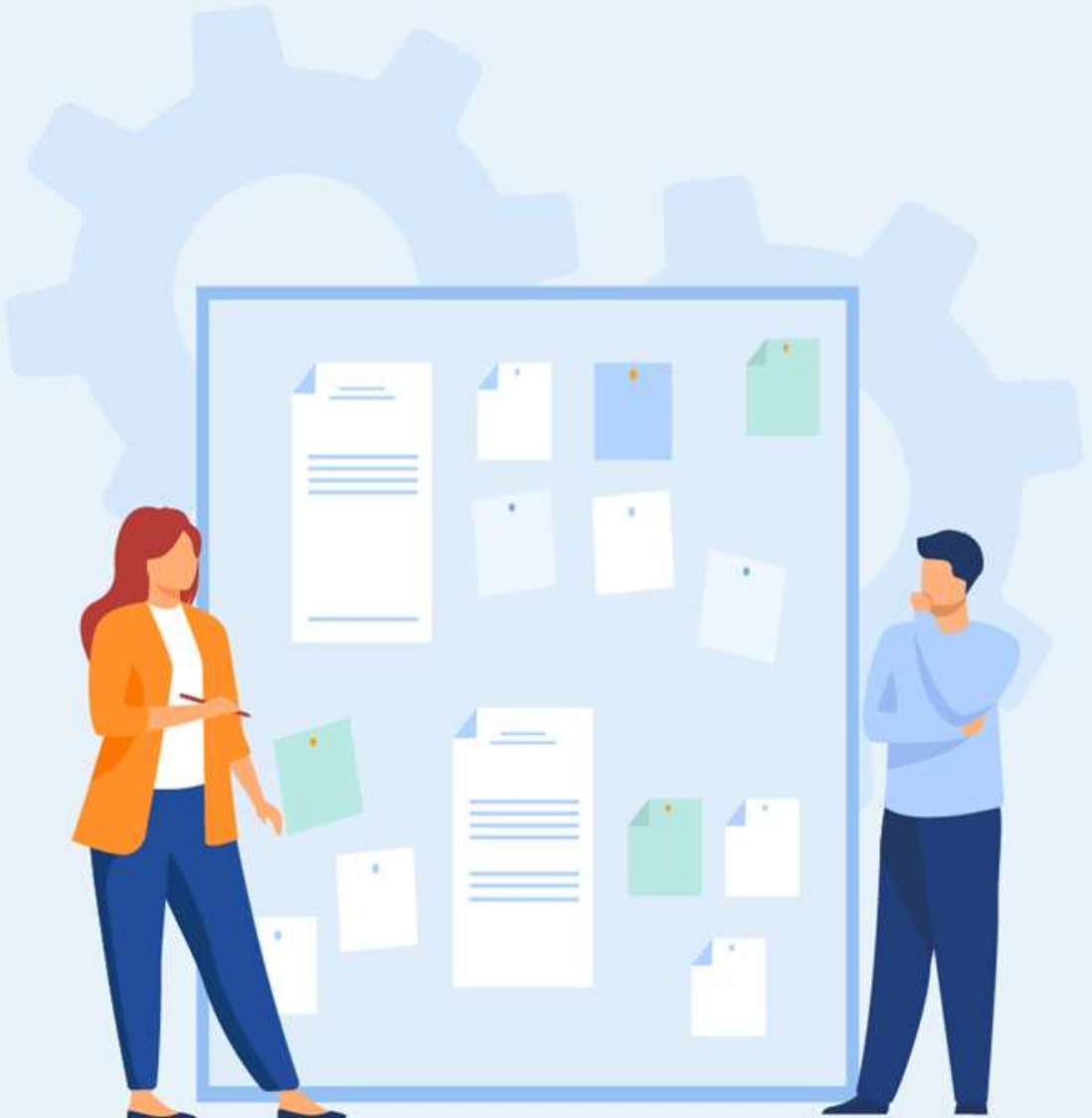
- **Lecture :**

String value = sharedpreferences.getString("key", null);

- **Suppression :**

editor.remove("key").commit();

Copyright - Tout droit réservé - OFPPT



Chapitre 2

Manipuler les données sous Android

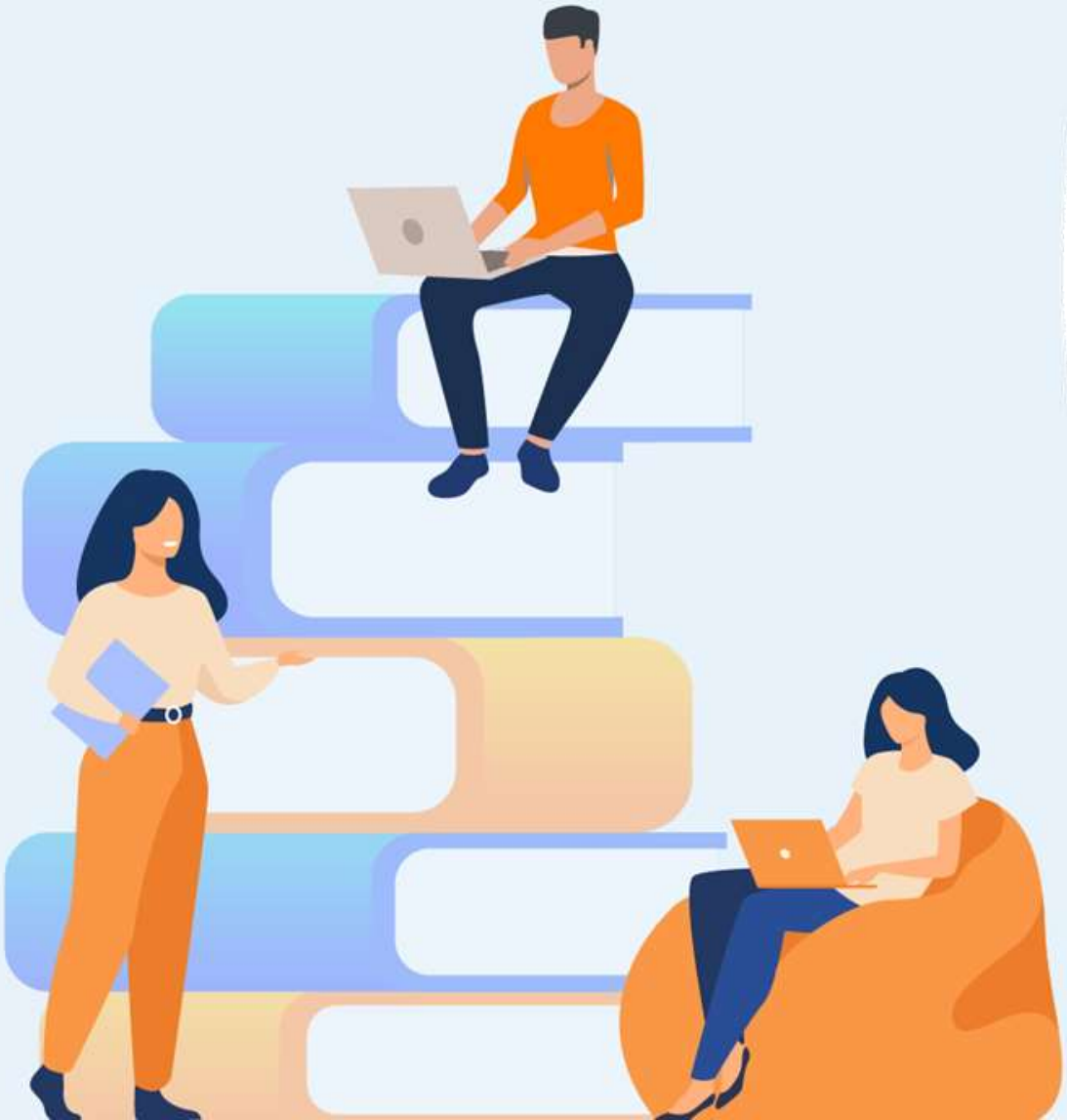
1. Utilisation de SqlLite
2. Découverte de ROOM
3. Utilisation des SharedPreferences
4. **Manipulation des fichiers**

Manipulation des fichiers

Pour écrire du texte dans **un fichier** qui sera mémorisé dans l'espace de stockage externe ou interne du téléphone, selon le deux cas :

- **Externe**
  - Le niveau de confidentialité :
    - **Public** : Le fichier sera stocké sur l'espace de *stockage externe* en mode public. Il ne sera donc *pas supprimé* quand l'utilisateur désinstallera son application.
    - **Privé** : Le fichier sera stocké sur l'espace de *stockage externe* en mode privé. Il sera donc *supprimé* quand l'utilisateur désinstallera son application.
- **Interne**
  - Le fichier est stocké en cache ?
    - **Oui** : Le fichier sera stocké sur l'espace de *stockage interne* dans le répertoire dédié au cache. Il pourra donc être *supprimé à n'importe quel moment*.
    - **Non** : Le fichier sera stocké sur l'espace de stockage interne.

Copyright - Tout droit réservé - OFPPT



PARTIE 3

Publier une application sur le store

Dans ce module, vous allez :

- Gérer des signatures et des versions
- Publier d’une application sur le store
- Utiliser des plateformes de déploiements continus

08 heures



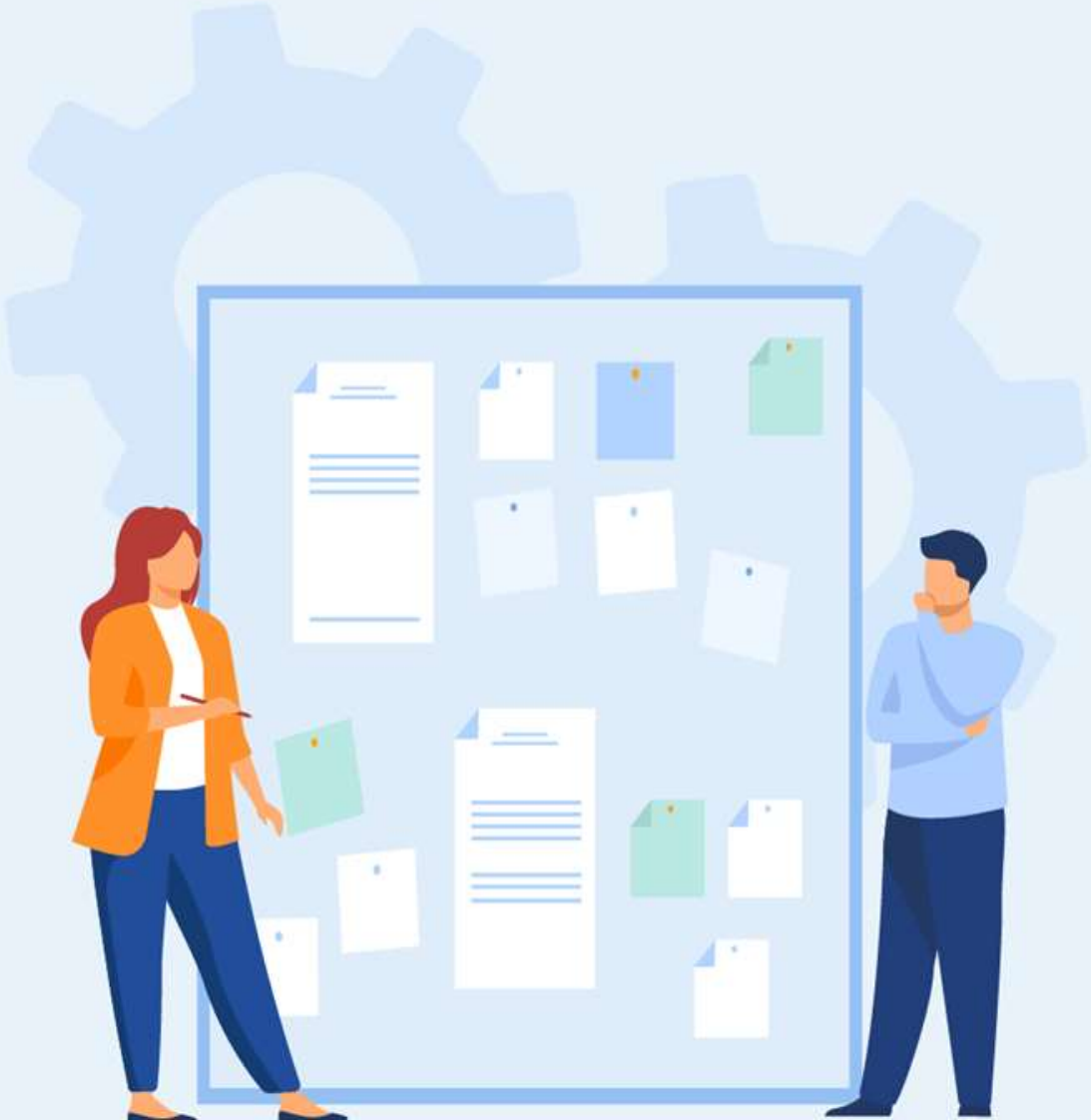
Chapitre 1

Gérer les signatures et versions

Ce que vous allez apprendre dans ce chapitre :

- Configuration de l’application pour la release
- Génération de l’upload key et keystore
- Signature de l’application avec l’upload key
- Déploiement de la version de l’application

04 heures



Chapitre 1

Gérer les signatures et versions

1. **Configuration de l’application pour la release**
2. Génération de l’upload key et keystore
3. Signature de l’application avec l’upload key
4. Déploiement de la version de l’application

Configuration de l'application pour la release

Les opérations à mettre en place pour préparer la version Release de l'application :

- **Mettre de l'ordre dans votre application.**

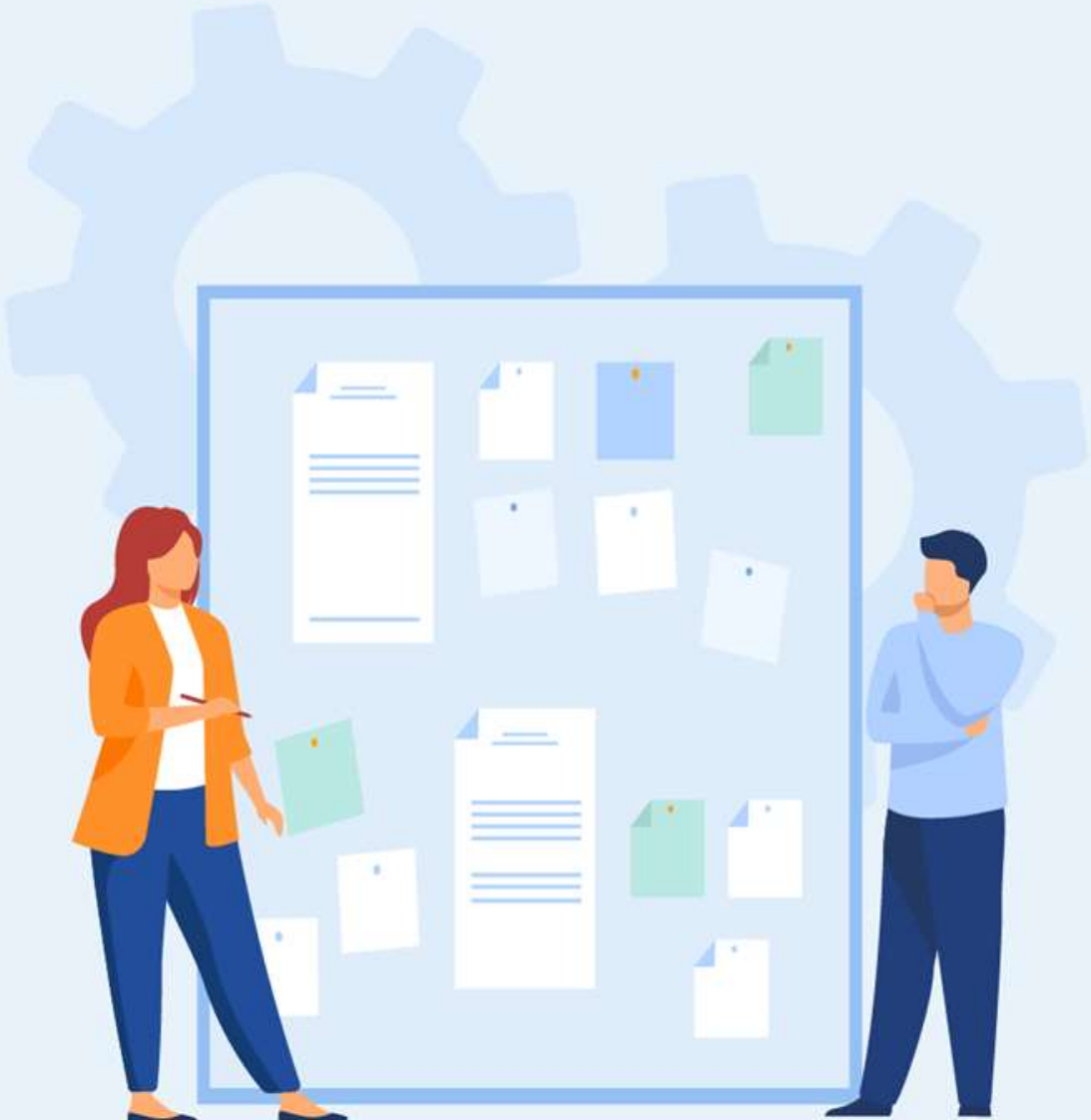
  Tests, nettoyage, version, nom, icône, licence, etc.
- **Signature de l'application.**

  Indispensable pour pouvoir la publier.
- **Teste de l'application comme un utilisateur final.**

  Se mettre à la place de l'utilisateur final
- **Publication de l'application.**

  Déployez votre application en ligne.

Copyright - Tout droit réservé - OFPPT



Chapitre 1

Gérer les signatures et versions

1. Configuration de l’application pour la release
2. **Génération de l’upload key et keystore**
3. Signature de l’application avec l’upload key
4. Déploiement de la version de l’application

Génération de l'upload key et keystore

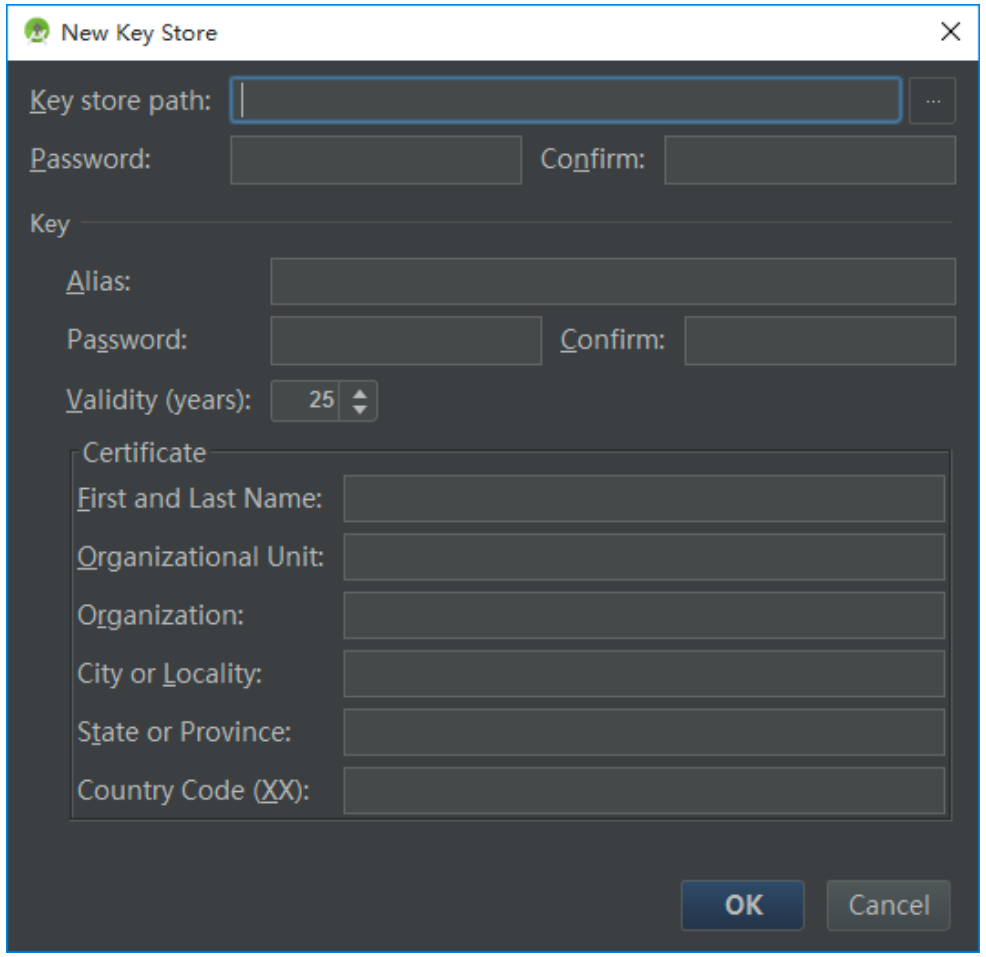
Il y a 2 méthodes pour générer l'upload key et keystore :

- **Générez une clé en utilisant Android Studio :**
  comme indiqué sur la capture d'écran

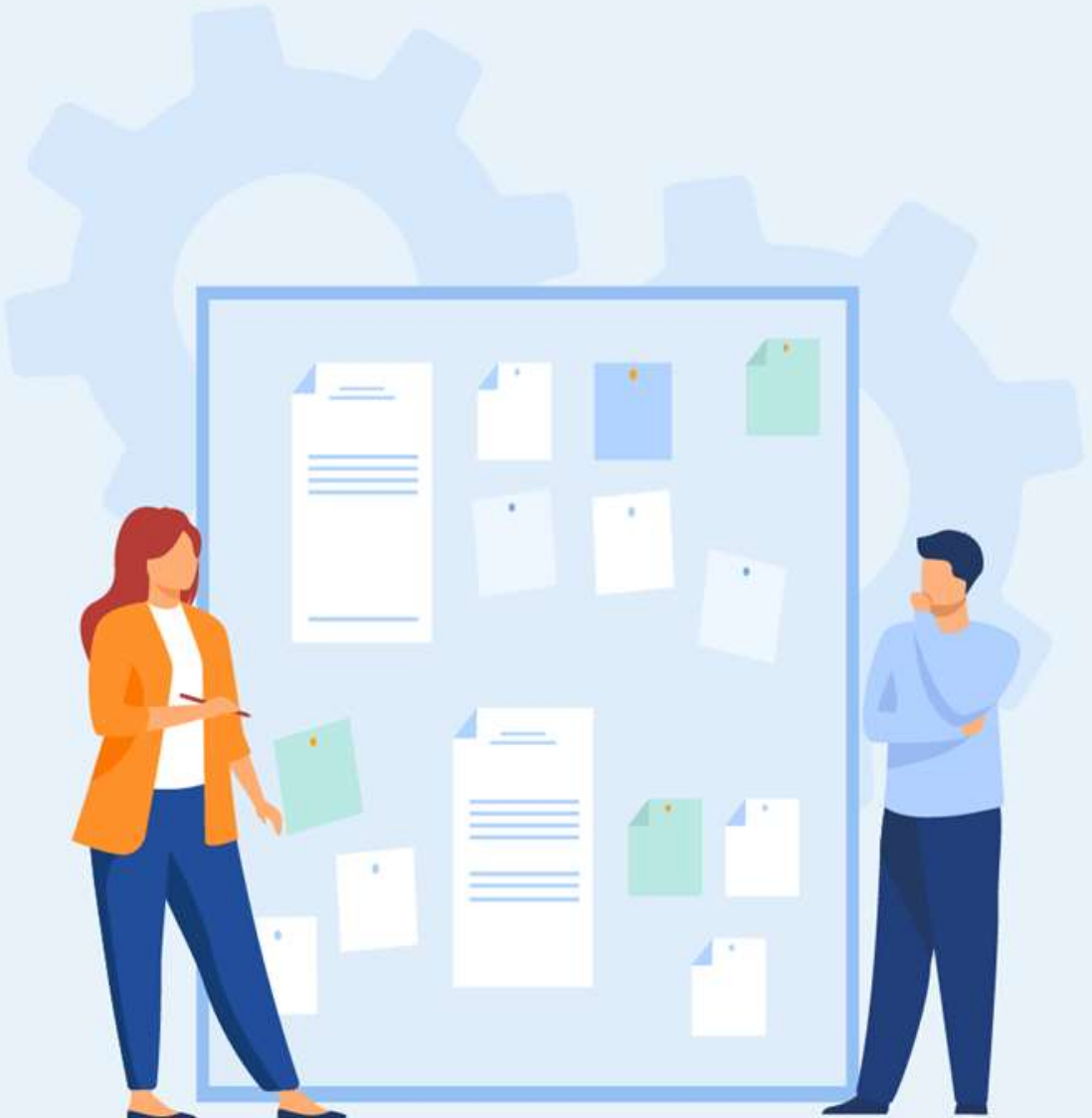
- **Générez une clé manuellement :**

keytool -genkey -v -keystore my-release-key.keystore -alias alias_name -keyalg RSA -keysize 2048 -validity 18250

« **my-release-key.keystore** » : votre fichier de clés.
« **validity** » : durée de validité de la clé. 50 ans ~ 18250 jours.



Copyright - Tout droit réservé - OFPPT



Chapitre 1

Gérer les signatures et versions

1. Configuration de l’application pour la release
2. Génération de l’upload key et keystore
3. **Signature de l’application avec l’upload key**
4. Déploiement de la version de l’application

Signature de l'application avec l'upload key

- **Signez l'application en utilisant Android Studio :**

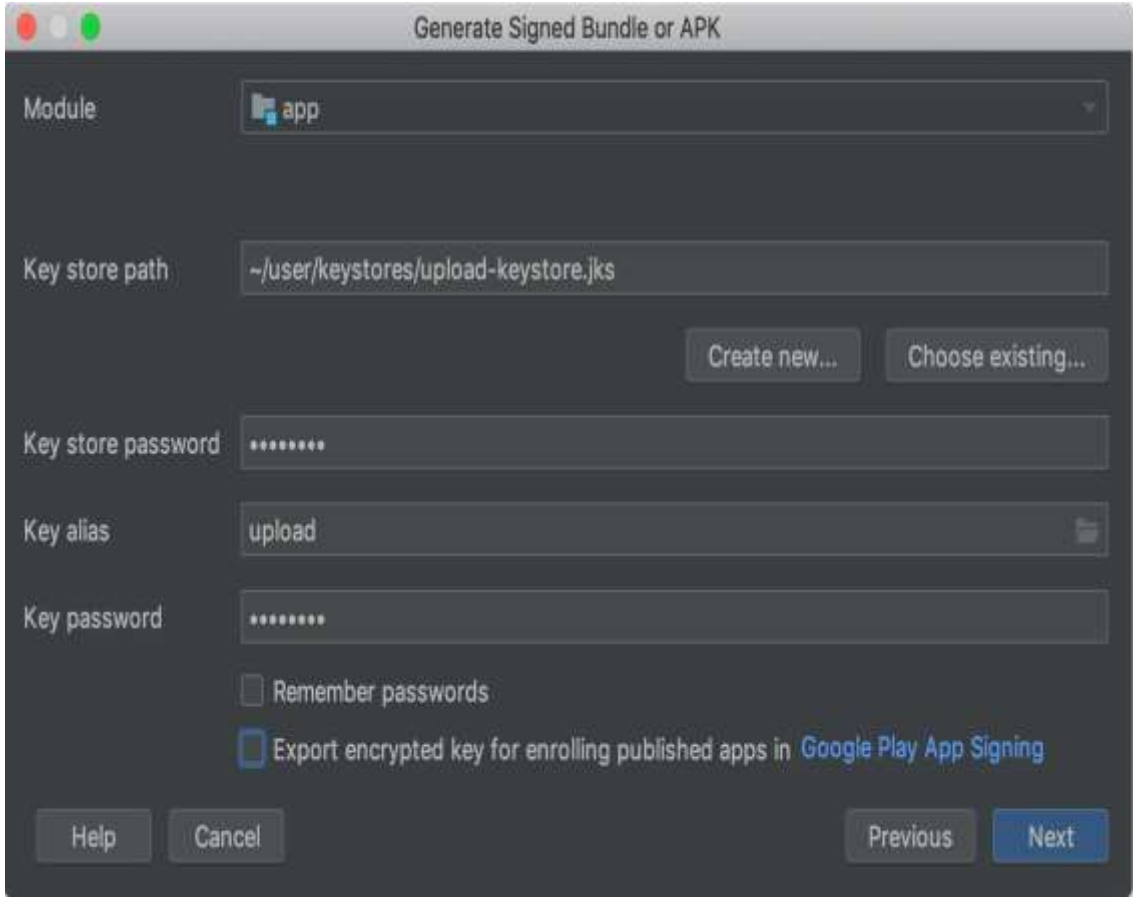
  https://developer.android.com/studio/publish/app-signing

  Dans le menu, cliquez sur « build »
  puis « **Generate signed Bundle/APK...** » ,
  cochez « **APK** », « **Next** »

- **Signez l'application manuellement :**

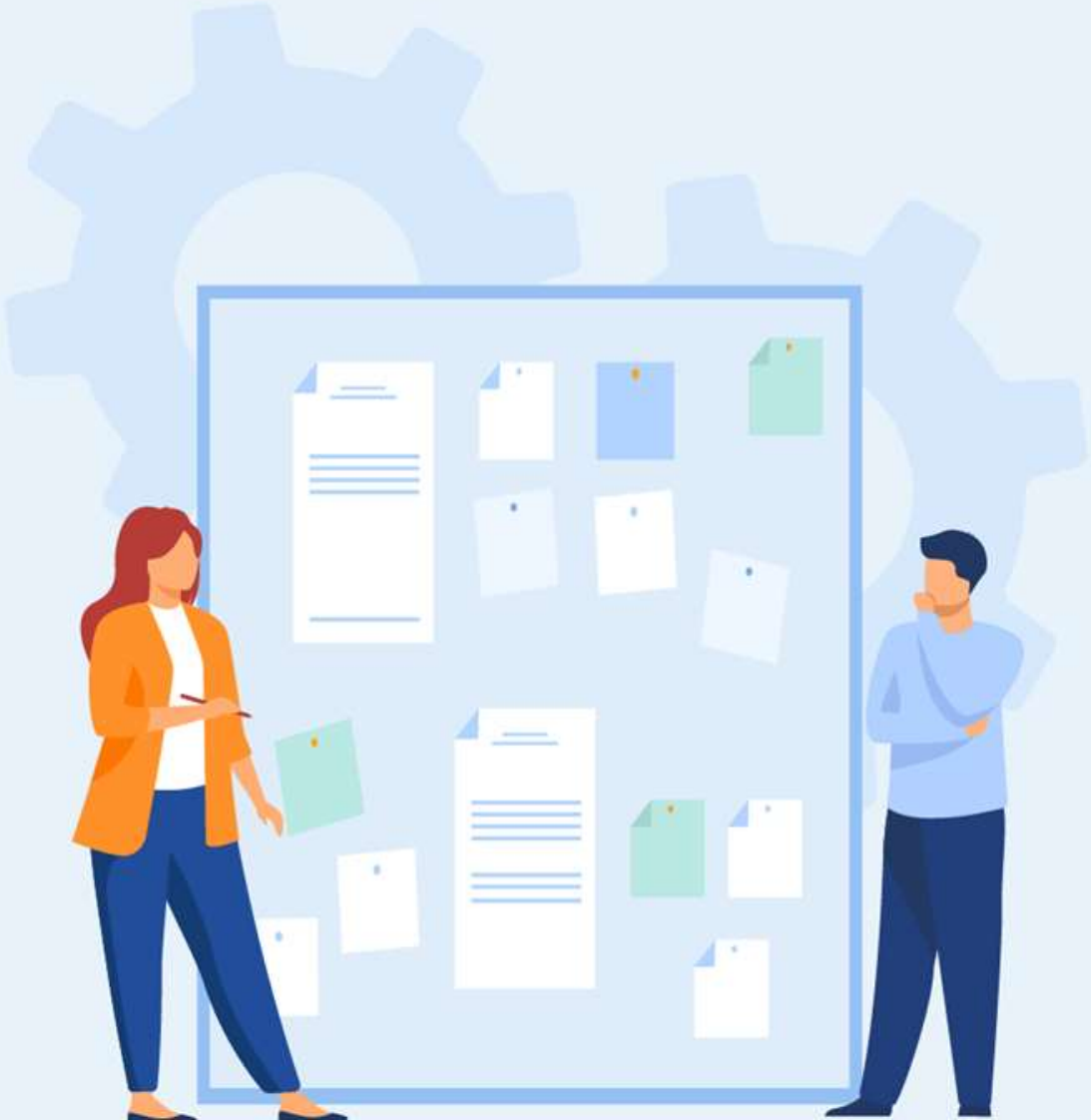
```
signingConfigs {
    releaseconfig {
        keyAlias 'upload'
        keyPassword 'password'
        storeFile file('upload-keystore.jks')
        storePassword 'password'
    }
}
```

Code à ajouter au fichier build.gradle



PARTIE 3

Copyright - Tout droit réservé - OFPPT



Chapitre 1

Gérer les signatures et versions

1. Configuration de l’application pour la release
2. Génération de l’upload key et keystore
3. Signature de l’application avec l’upload key
4. **Déploiement de la version de l’application**

Déploiement manuel de l'application

Vous pouvez donner accès à l'application via le web par exemple. Vous n'avez qu'à fournir le lien « URL » vers la page web en question.

```
<a href="monpaquetage.apk">Telecharger App</a>
```

- Vous devez activer dans votre appareil l'option « **installation à partir d'une source inconnue** ».
- Le serveur qui héberge la page doit ajouter ce « MIME » : « **application/vnd.android.package-archive** »

Copyright - Tout droit réservé - OFPPT

Déploiement automatique de l'application

Pour déployer l'application automatiquement, on utilise un outil de CI/CD :

- Une fois le développement d'une fonctionnalité est validé ou le bug est corrigé, le développeur pousse le projet par un outil de versionning (GIT,SVN…).

- L'outil de CI/CD détecte le changement des codes poussés, ensuite il démarre l'exécution de la méthode du déploiement :
  - Signer l'application
  - Générer l'APK release signé
  - Upload au Store.

Dans le développement d'applications mobiles, les outils CI/CD les plus utilisés sont :

Jenkins, **Bamboo** et **Travis CI, Bitrise, Fastlane, Circle CI…**

Copyright - Tout droit réservé - OFPPT



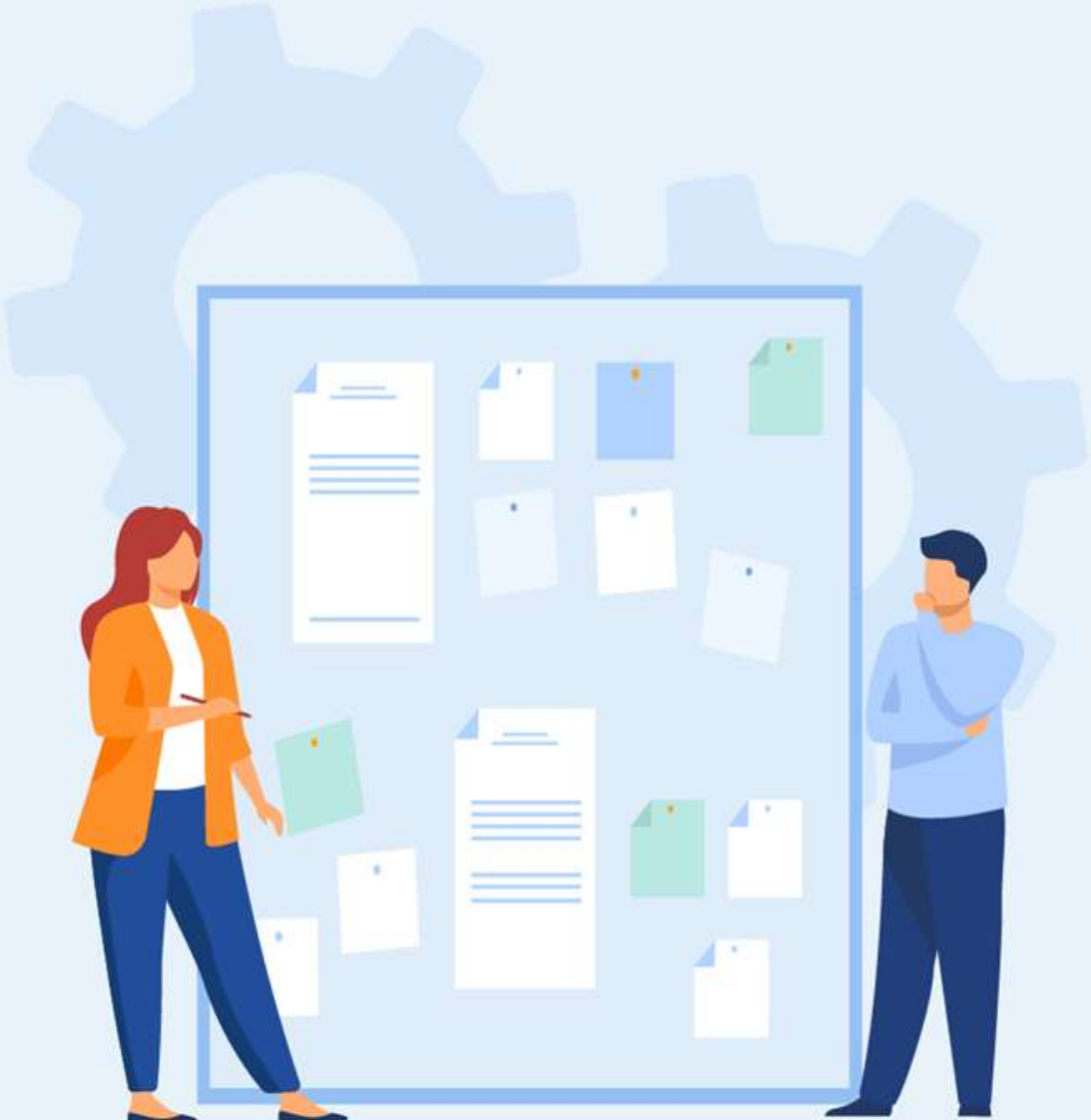
Chapitre 2

Utiliser des plateformes d’intégration et déploiements continus

Ce que vous allez apprendre dans ce chapitre :

- Définition des plateformes d’intégration et déploiements continus
- Utilisation de Gitlab CI
- Gestion de Pipeline
- Manipulation de Fastlane

04 heures



Chapitre 2

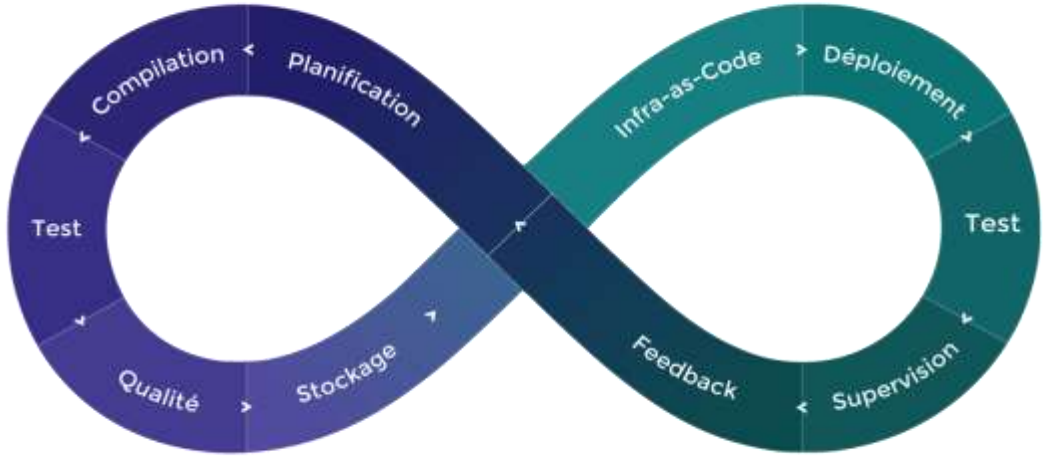
Utiliser des plateformes d’intégration et déploiements continus

1. **Définition des plateformes d’intégration et déploiements continus**
2. Utilisation de Gitlab CI
3. Gestion de Pipeline
4. Manipulation de Fastlane

Intégration continue

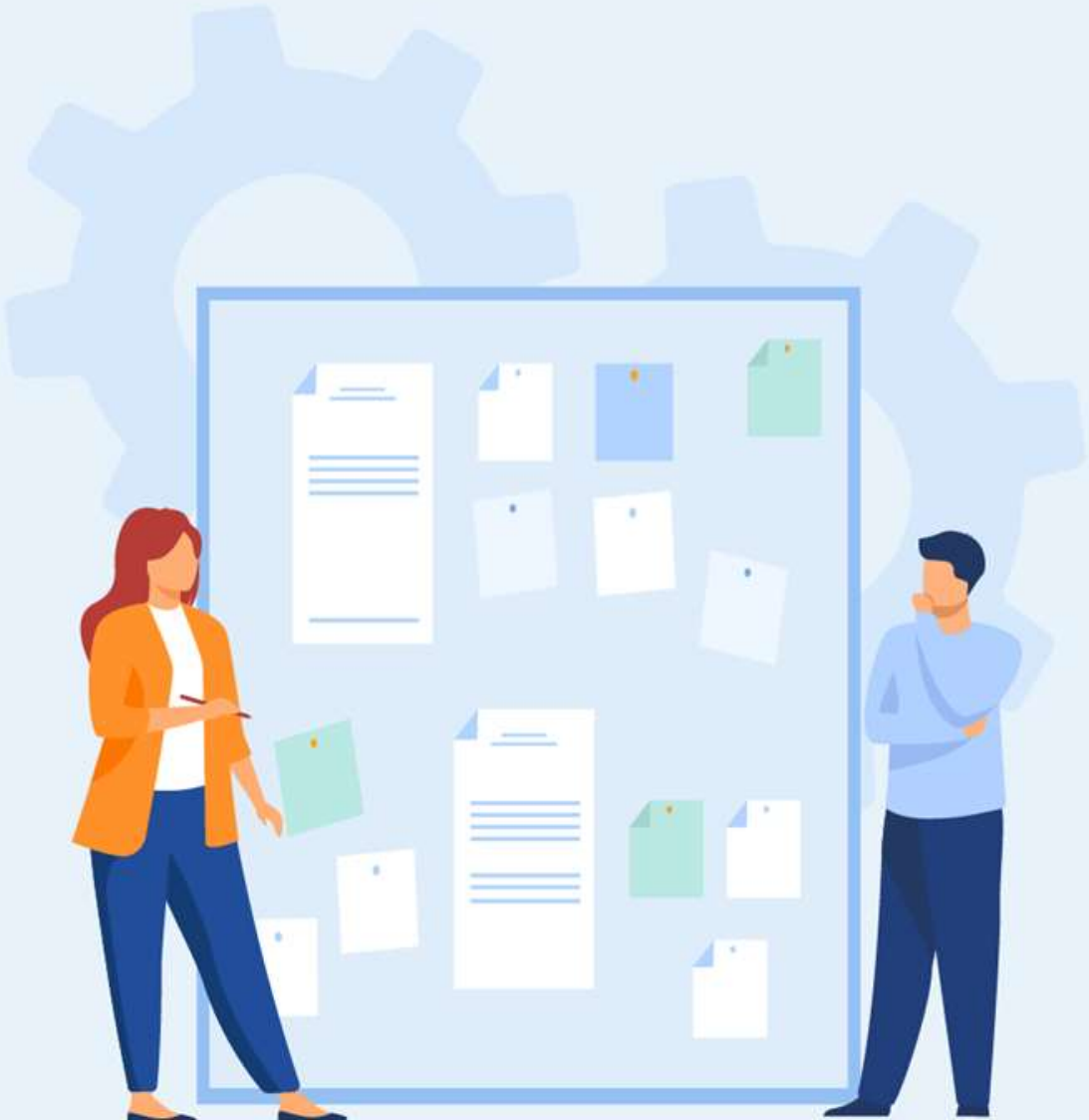
Le principe d'intégration continue consiste à dérouler le plan de tests après chaque modification du code source.

Les tests sont déroulés lorsque le développeur pousse les modifications vers le serveur de gestion de versions (par exemple **Git**). Ou alors lorsqu'il tente de fusionner les modifications dans la branche principale du projet. Si un seul test échoue, la fusion n'est pas possible. Cela permet d'éviter toute régression, et force le développeur à corriger son application.



source : https://openclassrooms.com/

Copyright - Tout droit réservé - OFPPT



Chapitre 2

Utiliser des plateformes d’intégration et déploiements continus

1. Définition des plateformes d’intégration et déploiements continus
2. **Utilisation de Gitlab CI**
3. Gestion de Pipeline
4. Manipulation de Fastlane

Utilisation de Gitlab CI

Pour utiliser GitLab, il faut créer un compte sur gitlab.com. A partir de là, on peut proposer des commits, commenter des projets et créer ses propres repository. Ensuite, il y a les classiques commits, merge request, les branches, etc.

Quand on crée un nouveau projet, on peut :

- **Créer** un projet vide pour stocker des fichiers, des roadmaps, du code, etc.
- **Créer** un projet via un template. En quelques clics, on crée un repository selon le type de langage ou de plateformes et GitLab vous mâche le travail de configuration
- **Importer un projet** : par exemple migrer son code de GitHub à GitLab
- **Exécuter** une chaîne de CI/CD pour un repository extérieur.



source : gitlab.com

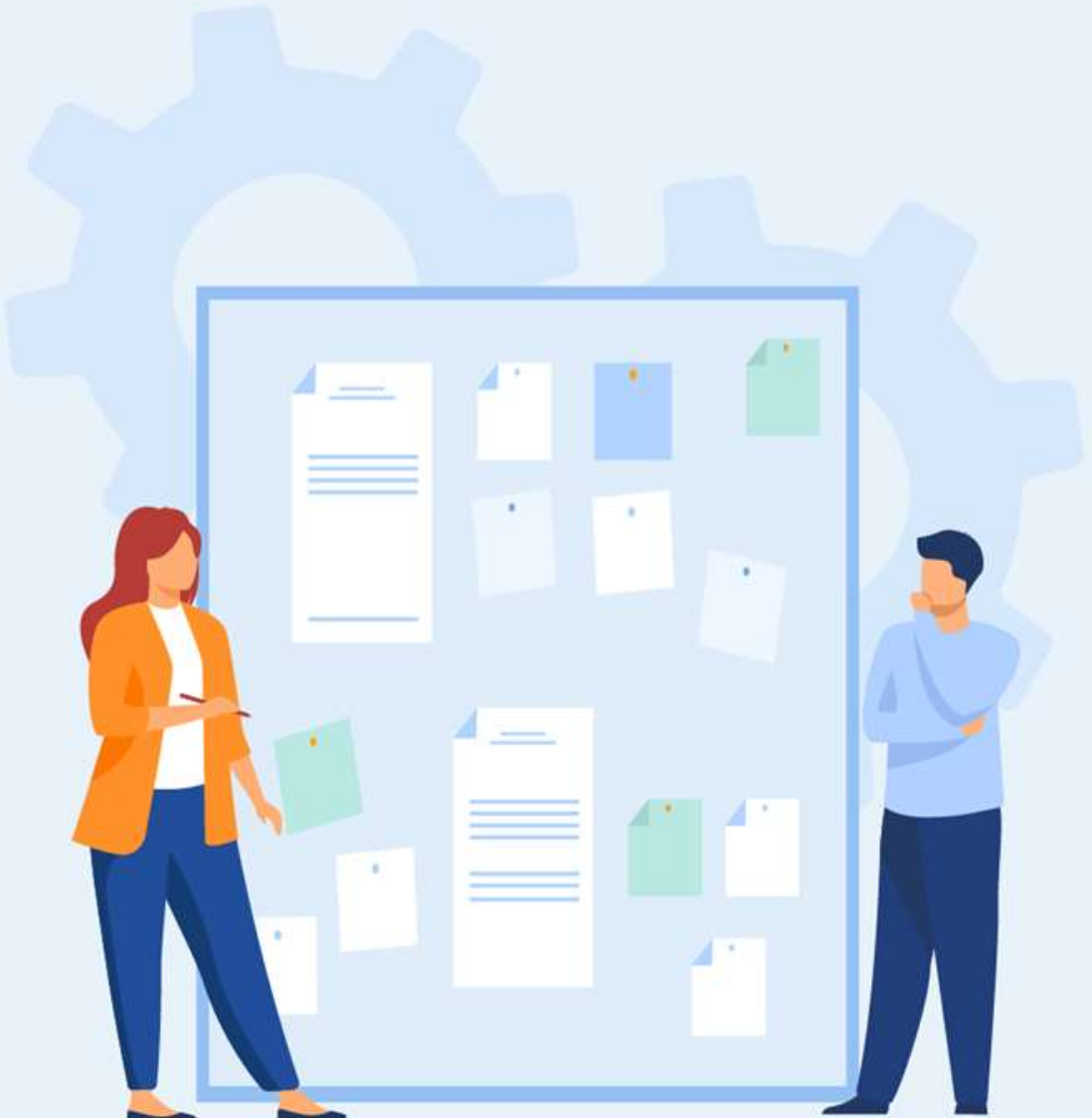
Le développeur peut utiliser GitLab directement dans l'IDE, via une interface graphique (typiquement un client desktop) ou en CLI, ou encore un mélange des trois.

Remarque

GIT : Système de contrôle de version pour:

- Mémoriser et retrouver différentes version d'un projet.
- Faciliter le travail collaboratif.

Copyright - Tout droit réservé - OFPPT



Chapitre 2

Utiliser des plateformes d’intégration et déploiements continus

1. Définition des plateformes d’intégration et déploiements continus
2. Utilisation de Gitlab CI
3. **Gestion de Pipeline**
4. Manipulation de Fastlane

Le pipeline de CI/CD

Un pipeline de CI/CD est une série d'étapes où le code passe pour aller du système de développement aux utilisateurs, en passant par des tests et des étapes intermédiaires.

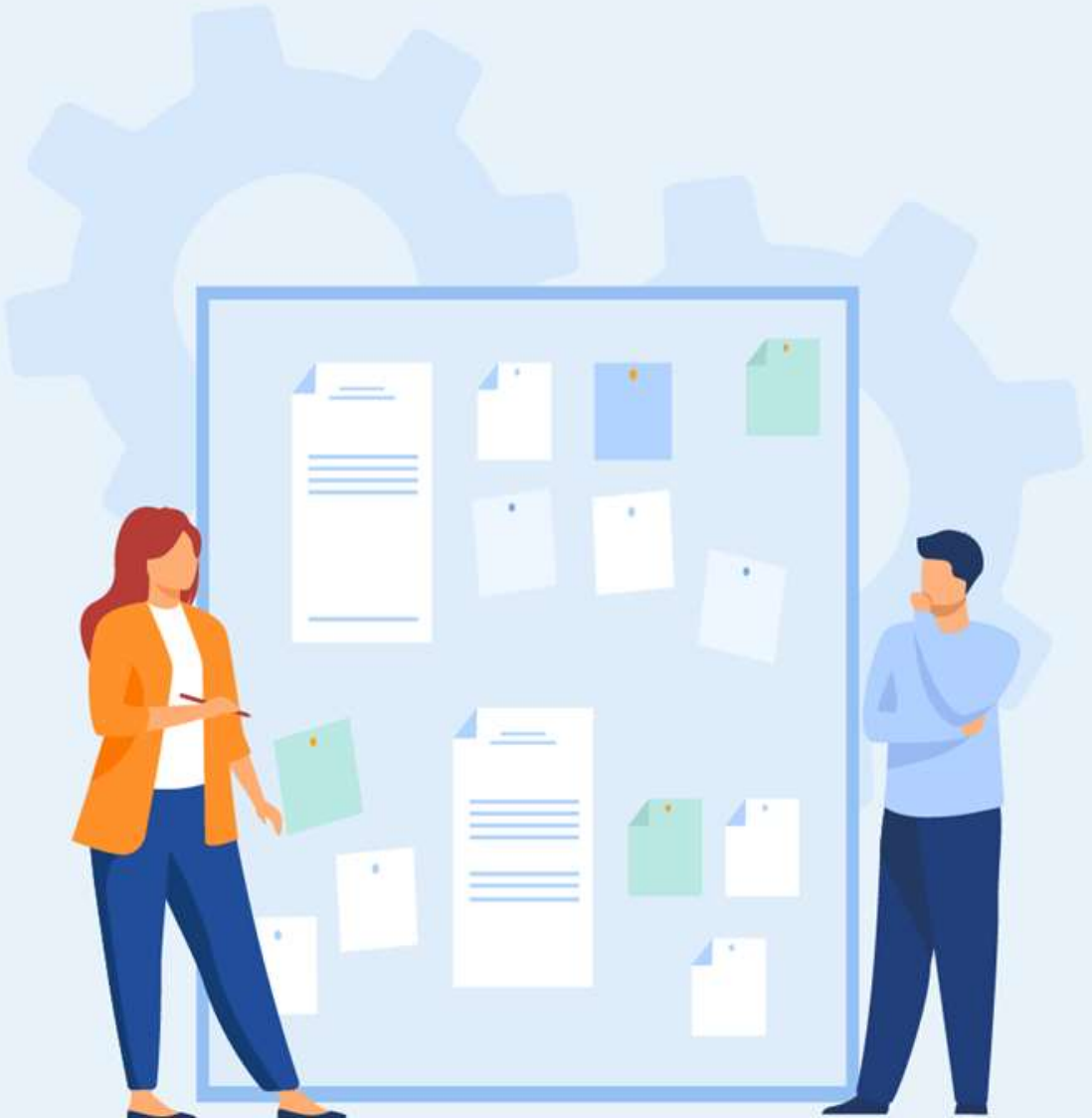
Voici les étapes de pipeline les plus courantes :

- **Création** : compilation de l'application.
- **Test** : test du code. L'automatisation permet ici d'épargner du temps et des efforts.
- **Lancement** : distribution de l'application au référentiel.
- **Déploiement** : déploiement du code en production.
- **Validation et conformité** : ces étapes de validation sont à adapter en fonction des besoins de votre entreprise. Des outils d'analyse de la sécurité des images tels que Clair peuvent garantir la qualité des images en les comparant à des vulnérabilités (CVE) connues.



source : redhat.com

Copyright - Tout droit réservé - OFPPT



Chapitre 2

Utiliser des plateformes d’intégration et déploiements continus

1. Définition des plateformes d’intégration et déploiements continus
2. Utilisation de Gitlab CI
3. Gestion de Pipeline
4. **Manipulation de Fastlane**

Fastlane

Fastlane est un outil open-source qui permet de faire du **Continuous Delivery** sous IOS et Android. Il permet d'automatiser un certain nombre de tâches fastidieuses comme gérer les screenshots, les certificats, déployer votre app…

https://fastlane.tools/

Voici la liste des libs à disposition avec Fastlane :

- Deliver
- Pem
- Produce
- Snapshot
- Sigh
- Gym
- Frameit
- Cert
- Scan



source : fastlane.com

Copyright - Tout droit réservé - OFPPT

Fastfile

Fastfile est un fichier **ruby** qui définit toutes vos **lanes**, on ajoute ce fichier dans le projet ensuite on le pousse, une fois l'outil CI/CD détecte sa présence, il l'exécute.

```
default_platform(:android)

platform :android do
  desc "Runs all the tests"
  lane :test do
    gradle(task: "test")
  end

  desc "Submit a new Beta Build to Crashlytics Beta"
  lane :beta do
    gradle(task: "clean assembleRelease")
    crashlytics

    # sh "your_script.sh"
    # You can also use other beta testing services here
  end

  desc "Deploy a new version to the Google Play"
  lane :deploy do
    gradle(task: "clean assembleRelease")
    upload_to_play_store
  end
end
```

Une **lane** est un ensemble d'instructions que vous souhaitez faire exécuter par Fastlane.

Ruby est un langage de programmation libre. Il est interprété, orienté objet et multi-paradigme.
Le langage a été standardisé au Japon en 2011

Copyright - Tout droit réservé - OFPPT