Les Application mobiles

Silem Abdelheq

1.0

Mars 2023

Table des matières

Objectifs

la finalité de cette matière est d'apporter à l'étudiant des connaissances en matière de développement d'application et système informatique dans des environnements mobiles. Avec l'arrivée des smartphones les applications mobiles sont omniprésents que l'on soit client (BtoC), fournisseur (BtoB) ou collaborateur (BtoE). Le but de ce cours est aussi d'apprendre la programmation sous Android, sa plate-forme de développement et les spécificités du développement embarqué sur smartphone[

I Chapitre 2 : Plateforme Android

La plateforme Android est l'un des systèmes d'exploitation mobiles les plus populaires au monde. Développée par Google, elle est utilisée sur un large éventail d'appareils, notamment des smartphones, des tablettes, des smartwatches et des téléviseurs. La plateforme est construite au-dessus du noyau Linux et est conçue pour être open source, ce qui permet aux développeurs de créer des applications et de personnaliser la plateforme en fonction de leurs besoins.

L'une des principales caractéristiques de la plateforme Android est le Google Play Store, qui est le principal canal de distribution des applications Android. Cela permet aux utilisateurs de découvrir et de télécharger facilement des applications, et aux développeurs d'atteindre un large public. Les applications Android sont généralement développées à l'aide du langage de programmation Java, bien que d'autres langages et frameworks puissent également être utilisés.

La plate-forme Android dispose d'une communauté de développeurs importante et active, avec de nombreuses ressources disponibles pour ceux qui souhaitent apprendre à développer des applications Android. Il s'agit notamment de la documentation officielle, de tutoriels et d'exemples de code. En outre, il existe de nombreuses bibliothèques et outils tiers disponibles pour aider à rationaliser le processus de développement.

Globalement, la plate-forme Android est une plate-forme puissante et flexible qui offre un large éventail de possibilités aux développeurs. Avec les bonnes compétences et les bons outils, les développeurs peuvent créer des applications de haute qualité qui touchent des millions d'utilisateurs dans le monde.

1. Objectives

L'objectif principal de ce chapitre est de fournir une vue d'ensemble de la plateforme Android, y compris son histoire, son architecture et ses composants clés. Les objectifs intermédiaires de ce chapitre peuvent être résumés comme suit :

- Définir ce qu'est la plateforme Android et en quoi elle diffère des autres systèmes d'exploitation mobiles.
- Décrire l'histoire d'Android, y compris son développement et ses principales étapes.
- Expliquer l'architecture du système d'exploitation Android, dont les différentes couches et composants qui constituent le système.
- Discuter des composants fondamentaux d'une application Android, notamment les activités, les services, les fournisseurs de contenu et les récepteurs de diffusion.

- Présenter le SDK Android, ses outils et ses ressources pour le développement d'applications Android.
- Démontrer comment installer et configurer les outils nécessaires au développement d'Android.
- Introduction à l'émulateur Android et comment créer un nouveau dispositif.
- Fournir des instructions étape par étape pour créer et exécuter une application Android basique.

2. Histoire d'Android

Comprendre l'histoire et l'évolution du système d'exploitation Android est important pour les développeurs, car cela les aide à comprendre le contexte et les motivations derrière les diverses décisions de conception, les contraintes techniques et les problèmes de compatibilité. En comprenant l'évolution d'Android OS, les développeurs peuvent mieux concevoir et optimiser leurs applications pour les versions actuelles et futures de la plate-forme.

Android a une histoire riche qui remonte au début des années 2000, lorsqu'une petite startup appelée Android Inc. a été fondée par Andy Rubin, Rich Miner, Nick Sears et Chris White. L'entreprise se concentrait sur le développement d'un système d'exploitation mobile basé sur le noyau Linux et capable d'alimenter des appareils photo numériques.

L'année 2005 est l'une des étapes importantes de l'histoire du système d'exploitation Android, lorsque Google a acquis Android Inc. et que le développement du système d'exploitation Android a continué sous l'égide de la société. Le premier appareil Android commercial, le HTC Dream, est sorti en 2008, fonctionnant avec la version 1.0 d'Android qui comprenait des fonctionnalités telles qu'un navigateur web, Google Maps et l'intégration de Gmail.

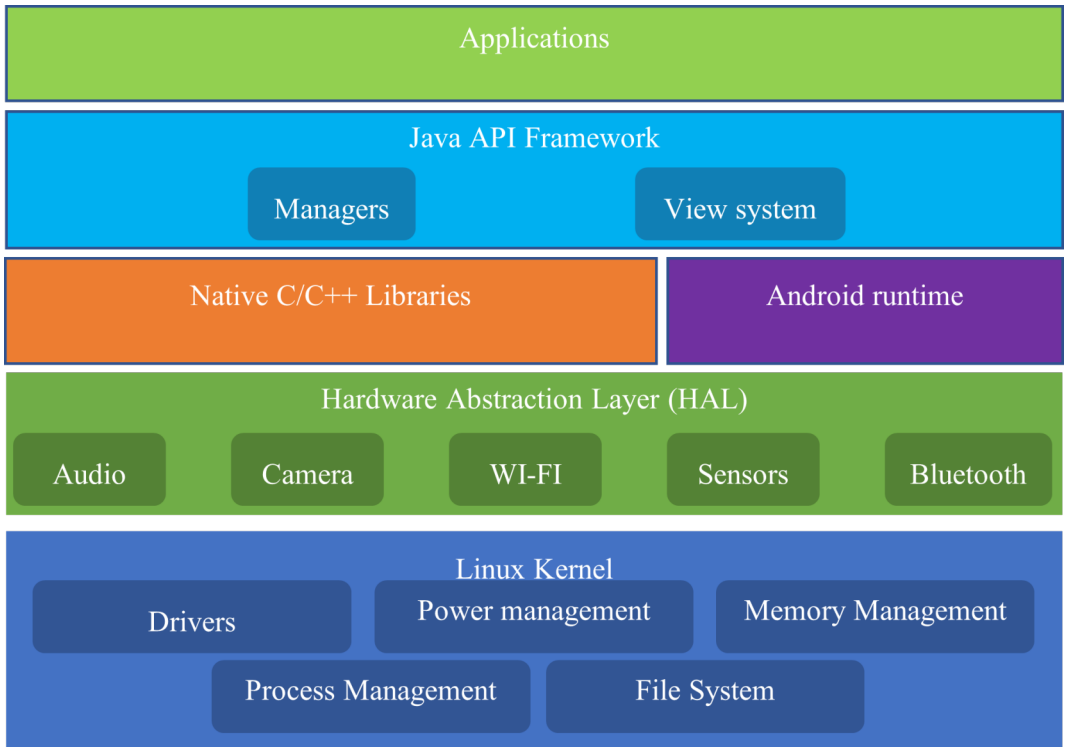
Au fil des ans, Google a publié de nombreuses versions d'Android, chacune avec son propre ensemble de fonctionnalités et d'améliorations. La convention de dénomination des versions d'Android a suivi un ordre alphabétique, chaque version étant nommée d'après un dessert.

Voici quelques exemples de l'évolution historique du système d'exploitation Android :

- Android 2.0 est sorti en 2009 et a introduit la prise en charge de tailles et de résolutions d'écran plus grandes, ainsi que de nouvelles fonctionnalités telles que la navigation virage par virage et la recherche vocale.
- En 2010, Android 2.2 a été publié, avec la prise en charge d'Adobe Flash et l'amélioration des performances.
- En 2011, Android 4.0, également connu sous le nom d'Ice Cream Sandwich, a été publié et a introduit une nouvelle interface utilisateur et des fonctionnalités telles que la reconnaissance faciale pour déverrouiller l'appareil.
- En 2013, Android 4.4, également connu sous le nom de KitKat, a été publié, avec des améliorations des performances et de nouvelles fonctionnalités telles que la prise en charge de l'impression à partir d'appareils mobiles.
- En 2014, Android 5.0, également connu sous le nom de Lollipop, a été publié, qui a introduit un nouveau langage de conception matérielle et des fonctionnalités telles que plusieurs comptes d'utilisateur et un mode ne pas déranger.
- En 2016, Android 7.0, également connu sous le nom de Nougat, a été publié, qui a introduit de nouvelles fonctionnalités telles que le multitâche à écran partagé et le support de la réalité virtuelle.
- En 2019, Android 10 a été publié, qui a introduit un mode sombre à l'échelle du système, une navigation gestuelle améliorée et des fonctions de confidentialité et de sécurité améliorées.

3. L'architecture du système d'exploitation Android

L'architecture du système d'exploitation Android est une architecture en couches conçue pour fournir une plate-forme souple et robuste pour l'exécution de dispositifs mobiles. L'architecture est divisée en cinq couches principales : la couche du noyau Linux, la couche d'abstraction matérielle (HAL), la couche du moteur d'exécution Android (ART), la couche du cadre d'application et la couche d'application.



Noyau Linux

Le noyau Linux fait partie essentielle du système d'exploitation Android. Il s'agit du composant central qui gère les ressources du système, telles que la mémoire, CPU et les périphériques d'entrée/sortie. Le noyau Linux fournit également l'interface nécessaire entre le matériel et les logiciels fonctionnant sur le système.

La plate-forme Android utilise une version modifiée du noyau Linux, qui a été optimisée pour les appareils mobiles. Le noyau est chargé de gérer les pilotes matériels de bas niveau et de fournir une plate-forme stable sur laquelle le reste du système Android peut fonctionner.

Le noyau Linux dans l'architecture Android comprend plusieurs composants clés :

- Gestion des processus : Le noyau Linux est responsable de la gestion des processus exécutés sur le système Android. Il alloue des ressources aux différents processus et s'assure qu'ils s'exécutent de manière fluide et sans interférence les uns avec les autres.
- Gestion de la mémoire : Le noyau gère l'allocation et la désallocation de la mémoire dans le système. Il garde la trace de la mémoire utilisée par chaque processus et veille à ce que le système ne soit pas à court de mémoire.
- Pilotes de périphériques : Le noyau Linux fournit les pilotes nécessaires aux différents composants matériels du système, notamment l'écran, l'écran tactile, la caméra et les capteurs.
- Système de fichiers : Le noyau gère également le système de fichiers de l'appareil Android, en veillant à ce que les fichiers soient enregistrés et récupérés correctement.

- Sécurité : Le noyau Linux fournit un cadre de sécurité robuste pour le système Android, le protégeant ainsi des menaces potentielles pour la sécurité.

Dans l'ensemble, le noyau Linux de l'architecture Android joue un rôle essentiel pour assurer la stabilité et la sécurité du système. Sa capacité à gérer efficacement les composants matériels et les ressources en fait un élément essentiel du développement des appareils mobiles.

La couche d'abstraction matérielle (HAL)

La couche d'abstraction matérielle (HAL) est un composant clé de l'architecture du système d'exploitation Android. Elle se situe entre le matériel et les couches logicielles de niveau supérieur, telles que le moteur d'exécution Android et le cadre d'application, et fournit une interface standard leur permettant de communiquer avec le matériel.

Le HAL abstrait les détails matériels des couches logicielles de plus haut niveau en fournissant un ensemble normalisé d'API qui décrivent les capacités et le comportement du matériel. Cela permet aux fabricants de périphériques de créer des implémentations HAL spécifiques au matériel qui peuvent être utilisées avec la pile logicielle Android standard, ce qui permet aux développeurs d'écrire des logiciels agnostiques au matériel qui peuvent fonctionner sur une large gamme de périphériques.

Le HAL est organisé en modules qui correspondent à différentes catégories de matériel, comme la caméra, l'audio ou le Wi-Fi. Chaque module contient un ensemble de fonctions spécifiques au matériel que les couches logicielles de niveau supérieur peuvent appeler pour accéder au matériel. Par exemple, le module caméra peut contenir des fonctions de capture et de traitement des images, tandis que le module audio peut contenir des fonctions d'enregistrement et de lecture audio.

Lorsqu'une application Android fait une demande liée au matériel, comme prendre une photo ou jouer un son, la demande est acheminée par le module HAL correspondant vers le composant matériel approprié. Le module HAL gère les détails de l'interaction avec le matériel et renvoie les résultats aux couches logicielles de niveau supérieur.

Dans l'ensemble, le HAL est un élément important de l'architecture Android, car il permet une interaction cohérente et normalisée avec le matériel sur différents appareils, ce qui permet aux développeurs de créer plus facilement des applications fonctionnant sur un large éventail d'appareils.

Bibliothèques C/C++

Dans l'architecture Android, une bibliothèque C/C++ native est une collection de fonctions et de ressources écrites dans les langages de programmation C ou C++ qui sont compilées et empaquetées sous forme de fichier de bibliothèque partagée (.so). Ces bibliothèques sont utilisées pour mettre en œuvre les fonctionnalités critiques du système Android et fournissent une interface entre le cadre Android et les composants matériels sous-jacents.

Les bibliothèques C/C++ natives font partie de l'Android Runtime (ART) et sont incluses dans le système d'exploitation Android en tant que partie du microprogramme. Ces bibliothèques comprennent un ensemble de bibliothèques préconstruites fournies par le projet Android Open Source, ainsi que des bibliothèques tierces qui peuvent être ajoutées au système par les développeurs.

Les bibliothèques C/C++ natives sont utilisées pour effectuer des tâches qui nécessitent un accès de bas niveau aux ressources du système, comme le rendu graphique, le traitement audio et le cryptage. En utilisant du code natif, les développeurs peuvent obtenir de meilleures performances et tirer parti des capacités du matériel sous-jacent.

Le NDK (Native Development Kit) d'Android est un ensemble d'outils qui permet aux développeurs d'écrire du code C /C++ natif pour Android et de créer des bibliothèques partagées qui peuvent être utilisées dans les applications Android. Le NDK comprend un ensemble d'en-têtes et de bibliothèques, une chaîne d'outils de compilation croisée et d'autres outils pour aider les développeurs à construire, déboguer et empaqueter du code natif pour Android.

Pour utiliser les bibliothèques C/C++ natives dans une application Android, les développeurs doivent d'abord créer une couche JNI (Java Native Interface) qui fournit un pont entre le code Java et le code natif. La couche JNI permet au code Java d'appeler les fonctions du code natif et fournit un moyen de faire passer les données entre les deux.

En résumé, les bibliothèques natives C/C++ sont une partie essentielle de l'architecture Android, fournissant un moyen d'accéder aux ressources système de bas niveau et d'obtenir de meilleures performances dans les applications Android. En utilisant le NDK Android, les développeurs peuvent créer des bibliothèques de code natif qui peuvent être utilisées dans les applications Android, et la JNI fournit un pont entre le code Java et le code natif.

Voici quelque bibliothèques natives C/C++ :

- OpenSSL : OpenSSL est une bibliothèque de cryptographie qui fournit divers protocoles, algorithmes et outils pour les communications sécurisées, tels que SSL et TLS.
- zlib : zlib est une bibliothèque de compression qui fournit divers algorithmes et outils pour la compression et la décompression de données, tels que gzip et zip.
- SQLite : SQLite est une bibliothèque de base de données qui fournit divers outils pour créer et gérer des bases de données SQL légères, telles que celles utilisées dans les applications mobiles.
- GTK+ : GTK+ est une bibliothèque d'interface utilisateur graphique qui fournit divers outils pour créer et gérer des interfaces utilisateur graphiques, comme des boutons, des menus et des fenêtres.
- SDL : SDL est une bibliothèque multimédia multiplateforme qui fournit divers outils pour la création et la gestion de jeux et d'applications multimédia, tels que les graphiques, le son et la gestion des entrées.

Runtime Android

En résumé, le Runtime d'Android est un composant clé du système d'exploitation Android qui est responsable de l'exécution du bytecode Java et comprend un certain nombre de fonctionnalités qui contribuent à améliorer les performances et à optimiser l'utilisation de la mémoire. Il prend également en charge les bibliothèques natives, ce qui permet aux développeurs d'inclure du code C/C++ dans leurs applications pour améliorer encore les performances.

ART est le successeur de Dalvik, le moteur d'exécution original utilisé par Android, et a été introduit pour la première fois dans Android 4.4 KitKat. La différence la plus significative entre ART et Dalvik est la manière dont ils exécutent le code. Dalvik utilise la compilation Just-In-Time (JIT), ce qui signifie que le bytecode est compilé en code machine au moment de l'exécution de l'application. ART, en revanche, utilise la compilation Ahead-Of-Time (AOT), ce qui signifie que le bytecode est compilé en code machine avant l'exécution de l'application. Cela permet à ART d'offrir de meilleures performances et une utilisation plus efficace de la mémoire.

ART comprend plusieurs fonctionnalités supplémentaires, telles que l'amélioration du garbage collection, la prise en charge de la fragmentation de la mémoire. Il comprend également de nouveaux outils de débogage et de profilage qui permettent aux développeurs d'optimiser plus facilement les performances de leurs applications.

Une autre caractéristique clé d'ART est la prise en charge des bibliothèques natives. Cela permet aux développeurs d'inclure du code C/C++ natif dans leurs applications Android, ce qui peut offrir de meilleures performances pour certaines tâches, comme le traitement des images ou du son. ART comprend également un compilateur juste-à-temps (JIT) pour le code natif, ce qui améliore encore les performances.

Cadre de l'API Java (JAVA API Framework)

Le cadre de l'API Java est un composant crucial de l'architecture Android qui fournit les outils nécessaires à la création d'applications Android basées sur Java. Le cadre API Java se compose d'une vaste collection de classes et de bibliothèques Java qui permettent aux développeurs de créer facilement des applications Android robustes et sophistiquées.

Le cadre de l'API Java fournit un ensemble de classes et d'interfaces Java préconstruites que les développeurs peuvent utiliser pour créer leurs applications. Ces classes et interfaces comprennent des composants fondamentaux tels que des activités, des services, des récepteurs de diffusion, des fournisseurs de contenu, etc. Le cadre de l'API Java comprend également des paquets pour les tâches courantes telles que la mise en réseau, l'entrée/sortie de fichiers, les graphiques, le multimédia, l'interface utilisateur, et bien d'autres encore.

L'une des caractéristiques essentielles du framework API Java est sa capacité à fournir un haut niveau d'abstraction, ce qui simplifie le processus de développement. Le cadre abstrait les détails de bas niveau du système Android, tels que le matériel des appareils, la gestion de la mémoire et les systèmes de fichiers, de sorte que les développeurs peuvent se concentrer sur la création des fonctions essentielles de leurs applications. En outre, le cadre de l'API Java fournit une interface cohérente et uniforme entre les différentes versions du système d'exploitation Android, ce qui facilite le développement d'applications fonctionnant sur un large éventail de dispositifs.

Application

La couche d'application est la couche la plus élevée de l'architecture Android, où l'utilisateur interagit avec l'appareil. Cette couche contient toutes les applications installées par l'utilisateur, qu'il peut télécharger depuis le Google Play Store ou d'autres sources. Les applications sont écrites en Java et conçues pour fonctionner sur la plate-forme Android.

Chaque application s'exécute au sein de son propre processus et dispose de sa propre instance de machine virtuelle, qui fournit un environnement "sandbox" pour l'exécution de l'application. Cette approche garantit que si une application tombe en panne, cela n'affectera pas la stabilité globale du système et les autres applications pourront continuer à fonctionner.

La couche applicative comprend un ensemble d'applications de base préinstallées avec le système Android, telles que l'application téléphone, l'application messagerie, l'application appareil photo, etc. Ces applications sont conçues pour fournir les fonctionnalités de base de l'appareil et peuvent être mises à jour ou remplacées par l'utilisateur ou le fabricant de l'appareil.

La couche applicative comprend également des applications de niveau système qui fournissent des services aux autres applications et au système, comme l'interface utilisateur du système, les paramètres et l'installateur de paquets. Ces applications font partie du système Android et ne peuvent être mises à jour que par des mises à jour du système.

Les développeurs peuvent créer leurs propres applications en utilisant le kit de développement logiciel (SDK) Android et le langage de programmation Java. Ils peuvent accéder à divers API et services fournis par la plate-forme Android pour créer des applications riches et attrayantes.

4. Les composants fondamentaux d'une application Android

Dans le développement Android, une application est constituée de différents composants qui remplissent des fonctions spécifiques. Ces composants peuvent être divisés en deux groupes : les composants principaux, tels que les activités, les services, les récepteurs de diffusion et les fournisseurs de contenu, et les composants secondaires, notamment les fragments, les intents, les widgets d'application et les notifications.

Activités

Dans Android, une activité est un composant fondamental d'une application qui représente un écran unique avec une interface utilisateur. Les activités sont utilisées pour construire les parties interactives d'une application que les utilisateurs peuvent voir et avec lesquelles ils peuvent interagir. Chaque activité correspond généralement à un écran de l'application, tel qu'un écran de connexion, un écran de paramètres ou un écran principal. Les activités sont conçues pour être modulaires et réutilisables, de sorte qu'elles peuvent être combinées de différentes manières pour créer des mises en page et des expériences utilisateur différentes.

Une activité est une unité autonome qui fournit une interface utilisateur et peut contenir des éléments tels que des boutons, des zones de texte et des images. Elle peut également être utilisée pour lancer d'autres activités ou pour recevoir des données de l'utilisateur. Lorsqu'une activité est lancée, elle est ajoutée à la pile d'activités, qui est une structure de données LIFO (last-in-first-out) qui garde la trace de toutes les activités en cours d'exécution. Lorsque l'utilisateur appuie sur le bouton retour, l'activité en cours est retirée de la pile et l'activité précédente s'affiche.

Les activités ont un cycle de vie qui se compose de différents états, tels que l'exécution, la pause ou l'arrêt. Le cycle de vie est géré par le système Android, qui appelle différentes méthodes dans le code de l'activité pour signaler les transitions entre les différents états.

Services

Dans une application Android, un service est un composant qui peut fonctionner en arrière-plan, même lorsque l'application n'est pas activement utilisée. Les services sont utilisés pour les tâches qui doivent être exécutées en permanence en arrière-plan, comme la lecture de musique ou le téléchargement de données. Les services s'exécutent sur le thread principal de l'application, mais ils peuvent également s'exécuter dans un thread distinct pour éviter de bloquer l'interface utilisateur.

Il existe deux types de services dans Android : les services de premier plan et les services d'arrière-plan. Les services d'avant-plan ont une notification visible que l'utilisateur peut voir dans la zone de notification. Ils sont généralement utilisés pour les tâches qui requièrent l'attention de l'utilisateur, comme la lecture de musique ou l'enregistrement audio. Les services d'arrière-plan n'ont pas de notification visible et sont utilisés pour des tâches qui ne nécessitent pas l'attention de l'utilisateur, comme le téléchargement de données ou le traitement de données en arrière-plan.

Les services peuvent être lancés et arrêtés par d'autres composants de l'application, tels que les activités ou les récepteurs de diffusion. Ils peuvent également être liés à d'autres composants pour interagir avec eux et partager des

données. Les services peuvent être lancés comme des tâches à long terme ou ponctuelles, et ils peuvent être arrêtés à tout moment par le système ou l'utilisateur.

les récepteurs de diffusion (Broadcast receivers)

Les Broadcast receivers sont un composant fondamental des applications Android, utilisés pour recevoir et répondre aux messages de diffusion du système. Un message de diffusion est un message qui peut être envoyé par le système ou par d'autres applications, notifiant l'application d'événements système, tels que l'avertissement de batterie faible, les changements de connexion réseau ou les appels téléphoniques entrants. Un récepteur de diffusion est un composant qui peut écouter ces messages de diffusion, et peut être utilisé pour exécuter du code en réponse à des événements système spécifiques.

Les récepteurs de diffusion sont généralement mis en œuvre en tant que sous-classe de la classe des récepteurs de diffusion et doivent être enregistrés dans le fichier manifeste de l'application. Un récepteur de diffusion peut être enregistré pour recevoir une diffusion spécifique en spécifiant un filtre d'intention qui précise le type de message de diffusion qui l'intéresse. Lorsque le message de diffusion est reçu, le système envoie le message à tous les récepteurs enregistrés qui correspondent au filtre d'intention.

Les récepteurs de diffusion peuvent être utilisés à diverses fins, comme le démarrage ou l'arrêt d'un service, la mise à jour de l'interface utilisateur ou la notification d'un événement spécifique à l'utilisateur. Par exemple, une application météo peut enregistrer un récepteur de diffusion pour recevoir le message de diffusion ACTION_BOOT_COMPLETED, ce qui permet à l'application de mettre à jour les données météorologiques au démarrage de l'appareil. De même, une application de lecture de musique peut enregistrer un récepteur de diffusion pour recevoir le message de diffusion ACTION_HEADSET_PLUG, permettant à l'application de lancer ou d'arrêter la lecture de musique lorsqu'un casque est branché ou débranché.

Fournisseurs de contenu (content providers)

Dans Android, un fournisseur de contenu est un composant qui gère l'accès à un ensemble structuré de données. Il permet à une application de partager des données avec d'autres applications, et fournit également un moyen pour les applications d'interroger ou de modifier les données stockées par d'autres applications. Les fournisseurs de contenu sont utiles pour gérer les données persistantes qui doivent être partagées entre différentes applications ou auxquelles on doit accéder depuis différentes parties d'une application.

L'objectif principal d'un fournisseur de contenu est de fournir une interface standardisée pour accéder à des données provenant de différentes sources, telles qu'une base de données locale, un serveur distant ou un système de fichiers. Les fournisseurs de contenu offrent généralement un ensemble de méthodes d'interrogation qui permettent à d'autres applications de lire et d'écrire des données. Par exemple, une application qui souhaite afficher une liste de contacts peut utiliser le fournisseur de contenu de l'application Contacts pour récupérer les données de contact.

Les fournisseurs de contenu sont également utiles pour renforcer la sécurité des données et le contrôle d'accès. Ils peuvent restreindre l'accès à certaines données en fonction des autorisations, garantissant ainsi que seules les applications autorisées peuvent accéder aux données sensibles.

5. Le SDK Android

Le SDK Android (Software Development Kit) est un ensemble d'outils logiciels et de bibliothèques fournis par Google pour permettre aux développeurs de créer et de tester des applications Android. Le SDK Android comprend un ensemble complet d'outils de développement, tels qu'un débogueur, des bibliothèques, un émulateur, de la documentation et des exemples de code.

Le SDK Android fournit les outils et les ressources nécessaires pour développer, déboguer et tester les applications Android. Cela comprend un ensemble complet d'outils de développement, comme Android Studio, qui est l'environnement de développement intégré (IDE) officiel pour le développement d'applications Android. Android Studio offre des fonctionnalités telles que la complétion de code, le débogage et des outils d'analyse des performances, entre autres.

Outre les outils de développement, le SDK Android comprend un ensemble de bibliothèques et d'API spécifiques à la plate-forme qui permettent aux développeurs de créer une gamme d'applications, allant des jeux et des outils de productivité aux applications de communication et de médias sociaux. Ces bibliothèques et API permettent d'accéder à diverses fonctionnalités d'Android, telles que l'appareil photo, les capteurs, la lecture multimédia et la connectivité réseau.

Le SDK Android comprend également un émulateur qui permet aux développeurs de tester leurs applications sur des appareils virtuels qui émulent le comportement des appareils réels. Cela permet aux développeurs de tester leurs applications sur une gamme de dispositifs et de configurations, sans avoir besoin de dispositifs physiques.

6. Installation et configuration des outils

L'installation et la configuration des outils de développement est très simple et il suffit d'installer Android studio qui installera automatiquement tous les outils (sauf java) (Voir le vidéo d'installation dans *moodle*).

Etape 1 : Téléchargez la dernière version d'Android Studio depuis le site officiel.

- Allez sur le site officiel d'Android Studio « https://developer.android.com/studio ».
- Cliquez sur le bouton "Download" pour télécharger le programme d'installation.
- Choisissez la version pour Windows si nécessaire.

Etape 2 : Exécutez le programme d'installation téléchargé.

- Ouvrez le fichier d'installation téléchargé (par exemple, android-studio-2023.1.1.0.exe).
- Si vous avez des problèmes avec le téléchargement, vous pouvez vérifier l'intégrité du fichier en utilisant son hash MD5.

Etape 3 : Si vous êtes invité à choisir le type d'installation, sélectionnez "Standard" et cliquez sur "Suivant".

- Cette option installera Android Studio avec les composants recommandés.

Etape 4 : Choisissez les composants à installer et cliquez sur "Suivant".

- Vous pouvez cocher/décocher les composants que vous voulez installer ou pas.

Etape 5 : Choisissez l'emplacement d'installation et cliquez sur "Suivant".

- Vous pouvez modifier le chemin d'installation si vous le souhaitez.

Etape 6 : Sur l'écran "Configuration des composants SDK", sélectionnez "Recommandé" et cliquez sur "Suivant".

- Cette option installera les composants SDK recommandés.

Etape 7 : Sur l'écran "Vérification des paramètres", cliquez sur "Installer".

Etape 8 : Android Studio commencera l'installation des composants sélectionnés. Attendez que l'installation se termine.

- Cette étape peut prendre un certain temps, en fonction de la vitesse de votre ordinateur, la vitesse de votre connexion et de la quantité de composants à installer.
- Une fois l'installation terminée, cliquez sur "Suivant", puis sur "Terminer".
- Android Studio est maintenant installé sur votre ordinateur.

Vous pouvez rencontrer une erreur qui dit "Aucune installation JVM trouvée. Veuillez installer un JDK 64 bits. Si vous avez déjà installé un JDK, définissez une variable JAVA_HOME".

Pour résoudre cette erreur, vous devez installer un JDK (Java Development Kit) et définir la variable d'environnement JAVA_HOME. Vous pouvez suivre ces étapes pour ce faire :

1. Allez sur la page de téléchargement Java SE officielle (https://www.oracle.com/java/technologies/javase-downloads.html).
2. Téléchargez le JDK pour Windows, en veillant à sélectionner la version appropriée (32 bits ou 64 bits) en fonction de votre système.
3. Exécutez l'installateur téléchargé et suivez les invites pour terminer l'installation.
4. Une fois l'installation terminée, ouvrez le Panneau de configuration Windows, allez dans "Système et sécurité" > "Système" > "Paramètres système avancés".
5. Dans la fenêtre Propriétés système, cliquez sur le bouton "Variables d'environnement".
6. Dans la section "Variables System", cliquez sur "Nouveau" pour créer une nouvelle variable d'environnement.
7. Entrez "JAVA_HOME" comme nom de variable et le chemin d'accès au répertoire d'installation du JDK comme valeur de variable. Par exemple, si vous avez installé le JDK à l'emplacement par défaut, la valeur devrait être "C:\Program Files\Java\jdk1.8.0_281".
8. Dans la même interface, recherchez la variable "Path" et ajoutez la valeur suivante : "%JAVA_HOME%\bin".
9. Cliquez sur "OK" pour enregistrer la variable d'environnement.
10. Fermez toutes les fenêtres ouvertes et redémarrez Android Studio.

7. Créer un émulateur Android

Android Emulateur offre la possibilité de simuler différents appareils Android sur un ordinateur afin de tester des applications sur diverses versions d'Android sans avoir besoin de posséder chaque appareil physique. Il permet une grande flexibilité dans la simulation de diverses configurations d'appareils, comme des téléphones, des tablettes, des montres connectées et des téléviseurs Android. En outre, l'émulateur offre presque toutes les fonctionnalités d'un véritable appareil Android, comme la possibilité de passer des appels et des SMS, de simuler la localisation, le réseau, la rotation et d'autres capteurs matériels, et d'accéder au Google Play Store. Enfin, tester votre application sur l'émulateur est plus rapide et plus facile que sur un appareil physique, notamment grâce à la possibilité de transférer des données vers l'émulateur plus rapidement que vers un appareil connecté en USB. La création et l'exécution d'un émulateur est très simple et se déroule selon les étapes suivantes :

Ouvrez Android Studio.

• Si vous n'avez pas encore installé Android Studio, suivez les étapes décrites dans le titre précédent pour le télécharger et l'installer.

1. Cliquez sur "AVD Manager" dans la barre d'outils ou allez dans "Tools" > "AVD Manager".
    - L'AVD Manager (Android Virtual Device Manager) vous permet de créer, configurer et gérer des émulateurs Android.
2. Cliquez sur "Create Virtual Device".
    - Cette option vous permet de créer un nouvel émulateur.
3. Sélectionnez le type de téléphone ou de tablette Android que vous souhaitez émuler.
    - Vous pouvez choisir parmi plusieurs options, telles que Nexus, Pixel ou Android TV.
4. Cliquez sur "Next".
5. Sélectionnez la version d'Android que vous souhaitez émuler.
    - Vous pouvez choisir parmi les versions les plus récentes d'Android, ainsi que certaines versions plus anciennes.
6. Configurez les paramètres de votre émulateur.
    - Vous pouvez modifier des paramètres tels que la taille de l'écran, la résolution, le stockage, la RAM et le système d'exploitation.
7. Cliquez sur "Finish".
    - Vous avez maintenant créé un nouvel émulateur Android.
8. Pour lancer l'émulateur, cliquez sur le bouton "Play" à côté de l'émulateur dans l'AVD Manager.
    - L'émulateur Android commencera à s'exécuter. Soyez patient, car cela peut prendre un certain temps en fonction de la performance de votre ordinateur.