Speech Presentation

Introduction (1 min)

Bonjour à tous, aujourd'hui nous allons vous présenter notre étude sur la **lecture de codes-barres dans une application Android**.

Ce travail s'inscrit dans le cadre d'un **travail théorique**, dont l'objectif est d'explorer une fonctionnalité avancée de l'écosystème Android et de produire un guide détaillé sur son utilisation. Notre mission était donc d'analyser cette technologie, d'expliquer ses usages et ses limites, et de fournir un tutoriel permettant de l'intégrer facilement dans une application mobile.

Nous avons structuré notre travail en plusieurs parties :

- **Comprendre la lecture de codes-barres et ses enjeux**
- **Explorer les outils et bibliothèques disponibles**
- **Présenter une approche d'implémentation**
- **Analyser les limites et les alternatives**

Dans cette présentation, nous allons donc vous donner une vision claire de cette fonctionnalité et de la manière dont elle peut être exploitée sur Android.

Comprendre la lecture de codes-barres (1 min)

Définition et types de codes supportés

La lecture de codes-barres est une technologie permettant d'extraire des données encodées dans un format graphique à l'aide d'un capteur, généralement une caméra.

On distingue principalement :

- **Les codes 1D** (EAN-13, Code 128), utilisés pour les articles de commerce et la logistique.
- **Les codes 2D** (QR Codes, Data Matrix), capables de stocker plus d'informations et souvent utilisés pour les paiements ou l'authentification.

Problématiques résolues

Cette technologie est utilisée pour :

- **Automatiser des tâches**, comme l'inventaire ou la gestion des stocks.
- **Faciliter l'identification rapide d'objets** dans les commerces ou les entrepôts.
- **Améliorer l'expérience utilisateur**, notamment dans les applications de paiement mobile ou d'authentification.

Dans le contexte d'Android, intégrer cette fonctionnalité permet d'apporter une solution rapide et efficace aux applications nécessitant un traitement de données via un simple scan.

Exploration des outils et bibliothèques disponibles (1 min)

Deux bibliothèques principales permettent d'intégrer la lecture de codes-barres sous Android :

1. **ML Kit Barcode Scanning** (Google)
2. **ZXing** (open-source)

Nous avons analysé leurs caractéristiques et avons retenu **ML Kit Barcode Scanning**, car :

- Il est **facile à intégrer** dans une application Android.
- Il est **optimisé pour les performances**, avec une détection rapide.
- Il **prend en charge de nombreux formats de codes-barres**.

En contrepartie, il nécessite **Google Play Services**, ce qui peut poser des limites dans certains cas.

Approche d'implémentation et tutoriel (1 min)

Dans notre guide, nous avons détaillé les **étapes clés** pour intégrer ML Kit Barcode Scanning dans une application Android :

1. **Ajout des dépendances** nécessaires dans le projet.
2. **Configuration des permissions caméra** dans `AndroidManifest.xml`.
3. **Utilisation de CameraX** pour capturer une image et l'envoyer à ML Kit.
4. **Traitement et affichage des résultats dans une interface utilisateur**.
5. **Gestion des erreurs** en cas de mauvaise lecture.

L'objectif de ce guide est d'aider un développeur découvrant cette fonctionnalité à l'intégrer rapidement dans son projet, en ayant un aperçu des bonnes pratiques et des éventuelles difficultés.

Explication du code (30 secondes)

Nous avons maintenant vu comment ML Kit Barcode Scanning s'intègre dans une application Android. Voici un **exemple de code** extrait de notre tutoriel :

1. **Nous créons un scanner ML Kit** pour analyser les codes-barres.
2. **Nous configurons CameraX** pour capturer les images en temps réel.
3. **Nous analysons chaque image** en ne traitant que la plus récente pour optimiser la performance.
4. **Nous envoyons l'image à ML Kit**, qui extrait les données du code-barres et les affiche dans l'application.

Ce code permet donc d'avoir une **lecture fluide et optimisée**, en s'assurant que le scan ne se déclenche que lorsque l'utilisateur le demande.

Limitations et alternatives (1 min)

Malgré son efficacité, la lecture de codes-barres présente certaines **limitations** :

- **Qualité de la caméra** : Une faible résolution ou un mauvais éclairage peut gêner la reconnaissance.
- **Dépendance à Google Play Services** : ML Kit ne fonctionne pas sans connexion à Google Play.

Pour contourner ces limites, certaines **alternatives** existent :

- **ZXing**, qui fonctionne sans dépendance aux services Google.
- **OCR (Reconnaissance Optique de Caractères)** pour des documents texte plutôt que des codes-barres.

Enfin, bien que cela n'ait pas été exigé dans le cadre du projet, nous avons choisi d'implémenter un exemple pratique de lecture de codes-barres afin de mieux comprendre son comportement et d'illustrer notre guide.

Conclusion

À travers cette étude, nous avons pu démontrer que la **lecture de codes-barres** est une **fonctionnalité essentielle** pour de nombreuses applications Android, notamment dans le commerce et la logistique. Nous avons analysé les différentes solutions disponibles, et proposé un **guide détaillé** pour aider les développeurs à intégrer ML Kit dans leurs projets.

Bien que nous ayons également exploré une implémentation pratique, notre travail s'est principalement concentré sur **l'analyse théorique** et la mise en place d'un **tutoriel avancé** conforme aux attentes du projet.

Merci pour votre attention.
