Pour mon application, j'ai pris la décision de sélectionner plusieurs solutions énumérées ci-dessous. Chacune de ces solutions est accompagnée des raisons qui ont motivé mes choix.

Concernant la persistance des données, dans le but d'explorer de nouvelles perspectives, j'ai opté pour l'utilisation de Firebase plutôt que CoreData que l'on a vu dans le cours.
Firebase offre une solution de persistance des données dans le Cloud, permettant aux utilisateurs de sauvegarder et de récupérer leurs données, même après la désinstallation de l'application.
De plus, Firebase est largement adopté dans le domaine du développement, ce qui fait de mon expérience avec cette plateforme un atout majeur lors de mes futurs entretiens professionnels.

Avantages : Option gratuite (Spark), le développeur se concentre sur le frontend, authentification, outils d'analyses intégrés.
Inconvénients : Base de données NoSQL (moins de rigueur), pas de query complexes.

Pour assurer le suivi des performances, Firebase Analytics apparaît comme une option pertinente.
Étant donné mon utilisation de Firebase, il était logique d'opter pour l'une de leurs solutions.
Firebase Analytics est spécialement conçu pour le monde du mobile, tandis que Google Analytics, par exemple, est davantage orienté vers le web.

Avantages : Intégration facile, événements personnalisés, création d'audiences (notifications ciblées, etc.)
Inconvénients : Personnalisation limitée, difficile à appréhender pour les nouveaux utilisateurs.

En complément de Firebase Analytics, j'ai ajouté Firebase Crashlytics afin d'avoir une gestion des crashs plus détaillées.

Avantages : Précision (nom de la classe + ligne du crash), Android + iOS, crash par version.
Inconvénients : Installation contraignante (scripts),

En ce qui concerne la gestion des dépendances, j'ai opté pour l'utilisation de Swift Package Manager.
Ayant déjà utilisé CocoaPods, j'ai souhaité explorer une autre approche en utilisant SPM (Swift Package Manager).
Les deux méthodes sont largement répandues et il est donc bénéfique de les maîtriser toutes les deux.

Avantages : Gère les dépendances des dépendances, standard d'Apple, facilité dans l'ajout d'un package.
Inconvénients : Maturité (outil récent), fonctionne à partir d'Xcode 11.

Pour la solution d'intégration continue, j'ai opté pour GitHub Actions qui a été ma première préférence.
La raison principale est que mon dépôt est hébergé sur GitHub et j'ai souhaité rester au plus près de cette plateforme pour bénéficier d'une certaine centralisation de mon application.

Avantages : Market, facile d'utilisation, choix des machines virtuelles (Linux, Windows, MacOS), jobs en simultanés.
Inconvénients : Dépendance à GitHub, connaissances en YAML (peut devenir compliqué en cas de workflows complexes)