Développement natif sous iOS

Cours n°9 - Aller plus loin

iOS

Quelques conseils

(Qui ne sont pas propres qu'au développement iOS)

- Logging
- Sécurité
- Performances
- Tests

Logging

- Logguer / « journaliser » les étapes de votre code qui peuvent échouer ou qui sont sensibles

- Utiliser le bon niveau de rapport :
  - `trace` / `debug` pour ce qui peut-être utile en debug
  - `info` pour ce qui peut être utile mais non essentiel
  - `notice` pour ce qui est essentiel
  - `error` / `warning` lorsqu'un évènement important a lieu

Logging

- Effectuer l'appel à la journalisation **avant** l'exécution du code pouvant être problématique
    - Sinon, si une erreur est levée, la journalisation n'aura pas lieu
- Dans le cas d'une application mobile, les journaux locaux sont pratiques pour débogguer
    - Pour être averti des erreurs rencontrées par les utilisateurs, il faudra rapporter les erreurs de manière distante (via un web service par exemple)
- Ne pas tout journaliser ; que ce qui peut-être utile

Logging

- En Swift, vous pouvez utiliser la classe `Logger`

- https://developer.apple.com/documentation/os/logging/generating_log_messages_from_your_code

Sécurité

- Ne pas partir du principe que si votre code n’est pas publique, il est sans risque

- Ne rien stocker ni communiquer de sensible en clair
    - Chiffrer dès que nécessaire

- Utiliser des standards éprouvés dès que possible (e.g. JWT pour communiquer avec votre éventuel web service)

- Ne **jamais** faire confiance aux données envoyées par l’utilisateur
    - C.f. failles XSS ou injections SQL

Sécurité

- Proposez de l’authentification à deux facteurs (2FA) si possible
    - Rappel : La 2FA avec confirmation par SMS n’est **pas** sécurisée
- Tenez-vous au courant des nouvelles versions des bibliothèques externes que vous utilisez (via GitHub, mailing-lists, etc.) pour scruter les mises à jour de sécurité

Performances

- Faire attention aux micro-benchmarks (mesure des performances d’un petit bout de code)
    - Un code rapide en micro-benchmark peut s’avérer lent une fois intégré dans une application
- Tout code n’est pas parallélisable / exécutable en concurrence
    - Par exemple, si votre code dépend de certains verrous (e.g. allocation mémoire, accès fichier, etc.)

Performances

- Ne jamais utiliser `DispatchQueue.global`
    - Le nombre de threads peut rapidement exploser
- Les files concurrentes (*concurrent queues*) sont moins optimisées que les files en série (*serial queues*)
    - Mieux vaut utiliser une file en série par défaut et passer à une file concurrente si les performances ne semblent pas idéales
- https://developer.apple.com/documentation/xcode/improving_your_app_s_performance

Tests

- Tester au niveau unitaire mais aussi l'intégration des différents composants

- Lors de la correction d'un bug, ajouter un test de régression
    - Ce test permettra d'éviter que le bug ne soit ré-introduit

- Ne tester qu'une fonctionnalité par test
    - Si des choses doivent être remises à 0 durant votre test, il faut sûrement le découper en plusieurs morceaux

Tests

- Si possible, utiliser de l'intégration continue (*continuous integration* ou CI) pour lancer vos tests à chaque modification

- Développer vos webservices pour qu'ils puissent être utilisés lors de tests (e.g. possibilité d'utiliser une base de données « bac à sable »)

Quelques outils utiles

- Eureka
- SwiftGen
- Fastlane

Eureka

- Bibliothèque permettant de créer des formulaires simplement

- https://github.com/xmartlabs/Eureka

SwiftGen

- Bibliothèque permettant de générer automatiquement du code Swift pour vos ressources

- Ceci permet de pallier les éventuels problèmes liés à la localisation de l’application

- https://github.com/SwiftGen/SwiftGen

Fastlane

- Outil permettant d’automatiser le déploiement de vos applications (envoi sur les stores, captures d’écran, etc.)
- Fonctionne également pour Android
- https://fastlane.tools/

Quelques liens utiles

- https://developer.apple.com/support/app-store/
    - Statistiques d’usage des dernières versions d’iOS
    - Documents utiles pour la mise en ligne de vos application sur le store
- https://github.com/dkhamsing/open-source-ios-apps
    - Liste de quelques applications iOS dont le code source est ouvert

Quelques liens utiles

- https://github.com/matteocrippa/awesome-swift
    - Liste de ressources intéressantes sur le langage Swift
- https://xcodebuildsettings.com
    - Glossaire des différentes options de la commande `xcodebuild` (permettant de compiler vos projets)