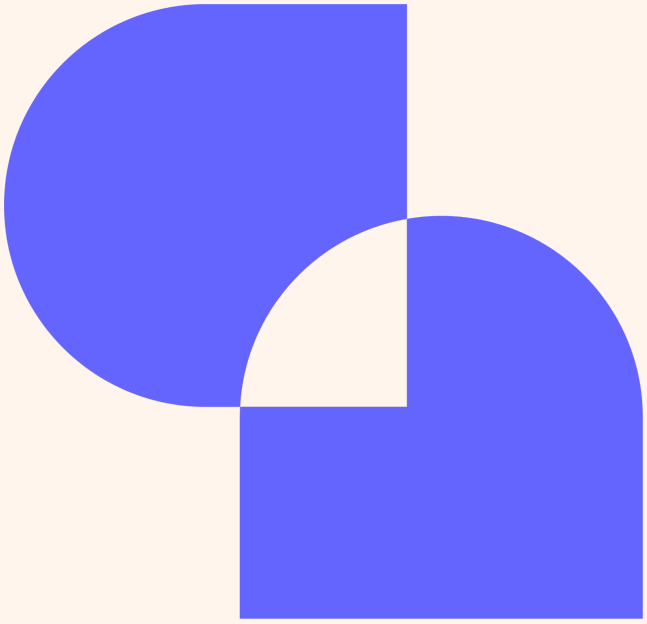
GoMind

“Agir avec l’esprit éveillé”

DOSSIER DE COMPÉTENCES



Développeur Sénior SwiftUI

Spécialisé : MVVM, SwiftUI

8 ans d’expérience

PRINCIPALES EXPÉRIENCES

SEPHORA: Développeur Swift/SwiftUI (2023-2025)
RADIO FRANCE: Développeur Swift/UIKit (2019 - 2023)
BFORBANK : Développeur Swift/UIkit (2019)
ARROWS : Développeur Swift/UIkit (2017-2019)

CONNAISSANCES TECHNIQUES

Langages : Swift, Objective-C, Kotlin, Java, PHP, JavaScript, Ruby, Typescript

Librairies/framework : SwiftUI, UIKit, Combine, RxSwift, Swift Data, Core Data, Swift Package Manager, Cocoapods, Async/Await, Foundation, Map Integration (MapKit, Core Location), Gestion Multimedia (AVFoundation, AVKit), Animation et Transition (Core Animation, Core Graphics), Phrase/Localisation, Keychain, Location Manager, Biométrie (FaceID, TouchID)

Architecture/Design Pattern : MVVM-C, Clean Archi, TCA,, Observer (Delegate, KVO), Singleton, DI, DTO, Coordinator, Adapter, SOLID, Composition, Factory, Facade, DRY, KISS,

Base de données : CoreData, Swift Data, API Rest, Realm, Firebase, UserDefault, SQLlite

Bonnes pratiques : TDD, BDD, XCTest, TestFlight, Postman, TestUI system, Swiftlint, ViewInspector, Sourcery, Quick&Numble

Outils CI/CD : Jenkins, Github Action,, Bitrise, Fastlane, Git, Gitflow, Crashlytics, Google Tag Analytics, Firebase Performance & APP Distribution, SonarQube, SwiftLint, XcodeGEn, Azure DevOps, Postman

Gestion de projet : Agile, Scrum, Kanban, Figma, JIRA, Confluence, Notion

EXPÉRIENCES TECHNIQUES

SEPHORA : Développeur Swift/SwiftUI (2023-2025)

Refonte complète de l'application Sephora destinés aux clients B2C.

- Analyse du besoin
- Ecriture du cahier des charges
- Écritures des spécifications fonctionnelles et techniques
- Échange avec la PO et les UX
- Suivi des wireframes avec Figma
- Équipe de 29 personnes :
    - 4 PO
    - 5 testeurs
    - 16 développeurs
    - 4 UX/UI Designer
- Gestion de projet en SCRUM :
    - Sprint de 3 semaines
    - Daily meeting, Retro, Planning, Refinement
    - Animation de iOS Technical meeting
- Mise en place de l'architecture MVVM :
    - Mise en place de la clean architecture : SOLID, Kiss, Dry,
    - Mise en place des designs patterns : Factory, Singleton, Strategy
    - Utilisation de Combine : observable object, publisher, operator
    - Migration de React Native vers SwiftUI
    - Migration de Cocoapods vers SPM
        - Création de features modules
    - Mise en place de l'injection de dépendances avec Swinject
- Exemples de features développées :
    - Checkout (SwiftUI)
    - Page d'accueil (UIkit)
- Développement Swift :
    - Utilisation des API PayPal, Klarna, ApplePay
    - Gestion du Lazy
    - Utilisation de Guards
- Développement SwiftUI :
    - Front :
        - Migration des vues UIkit vers SwiftUI
        - Migration de RxSwift vers Combine
        - Gestion de States Object
        - Gestion des Observed object
        - Gestion de la navigation
        - Gestion du Binding
        - Mise en place des compositional Layout pour afficher les éléments en list ou à l'horizontal
    - Back :
        - Utilisation de Swift concurrency pour la partie back
        - Utilisation d'URLSession
        - Utilisation de Firebase
        - Gestion du tracking avec Analytics

        - Ajout d'une pagination et pull to refresh sur la liste des produits
        - Mise en place d'un gestionnaire de DeepLink
        - Mise en place des Retry pour les call API
- Gestion de la base de données :
    - Test d'API avec Postman et Swagger
- Versioning avec Git
    - Suivi du Gitflow
- Écriture des tests unitaires de XCtests :
    - Tester les ViewModels, tests des services (Network) et UseCases
    - Création des Script Shell pour passer d'un protocole à une classe (génération des mock) avec Sourcery et SwiftyMocky
- Pratique du TDD
- Écriture des tests unitaires avec Quick & Numble
- Utilisation de Birtrise
- Utilisation de SwiftLint
- Bonne pratiques :
    - Code review
    - Écriture de la documentation
    - Refacto de code
    - Pair programming

Environnement technique :
UseCases, SOLID, DRY, KISS, Factory, Singleton, Strategy, Swinject, Swift, SwiftUI, Combine, async/await, URLSession, Firebase, API Paypal, Klarna, ApplePay, Binding, SwiftyMocky, TDD, XCTest, Lazy, Guards, Xcode, SPM, Postman, Swagger, SwiftLint, XcodeGen, Figma, Gitflow

---

RADIO FRANCE: Développeur Swift/UIKit (2019 - 2023)

Développement from scratch des applications FIP et Mouv'.

- Analyse du besoin
- Ecriture du cahier des charges
- Écritures des spécifications fonctionnelles et techniques
- Échange avec la PO et les UX
- Suivi des wireframes avec Figma
- Équipe de 7 personnes :
    - 1 PO
    - 1 testeurs
    - 4 développeurs
    - 1 UX/UI Designer
- Gestion de projet en SCRUM :
    - Sprint de 2 semaines
    - Daily meeting, Retro, Planning, Refinement
    - Animation de iOS Technical meeting
- Mise en place de l'architecture MVVM :
    - Mise en place de la clean architecture : SOLID, Kiss, Dry
    - Mise en place des designs patterns : Factory, Singleton, Strategy
    - Gestion de l'injection de dépendances avec Swinject

    - Utilisation de RxSwift/RxCocoa
- Développement Swift :
    - Gestion du Lazy
    - Utilisation de Guards
- Développement UIKit :
    - Utilisation de RxSwift
    - Gestions des animations avec Hero pour gérer les transitions entre pages
    - Gestion du Compositional Layout
    - Gestion de la navigation
    - Utilisation de Alamofire
    - Utilisation de Firebase
    - Gestion du tracking avec Analytics
    - Ajout d'une pagination et pull to refresh sur la liste des médias
    - Utilisation de AVFoundation,
    - Gestion de la lecture en différé,
    - Gestion de la lecture en streaming via Airplay, GoogleCast, CarPlay et AndroidAuto
- Gestion de la base de données :
    - Realm
- Versioning avec Git
    - Suivi du Gitflow
- Écriture des tests unitaires de XCtests :
    - Tester les ViewModels, tests des services (Network) et UseCases
    - Création des Script Shell pour passer d'un protocole à une classe (génération des mock) avec Sourcery et SwiftyMocky
- Pratique du TDD
- Utilisation de SwiftLint
- Mise en place de la CI avec Gitlab
- Mise en place de la CD avec Fastlane
    - Utilisation Slather pour la récupération du code coverage
    - Déploiement sur Firebase App Distribution
- Mise en place de Bitrise
    - Écriture de feature build
    - Mise en place du nightly build
- Bonne pratiques :
    - Code review
    - Écriture de la documentation
    - Refacto de code
    - Pair programming

Environnement technique:
MVVM, Clean Architecture, SOLID, KISS, DRY, UseCases, Swinject, Factory, Singleton, Strategy, Swift, Script Shell, RxSwift/RxCocoa, UIKit, AVFoundation, Airplay, GoogleCast, CarPlay, AndroidAuto, XCTest, SwiftyMocky, Sourcery, TDD, Lazy, Guards, Figma, SwiftLint, Code review, Documentation, Refactorisation, Pair programming, Agile (SCRUM),Git, Gitflow

---

BFORBANK : Développeur Swift/UIkit (2019)

Développement from scratch d'une application BForBank à destination aux clients B2C;

- Analyse du besoin
- Ecriture du cahier des charges
- Écritures des spécifications fonctionnelles et techniques
- Équipe de 12 personnes :
    - 1 PO
    - 2 testeurs
    - 8 développeurs
    - 1 UX/UI Designer
- Gestion de projet en SCRUM :
    - Sprint de 2 semaines
    - Daily meeting, Retro, Planning, Refinement
    - Animation de iOS Technical meeting
- Mise en place de l'architecture MVVM :
    - Mise en place de la clean architecture : SOLID, Kiss, Dry,
    - Mise en place des designs patterns : Factory, Singleton, Strategy
    - Gestion de l'injection de dépendances avec Factory
    - Utilisation de RxSwift/RxCocoa
- Développement Swift :
    - Gestion du Lazy
    - Utilisation de Guards
- Développement UIKit :
    - Implémentation de l'écran "Mes Budgets"
    - Mise en place de l'injection de dépendances (maison)
    - Implémentation de l'authentification biométrique avec TouchId et FaceId
    - Mise en conformité avec DSP2
    - Gestion de la navigation
    - Utilisation d'URLSession
    - Utilisation de Firebase
    - Gestion du tracking avec Analytics
- Gestion de la base de données :
    - Realm
- Versioning avec Git
    - Suivi du Gitflow
- Pratique du TDD
- Écriture des tests unitaires et d'intégrations avec XCtests
- Mise en place de SwiftLint
- Bonne pratiques :
    - Code review
    - Écriture de la documentation
    - Refacto de code
    - Pair programming

Environnement technique :
MVVM, Clean Architecture, SOLID, KISS, DRY, UseCases, Swinject, Factory, Singleton, Strategy, Swift, Script Shell, RxSwift/RxCocoa, UIKit Firebase, Git, Gitflow, TDD, XCtests, Jira, Firgma

---

ARROWS : Développeur Swift/UIkit (2017-2019)

Développement from scratch de *Tesla4U*, une application de suivi d'usage des véhicules Tesla destinée aux usagers.

- Analyse du besoin
- Ecriture du cahier des charges
- Écritures des spécifications fonctionnelles et techniques
- Équipe de 12 personnes :
    - 1 PO
    - 4 développeurs
    - 1 UX/UI Designer
- Gestion de projet en SCRUM :
    - Sprint de 2 semaines
    - Daily meeting, Retro, Planning, Refinement
    - Animation de iOS Technical meeting
- Mise en place de l'architecture MVVM :
    - Mise en place de la clean architecture : SOLID, Kiss, Dry,
    - Mise en place des designs patterns : Factory, Singleton, Strategy
    - Gestion de l'injection de dépendances avec Factory
    - Utilisation de RxSwift/RxCocoa
- Développement Swift :
    - Gestion du Lazy
    - Utilisation de Guards
- Développement UIKit :
    - Implémentation de l'écran "Mon véhicule" pour visualiser l'autonomie, la batterie et l'efficience de conduite
    - Implémentation de l'écran "Historique" pour visualiser les trajets sur Maps, les charges et la consommation de la batterie
    - Implémentation de l'écran "Automatisations" pour lancer le chauffage et la charge
    - Gestion de la navigation
    - Utilisation d'URLSession
    - Utilisation de Firebase
    - Gestion du tracking avec Analytics
- Gestion de la base de données :
    - CoreData
- Versioning avec Git
    - Suivi du Gitflow
- Pratique du TDD
- Écriture des tests unitaires de XCtests :
    - Tester les ViewModels
- Utilisation de SwiftLint
- Bonne pratiques :
    - Code review
    - Écriture de la documentation
    - Refacto de code
    - Pair programming

Environnement technique :
MVVM, Clean Architecture, SOLID, KISS, DRY, injection de dépendances,Factory, Singleton, Strategy, Swift, RxSwift/RxCocoa, UIKit, URLSession, Firebase, Analytics, CoreData, XCTest, TDD, Lazy, Guards,SwiftLint, Code review, Documentation, Refactorisation, Pair programming,SCRUM,Gitflow

FORMATION

Université De Caen Normandie: Master Pro Document Numérique en Réseaux et Ingénierie de l'Internet (2017)

LANGUES

Français

Anglais