Mindful

Guide Architectural de Référence

Kickstart du Projet - 3 Parties

Table des Matières

Partie 1

Backend Local de l'App

Stack Technique Jour 1

- SwiftData : Persistence moderne (remplace Core Data)
- Actors : Thread-safety garantie
- async/await : Toutes les opérations asynchrones
- Sendable : Sécurité compile-time Swift 6

Objectif : Zero data races, 100% thread-safe

Models à Créer

```
@Model class JournalEntry
@Model class Photo

struct Location: Codable, Sendable
struct Weather: Codable, Sendable
enum Mood: String, Codable, Sendable
```

Ressources :

- SwiftData Documentation
- WWDC23: Meet SwiftData
- WWDC23: Model your schema with SwiftData

JournalEntry - Propriétés Clés

```
@Model final class JournalEntry {
    var id: UUID
    var createdAt: Date
    var modifiedAt: Date
    var title: String
    var content: String
    var mood: Mood
    var tags: [String]

    @Relationship(deleteRule: .cascade)
    var photos: [Photo]

    var location: Location?
    var weather: Weather?
    var isFavorite: Bool
}
```

Mood - Enum pour Humeurs

```
enum Mood: String, Codable, Sendable {
    case veryHappy, happy, neutral, sad, verySad

    var emoji: String { /* 😄🙂😐😔😢 */ }
    var displayName: String { /* "Très Joyeux" */ }
    var color: Color { /* .green, .blue, .red */ }
    var score: Int { /* -2 à +2 */ }

    static func from(sentimentScore: Double) -> Mood
}
```

Photo - Model pour Images

```
@Model final class Photo {
    var id: UUID
    var createdAt: Date

    @Attribute(.externalStorage)
    var imageData: Data

    var thumbnailData: Data
    var caption: String?
    var entry: JournalEntry? // Inverse relationship
    var width: Int
    var height: Int
}
```

SwiftData Configuration

```
@MainActor
class DataController {
    static let shared = DataController()
    let container: ModelContainer

    private init() {
        let schema = Schema([
            JournalEntry.self,
            Photo.self
        ])

        let config = ModelConfiguration(
            cloudKitDatabase: .automatic
        )

        container = try! ModelContainer(
            for: schema,
            configurations: config
        )
    }
}
```

Actors à Implémenter

```
actor JournalRepository {
    // CRUD operations
    func create(_ entry: JournalEntry) async throws
    func fetchAll() async throws -> [JournalEntry]
    func update(_ entry: JournalEntry) async throws
    func delete(_ entry: JournalEntry) async throws
    func search(query: String) async throws -> [JournalEntry]
}

actor PhotoProcessor {
    func process(_ image: UIImage) async throws -> ProcessedPhoto
    func generateThumbnail(_ image: UIImage) async -> Data
    func compress(_ data: Data) async -> Data
}

actor ExportService {
    func exportToPDF(_ entries: [JournalEntry]) async throws -> URL
    func exportToMarkdown(_ entries: [JournalEntry]) async throws -> URL
    func exportToJSON(_ entries: [JournalEntry]) async throws -> URL
}
```

JournalRepository - Méthodes Clés

CRUD :

- `create()`, `update()`, `delete()`
- `fetchAll()`, `fetch(from:to:)`, `fetch(mood:)`

Recherche :

- `search(query: String)` - Full-text search
- `fetchFavorites()` - Entrées favorites

Stats :

- `count()` - Nombre total d'entrées
- `currentStreak()` - Jours consécutifs d'écriture
- `moodDistribution()` - Répartition des humeurs

PhotoProcessor - Pipeline

```
UIImage → [compress] → [generateThumbnail] → [upload?]
                                                  ↓
                                            ProcessedPhoto
```

Méthodes :

- `process(_ image: UIImage) async throws -> ProcessedPhoto`
- `processMultiple(_ images: [UIImage]) async throws -> [ProcessedPhoto]`
- `generateThumbnail(size: CGSize) async -> Data`
- `compress(quality: CGFloat) async -> Data`

Performance : *TaskGroup pour traiter plusieurs photos en parallèle*

ExportService - Formats

Formats supportés :
- PDF (avec images et formatting)
- Markdown (texte pur, portable)
- JSON (backup complet)

Méthodes :

```swift
func exportToPDF(_ entries: [JournalEntry]) async throws -> URL
func exportToMarkdown(_ entries: [JournalEntry]) async throws -> URL
func exportToJSON(_ entries: [JournalEntry]) async throws -> URL
func exportAll(formats: [ExportFormat]) async throws -> [URL]
```

Ressource : PDFKit pour génération PDF

Ressources Backend - SwiftData

Documentation officielle :
- SwiftData Overview
- Modeling data with SwiftData

WWDC Sessions :
- WWDC23: Meet SwiftData (10187)
- WWDC23: Model your schema with SwiftData (10195)
- WWDC23: Dive deeper into SwiftData (10196)

Articles :
- SwiftData by Example

Ressources Backend - Concurrency

Swift Concurrency :
- Concurrency Documentation
- Actor Documentation

WWDC Sessions :
- WWDC21: Meet async/await in Swift (10132)
- WWDC21: Protect mutable state with Swift actors (10133)
- WWDC22: Eliminate data races using Swift Concurrency (110351)

Swift Evolution :
- SE-0296: async/await
- SE-0304: Structured concurrency
- SE-0306: Actors

Partie 2

Liquid Glass UI

Design System - Palette de Couleurs

```swift
extension Color {
    // Gradients principaux
    static let mindfulPrimary = Color(hex: "667eea")
    static let mindfulSecondary = Color(hex: "764ba2")

    // Glass effects
    static let glassBackground = Color.white.opacity(0.2)
    static let glassBorder = Color.white.opacity(0.6)

    // Moods
    static let moodVeryHappy = Color.green
    static let moodHappy = Color(hex: "90EE90")
    static let moodNeutral = Color.blue
    static let moodSad = Color.orange
    static let moodVerySad = Color.red
}
```

Design System - Typography

```swift
extension Font {
    // Headings
    static let entryTitle = Font.system(
        size: 28, weight: .bold, design: .rounded
    )
    static let cardTitle = Font.system(
        size: 18, weight: .semibold
    )

    // Body
    static let entryBody = Font.system(
        size: 16, weight: .regular
    )
    static let caption = Font.system(
        size: 14, weight: .regular
    )

    // Special
    static let date = Font.system(
        size: 12, weight: .medium, design: .monospaced
    )
}
```

Components à Créer

Core Components :

- `GlassCard` - Container de base
- `GlassButton` - Boutons avec effet glass
- `MoodSelector` - Picker d'humeur animé
- `TagPill` - Pills pour les tags
- `PhotoGallery` - Galerie de photos

Complex Components :

- `EntryCard` - Card pour timeline
- `EntryEditor` - Éditeur rich text
- `StatsCard` - Card pour statistiques
- `SearchBar` - Barre de recherche glass

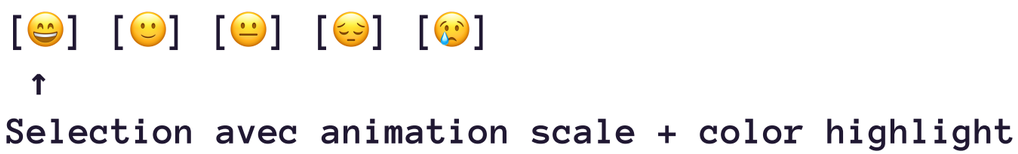
GlassCard - Signature

```
struct GlassCard<Content: View>: View {
    let content: Content

    init(@ViewBuilder content: () -> Content)

    var body: some View {
        content
            .padding()
            .background(.ultraThinMaterial)
            .cornerRadius(20)
            .shadow(...)
            .overlay(borderGradient)
    }
}
```

Usage :

```
swift
GlassCard {
    Text("Hello")
}
```

MoodSelector - UI



Features :

- Sélection tactile
- Animation spring au tap
- Couleur de fond change selon mood
- Haptic feedback

Ressource :

- UIFeedbackGenerator pour haptics
- Human Interface Guidelines - Haptics

Views Architecture

```
MindfulApp
└── ContentView
    ├── TimelineView (liste des entrées)
    │   └── EntryCard (row)
    ├── EditorView (nouvelle/édition entrée)
    │   ├── MoodSelector
    │   ├── TextEditor
    │   └── PhotoGallery
    ├── DetailView (lecture entrée)
    │   └── EntryContent + Photos
    └── StatsView (dashboard)
        └── StatsCards
```

TimelineView - Structure

```
struct TimelineView: View {
    @State private var viewModel = TimelineViewModel()
    @Namespace private var animation

    var body: some View {
        ScrollView {
            LazyVStack(spacing: 20) {
                ForEach(viewModel.entries) { entry in
                    EntryCard(entry: entry)
                        .matchedGeometryEffect(
                            id: entry.id,
                            in: animation
                        )
                }
            }
        }
        .background(gradientBackground)
    }
}
```

EditorView - Features

Composants :
- `TextEditor` natif SwiftUI
- `MoodSelector` en header
- `PhotoPicker` (PhotosUI)
- `TagField` avec suggestions
- Toolbar glass en bas (save, cancel, add photo)

Auto-save :
- Debouncing avec Task cancellation
- Sauvegarde toutes les 2 secondes
- Indicateur "Saved" subtil

Writing Tools : Intégration automatique iOS 18+

Animations - Patterns

Hero Animation :
```swift
@Namespace private var animation


.matchedGeometryEffect(id: entry.id, in: animation)
```

Spring Animations :
```swift
withAnimation(.spring(response: 0.6, dampingFraction: 0.7)) {
    // state change
}
```

Phase Animator :
```swift
PhaseAnimator([false, true]) { phase in
    // animate through phases
}
```

Animations - Transitions

```
extension AnyTransition {
    static var slideAndFade: AnyTransition {
        .asymmetric(
            insertion: .move(edge: .trailing)
                       .combined(with: .opacity),
            removal: .move(edge: .leading)
                     .combined(with: .opacity)
        )
    }
}
```

Usage :

```
swift
if showDetail {
    DetailView()
        .transition(.slideAndFade)
}
```

Skeleton Loaders

```
struct SkeletonView: View {
    @State private var shimmerOffset: CGFloat = -1

    var body: some View {
        RoundedRectangle(cornerRadius: 12)
            .fill(.gray.opacity(0.3))
            .overlay(shimmerGradient)
            .onAppear {
                withAnimation(.linear(duration: 1.5)
                    .repeatForever(autoreverses: false)) {
                    shimmerOffset = 1
                }
            }
    }
}
```

Ressources UI - SwiftUI

Documentation officielle :
- SwiftUI Documentation
- SwiftUI Tutorials

WWDC Sessions :
- WWDC23: Animate with springs (10158)
- WWDC23: Demystify SwiftUI performance (10160)
- WWDC22: SwiftUI on iPad: Add toolbars (10069)

Communauté :
- Hacking with Swift - SwiftUI
- SwiftUI Lab

Ressources UI - Design

Apple Design Resources :

- Human Interface Guidelines
- SF Symbols App

Inspiration :

- Dribbble : recherche "journal app" ou "glass design"
- Mobbin : collection d'apps iOS design

Glassmorphism :

- Articles sur le style "Liquid Glass" / "Glassmorphism"
- Figma templates pour référence

Partie 3

Apple Intelligence & Data

Apple Intelligence - Stack

On-Device :

- Natural Language : Sentiment, keywords, language detection
- Vision : Analyse d'images (optionnel)
- Core ML : Models custom (optionnel)

iOS 18+ Features :

- App Intents : Siri, Shortcuts, Spotlight
- Writing Tools : Intégration automatique
- Image Playground : Génération d'images (optionnel)

App Intents - Vue d'ensemble

Intents à créer :

```
swift
struct AddJournalEntryIntent: AppIntent
struct SearchEntriesIntent: AppIntent
struct ShowMoodEntriesIntent: AppIntent
struct GetStreakIntent: AppIntent
```

Usage :

- "Hey Siri, add a journal entry about my day"
- "Show me my happy entries in Mindful"
- "What's my writing streak?"

Ressource :

- App Intents Documentation
- WWDC22: Dive into App Intents (10032)

AddJournalEntryIntent - Structure

```
struct AddJournalEntryIntent: AppIntent {
    static var title: LocalizedStringResource = "Add Journal Entry"
    static var description = IntentDescription("Create a new journal entry")

    @Parameter(title: "Entry text")
    var text: String

    @Parameter(title: "Mood")
    var mood: MoodEntity?

    func perform() async throws -> some IntentResult {
        // Créer l'entrée via JournalRepository
        // Retourner succès
    }
}
```

Entity Types - AppEntity

```
struct JournalEntryEntity: AppEntity {
    static var typeDisplayRepresentation =
        TypeDisplayRepresentation(name: "Journal Entry")

    var id: UUID
    var title: String
    var preview: String

    var displayRepresentation: DisplayRepresentation {
        DisplayRepresentation(
            title: "\(title)",
            subtitle: "\(preview)"
        )
    }
}

struct MoodEntity: AppEntity {
    var id: String
    var mood: Mood
    // ...
}
```

Shortcuts - Phrase Suggestions

```
struct MindfulAppShortcuts: AppShortcutsProvider {
    static var appShortcuts: [AppShortcut] {
        AppShortcut(
            intent: AddJournalEntryIntent(),
            phrases: [
                "Add a journal entry in \(.applicationName)",
                "Write about my day in \(.applicationName)",
                "Note my thoughts in \(.applicationName)"
            ],
            shortTitle: "New Entry",
            systemImageName: "square.and.pencil"
        )
    }
}
```

Natural Language - Sentiment Analysis

```swift
import NaturalLanguage

actor SentimentAnalyzer {
    private let tagger = NLTagger(tagSchemes: [.sentimentScore])

    func analyze(_ text: String) async -> Mood {
        tagger.string = text

        let (tag, _) = tagger.tag(
            at: text.startIndex,
            unit: .paragraph,
            scheme: .sentimentScore
        )

        if let score = tag?.rawValue.flatMap(Double.init) {
            return Mood.from(sentimentScore: score)
        }

        return .neutral
    }
}
```

Natural Language - Keyword Extraction

```
actor KeywordExtractor {
    private let tagger = NLTagger(tagSchemes: [.nameType])

    func extract(from text: String) async -> [String] {
        tagger.string = text
        var keywords: [String] = []

        tagger.enumerateTags(
            in: text.startIndex..<text.endIndex,
            unit: .word,
            scheme: .nameType
        ) { tag, range in
            if tag == .personalName ||
               tag == .placeName ||
               tag == .organizationName {
                keywords.append(String(text[range]))
            }
            return true
        }

        return keywords
    }
}
```

Writing Tools - iOS 18

Intégration automatique :

```swift
TextEditor(text: $entry.content)
    // Writing Tools s'activent automatiquement !
```

Features disponibles :

- Proofread (correction orthographe/grammaire)
- Rewrite (réécriture)
- Make Friendly / Professional / Concise
- Summarize
- Extract Key Points
- Create Table

Ressource :

- WWDC24: Meet Writing Tools (10168)

Smart Features à Implémenter

Smart Prompts :
- "Il y a 1 an aujourd'hui, tu écrivais..."
- "Tu n'as pas écrit depuis 3 jours, comment vas-tu ?"
- "Ta semaine en résumé..."

Mood Insights :
- Tendances sur 7/30 jours
- Corrélations (météo, jour de semaine, etc.)
- Suggestions basées sur patterns

Auto-tagging :
- Extraction automatique de keywords
- Suggestions de tags basées sur contenu
- Tags fréquents

Analytics & Stats - Métriques

Streak :
- Jours consécutifs d'écriture
- Record personnel
- Calendrier visuel

Mood Distribution :
- Pourcentage par mood
- Évolution dans le temps
- Graphiques

Writing Stats :
- Nombre total d'entrées
- Nombre de mots écrits
- Temps de lecture total
- Jours les plus productifs

Swift Charts - Exemple

```swift
import Charts

struct MoodChart: View {
    let moodData: [(Date, Mood)]

    var body: some View {
        Chart(moodData, id: \.0) { date, mood in
            BarMark(
                x: .value("Date", date),
                y: .value("Score", mood.score)
            )
            .foregroundStyle(mood.color)
        }
    }
}
```

Ressource :

- Swift Charts Documentation
- WWDC22: Hello Swift Charts (10136)

Privacy - Considérations

Données sensibles :
- Journal = données très personnelles
- Toujours chiffrer (SwiftData le fait automatiquement)
- iCloud sync = chiffré end-to-end

Sentiment Analysis :
- On-device uniquement (NaturalLanguage)
- Aucune donnée envoyée à un serveur
- Transparence dans la Privacy Policy

Best Practices :
- Face ID / Touch ID pour ouvrir l'app (optionnel)
- Pas de screenshot dans app switcher
- Export sécurisé (demander confirmation)

Ressources Apple Intelligence

App Intents :
- App Intents Documentation
- WWDC22: Dive into App Intents (10032)
- WWDC23: Explore enhancements to App Intents (10103)

Natural Language :
- NaturalLanguage Documentation
- WWDC19: Understanding Images in Vision (222)

Writing Tools :
- WWDC24: Meet Writing Tools (10168)
- Writing Tools API

Ressources - Privacy

Documentation :

- App Privacy Details
- Protecting User Privacy

WWDC Sessions :

- WWDC20: Build trust through better privacy (10676)
- WWDC21: Secure your app with App Privacy (10158)

Guidelines :

- Privacy Best Practices

Features Optionnelles (Bonus)

Image Playground (iOS 18) :
- Générer illustrations pour entrées
- "Illustre mon entrée du jour"

Visual Intelligence :
- Scanner du texte manuscrit → entrée digitale
- OCR avec Vision framework

Geolocation :
- Ajouter automatiquement le lieu
- Maps integration

Weather :
- Fetch météo du jour via WeatherKit

Architecture Complète - Recap

```
MindfulApp
├── Data Layer (SwiftData)
│   ├── Models (@Model classes)
│   └── ModelContainer
├── Domain Layer (Actors)
│   ├── JournalRepository
│   ├── PhotoProcessor
│   └── ExportService
├── UI Layer (SwiftUI)
│   ├── Views (Timeline, Editor, Detail, Stats)
│   ├── Components (GlassCard, MoodSelector, etc.)
│   └── ViewModels (@Observable)
└── Intelligence Layer
    ├── App Intents
    ├── NaturalLanguage
    └── Analytics
```

Checklist de Démarrage

Jour 1 :
- [ ] Créer projet Xcode avec SwiftData
- [ ] Définir Models (@Model)
- [ ] Setup ModelContainer
- [ ] Implémenter JournalRepository actor
- [ ] Implémenter PhotoProcessor actor
- [ ] Tests CRUD basiques

Jour 2 :
- [ ] Créer design system (Colors, Fonts)
- [ ] Implémenter GlassCard
- [ ] Créer TimelineView
- [ ] Créer EditorView
- [ ] Animations hero

Checklist de Démarrage (suite)

Jour 3 :
- [ ] Créer App Intents (Add, Search)
- [ ] Implémenter SentimentAnalyzer
- [ ] Implémenter KeywordExtractor
- [ ] Tester Writing Tools
- [ ] Créer StatsView avec Charts
- [ ] Configurer iCloud sync

Finitions :
- [ ] App Icon & Launch Screen
- [ ] Dark mode support
- [ ] Accessibility labels
- [ ] Privacy policy

Points d'Attention

Performance :
- LazyVStack pour longues listes
- Thumbnails pour images (pas full size)
- Pagination si 1000+ entrées

UX :
- Auto-save toujours (jamais perdre du contenu)
- Feedback visuel pour toutes les actions
- États de chargement clairs
- Messages d'erreur explicites

Code Quality :
- Actors pour thread-safety
- async/await partout (pas de callbacks)
- Sendable pour tous les types partagés
- Tests unitaires pour logique critique

Outils de Développement

Xcode :
- Xcode 16+ (pour Swift 6 complete checking)
- Swift Playgrounds (pour tester concepts)
- Instruments (profiling performance)

Design :
- SF Symbols App (icônes)
- Figma (si besoin de mockups)

Testing :
- XCTest pour unit tests
- UI Testing pour flows critiques
- TestFlight pour beta testing

Version Control :
- Git + GitHub
- .gitignore pour Xcode

Ressources Générales

Documentation Apple :
- Apple Developer Documentation
- Swift.org

Communauté :
- Swift Forums
- r/iOSProgramming
- Stack Overflow

Newsletters :
- iOS Dev Weekly
- Swift Weekly Brief
- Hacking with Swift newsletter

Prochaines Étapes

Après l'atelier :

1. Continuer le développement de Mindful
2. Publier sur TestFlight (beta testing)
3. Itérer selon feedback
4. Soumission App Store

Apprentissage continu :

- Regarder WWDC sessions manquées
- Lire Swift Evolution proposals
- Contribuer open source
- Participer à des meetups iOS

Questions ?

Références Rapides

SwiftData : developer.apple.com/swiftdata

SwiftUI : developer.apple.com/swiftui

App Intents : developer.apple.com/app-intents

Natural Language : developer.apple.com/naturallanguage

WWDC : developer.apple.com/videos

Bon Courage ! 🚀

Ready to build Mindful?

Remember :

- Start simple, iterate often
- Test on real devices
- Privacy first
- Make it beautiful