MeetMind

Guide Architectural de Référence

Kickstart du Projet - 3 Parties

Table des Matières

Partie 1 : Backend Local de l'App
Partie 2 : Liquid Glass UI
Partie 3 : Apple Intelligence & Data

Partie 2

Liquid Glass UI

Design System - Palette de Couleurs

```swift
extension Color {
    // Gradients principaux (plus corporate)
    static let meetMindPrimary = Color(hex: "4F46E5") // Indigo
    static let meetMindSecondary = Color(hex: "7C3AED") // Purple

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

    // Meeting types
    static let standupColor = Color.blue
    static let oneOnOneColor = Color.green
    static let reviewColor = Color.orange
    static let planningColor = Color.purple

    // Priority
    static let priorityLow = Color.gray
    static let priorityMedium = Color.orange
    static let priorityHigh = Color.red
    static let priorityUrgent = Color(hex: "DC2626")
}
```

Design System - Typography

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

    // Body
    static let meetingNotes = Font.system(
        size: 16, weight: .regular
    )
    static let actionItem = Font.system(
        size: 15, weight: .medium
    )

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

Components à Créer

Core Components :

- `GlassCard` - Container de base
- `MeetingTypeSelector` - Picker de type
- `ParticipantPill` - Pills pour participants
- `ActionItemRow` - Row avec checkbox
- `DurationPicker` - Picker de durée

Complex Components :

- `MeetingCard` - Card pour timeline
- `NotesEditor` - Éditeur avec auto-save
- `ActionItemsList` - Liste avec completion
- `ParticipantPicker` - Sélection participants

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)
    }
}
```

MeetingTypeSelector - UI

```
[📱 Standup] [👥 1-on-1] [📊 Review] [📋 Planning] [🔄 Retro]
      ↑
Selection avec animation scale + color highlight
```

Features :

- Sélection tactile
- Animation spring au tap
- Couleur de fond selon type
- Durée par défaut appliquée

Views Architecture

```
MeetMindApp
└── ContentView
    ├── TimelineView (liste des meetings)
    │   └── MeetingCard (row)
    ├── EditorView (nouveau/édition meeting)
    │   ├── MeetingTypeSelector
    │   ├── TextEditor (notes)
    │   ├── ParticipantPicker
    │   └── ActionItemsList
    ├── DetailView (lecture meeting)
    │   ├── MeetingHeader
    │   ├── NotesContent
    │   └── ActionItemsList
    └── StatsView (analytics dashboard)
        └── MetricsCards
```

TimelineView - Structure

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

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

EditorView - Features

Composants :
- `TextEditor` natif SwiftUI (notes)
- `MeetingTypeSelector` en header
- `DatePicker` pour date/heure
- `DurationPicker` pour durée
- `ParticipantPicker` (multi-select)
- `ActionItemsList` (ajout/édition)
- Toolbar glass en bas

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

Writing Tools : Intégration automatique iOS 18+

ActionItemsList - Component

```
struct ActionItemsList: View {
    @Binding var items: [ActionItem]

    var body: some View {
        VStack(spacing: 12) {
            ForEach($items) { $item in
                ActionItemRow(item: $item)
            }

            Button("Add Action Item") {
                // Ajouter nouvel item
            }
        }
    }
}

struct ActionItemRow: View {
    @Binding var item: ActionItem

    var body: some View {
        HStack {
            Button { item.isCompleted.toggle() } label: {
                Image(systemName: item.isCompleted ?
                    "checkmark.circle.fill" : "circle")
            }
            TextField("Action item", text: $item.title)
            PriorityBadge(priority: item.priority)
        }
    }
}
```

Animations - Patterns

Hero Animation :

```swift
@Namespace private var animation
.matchedGeometryEffect(id: meeting.id, in: animation)
```

Completion Animation :

```swift
withAnimation(.spring(response: 0.3)) {
    actionItem.isCompleted.toggle()
}
```

Card Flip :

```swift
.rotation3DEffect(
    .degrees(isFlipped ? 180 : 0),
    axis: (x: 0, y: 1, z: 0)
)
```

Calendar Integration UI

```
struct CalendarSyncBanner: View {
    @State private var isSyncing = false

    var body: some View {
        HStack {
            Image(systemName: "calendar.badge.clock")
            Text("Sync with Calendar")
            Spacer()
            if isSyncing {
                ProgressView()
            } else {
                Button("Sync") {
                    Task { await syncCalendar() }
                }
            }
        }
        .padding()
        .background(.thinMaterial)
        .cornerRadius(12)
    }
}
```

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