LEARNIO

# Documentation technique

**Date mis à jour : 11/04/2023**

# Introduction

**LearnIO est une application développée avec Swift et SwiftUI, conçue pour aider les utilisateurs à apprendre et mémoriser de nouvelles informations de manière efficace. En utilisant LearnIO, vous pouvez créer des listes de cartes, chaque carte contenant des informations à apprendre, et les réviser à intervalles réguliers pour vous aider à retenir les informations sur le long terme. L'application utilise une méthode adaptative de révision des cartes, en fonction de la difficulté de chaque carte, et permet aux utilisateurs d'attribuer un état (Echec, Difficile, Bon, Facile) pour recalculer automatiquement le niveau d'apprentissage de chaque carte. De plus, LearnIO permet aux utilisateurs de suivre leur progression et de se concentrer sur les cartes les plus difficiles pour les aider à les maîtriser.**

**LearnIO est disponible sur iOS, mais également sur iPadOS et MacOS, offrant une expérience cohérente et fluide sur tous les appareils Apple. Avec son interface intuitive et facile à utiliser, LearnIO est l'application idéale pour ceux qui cherchent à améliorer leur apprentissage et leur mémorisation, quelle que soit leur plateforme de prédilection.**

***La version actuelle 1.0 ne supporte que iOS. La version 2.0 est en cours de développement est supporta IpadOS et MacOS .***

# Table des matières

# 1 Technologies utilisées

## 1.1 Swift

Swift est un langage de programmation moderne et puissant, développé par Apple, qui est utilisé pour créer des applications pour les plateformes iOS, iPadOS, macOS, watchOS et tvOS. Il est conçu pour être sûr, rapide et facile à utiliser, avec une syntaxe concise et expressive.

## 1.2 SwiftUI

SwiftUI est un framework de développement d'interfaces utilisateur déclaratives pour les applications Apple. Il permet aux développeurs de créer des interfaces utilisateur fluides et interactives pour iOS, iPadOS, macOS, watchOS et tvOS en utilisant le langage de programmation Swift.

## 1.3 XCode

Xcode est un environnement de développement intégré (IDE) pour les développeurs d'applications macOS, iOS, iPadOS, watchOS et tvOS. Il comprend un éditeur de code, un compilateur, des outils de débogage, des assistants de création d'interface utilisateur, des simulateurs d'appareils et bien plus encore. Xcode est utilisé pour développer des applications natives pour les appareils Apple.

# 2 Les sections/pages de l'application

## 2.1 Navbar

La navbar est situé en bas de l'écran de l'application. Elle contient 3 boutons : Accueil, Play, Créer

## 2.2 Accueil (Content View)

La page d'accueil affiche l'ensemble des listes. Ces listes sont triées automatiquement, en 2 catégories :

- Les listes à réviser (afficher le nombre de liste et le nombre de carte)
- Les autres listes

Les listes sont affichées sous forme de rectangles, avec le nombre de carte qu'elle contienne, le nom de la liste, ainsi que le nombre de jour avant de devoir réviser la liste à nouveau.

Si un utilisateur clique sur la liste, cela le redirigera vers la page AfficherUneListeView().

## 2.3 Play (S'entrainer)

Le bouton Play lance une session de révision avec l'ensemble des cartes (toutes listes confondues sans prendre en compte la nécessité les réviser ou non).

## 2.4 CreerUneListeView

Cette page permet de créer une liste. Le formulaire demande, le nom de la liste. Une fois la liste créer, il faut rediriger l'utilisateur vers la page AfficherUneListeView(la-liste-qui-vient-d-etre-créée).

## 2.5 AfficherUneListeView(liste : Liste)

Cette page permet d'afficher les informations associées à une Liste. Cette page affichera également 2 boutons, qui permettra de modifier la liste et supprimer la liste.

## 2.6 ModifierUneListeView(liste : Liste)

Cette page permet de modifier les informations associées à une Liste.

## 2.7 ReviserUneListeView(liste : Liste)

Cette page permet de faire une session de révision pour une liste passée en paramètre. Les cartes qui seront affichées seront celles qui doivent être réviser (toutes les cartes dont la formule de *TempsDeRetard* est inférieure ou égale à 0) OU celles dont le score est inférieur ou égal à 3.

## 2.8 ModifierUneCarteView(carte : Carte)

Cette page permet de modifier les informations associées à une carte.

# 3 Les modèles/entités

## 3.1 Carte

### 3.1.1 Définition

Une carte contient :

- Un recto et un verso
- Un tableau : l'ensemble des sessions de révision de cette carte (date et l'état associé par l'utilisateur)
- La date de la prochaine révision à faire (cf : formules DateProchaineRévision 3.1.1)
- La date du dernier malus : la dernière fois ou des points ont été retiré car il y avait du retard. -> initialisation à dateDuJour
- Un niveau : calculé automatiquement (grâce à un score invisible) au lancement de l'application et à chaque choix d'état lié à la carte par l'utilisateur. (cf : Niveau d'une carte 2.1.3)
- Un Score : varie dynamiquement en fonction des actions de l'utilisateur (cf : Score d'une carte 3.1.4)

### 3.1.2 Etat d'une carte

L'état de la carte est choisi durant la session de révision, lors du passage de la carte. Cet état est choisi par l'utilisateur : Echec, Difficile, Bon, Facile. Cet état est stocké dans le tableau des sessions sur cette carte associant l'état et la date.

### 3.1.3 Niveau d'une carte

A chaque attribution d'un état sur une carte par un utilisateur, il faut recalculer le niveau de la carte qui lui est estimé automatiquement par des lettres (G F E D C B A).

Le niveau d'apprentissage est attribué à chaque carte grâce à un score invisible.

Fréquence de révision en fonction du niveau :

| Niveau associé | Fréquence |
|---|---|
| G | +1 jour après dernière session |
| F | +3 jours après dernière session |
| E | +6 jours après dernière session |
| D | +10 jours après dernière session |
| C | +20 jours après dernière session |
| B | +40 jours après dernière session |
| A | +80 jours après dernière session |

### 3.1.4 Score d'une carte

**Le score varie en fonction du temps écoulé entre la dernière session et la session actuelle. Ainsi qu'à chaque choix d'un état lors d'une session de révision sur une carte. (cf : fonctions recalculerScoreCarte 3.2.1)**

| Niveau associé | Score |
|---|---|
| G | [ 0 ; 5 ] |
| F | [ 6 ; 12 ] |
| E | [ 12 ; 18 ] |
| D | [ 18 ; 24 ] |
| C | [ 24 ; 30 ] |
| B | [ 30 ; 36 ] |
| A | [ 37 ; 42 [ |

**+/- de point lié au niveau que l'utilisateur attribue à la carte :**

| Point +/- | Action |
|---|---|
| +2 | Attribuer « Bon » à la carte |
| +3 | Attribuer « Facile » à la carte |
| -3 | Attribuer « Difficile à la carte » |
| -6 (retour au niveau précédent) | Attribuer « Echec à la carte » |

**A chaque lancement de l'application, le score de chaque carte doit être mis à jour. Pour chaque carte de chaque liste il faut recalculer le score de la carte (recalculerScoreCarte(carte : Carte)).**

# 4 Les formules et fonctions

## 4.1 Formules

### 4.1.1 Date à laquelle la carte doit être révisée : DateProchaineRévision

```
DateProchaineRévision = DateDuJour + NiveauAssocié(Fréquence)
```

**DateProchaineRévision : Date à laquelle la carte doit être révisée**

**Cette formule est utilisée après chaque passage de la carte lors d'une session de révision.**

### 4.1.2 Temps de retard

```
TempsDeRetard = DateProchaineRévision - DateDuJour
```

**TempsDeRetard: temps entre la date où la carte doit être réviser et la date du jour.**

### 4.1.3 Temps entre dernier malus et aujourd'hui

```
TempsEntreDernierMalusEtAujourdhui = DateDernierMalus - DateDuJour
```

**TempsEntreDernierMalusEtAujourdhui: temps entre la date du dernier malus (suspension de point(s)) et la date du jour. Si l'écart est inférieur à 0, alors il faut suspendre des points. Chaque jour de retard, -1 point sur la carte. La suspension sur le score n'est pas effectuée ici (cf fonctions : Recalculer le score d'une carte 3.2.1).**

## 4.2 Fonctions

### 4.2.1 Recalculer le score d'une carte

```
fonction recalculerScoreCarte(carte : Carte) {
    TempsDeRetard = carte->DateProchaineRévision – DateDuJour
     Si (TempsDeRetard < 0) // alors il y du retard
        TempsEntreDernierMalusEtAujourdhui = DateDernierMalus – DateDuJour
        Si(TempsEntreDernierMalusEtAujourdhui<0) // alors il faut suspendre des points
          carte->score = carte->score + TempsEntreDernierMalusEtAujourdhui(négatif)
        Fin Si
      Fin Si
Fin
```