POLYTECH MONTPELLIER

FESTIVAL DU JEU

ARCHITECTURE APPLICATION MOBILE

DELOIRE Alexandre
JORGE Rémi

EDITION 2023-2024

Introduction

Cette application mobile a pour but de permettre aux bénévoles du Festival du Jeu de Montpellier de :

- Consulter et s'inscrire aux différents postes.
- Consulter et s'inscrire aux différentes zones bénévoles.
- Être flexible.
- Gérer l'ensemble de leurs inscriptions (s'ils souhaitent se désinscrire en masse, etc.).
- Consulter leur planning.
- Consulter les jeux pour chaque zone bénévole.
- Consulter les référents pour les postes.
- Gérer leurs messages.
- Répondre aux notifications.
- Gérer leur profil.
- Supprimer toutes ses données (conformité au RGPD)

Technologies

Cette application a été conçue en Swift, principalement sur Playground, et aucun module externe n'a été utilisé.

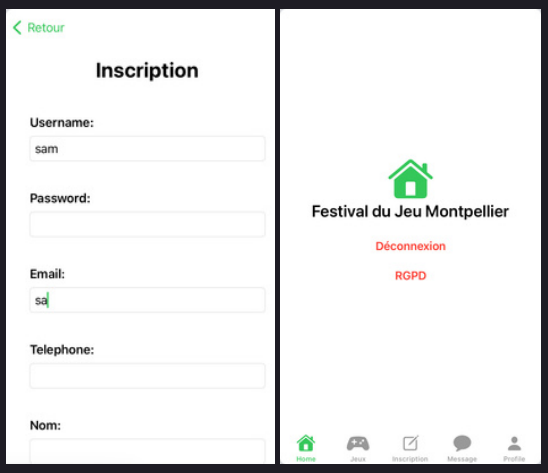
L'application mobile est liée à un backend d'où elle tire toutes ses données. Il existe également une application web avec des fonctionnalités analogues.

En ce qui concerne le backend, nous ne le présentons pas ici.

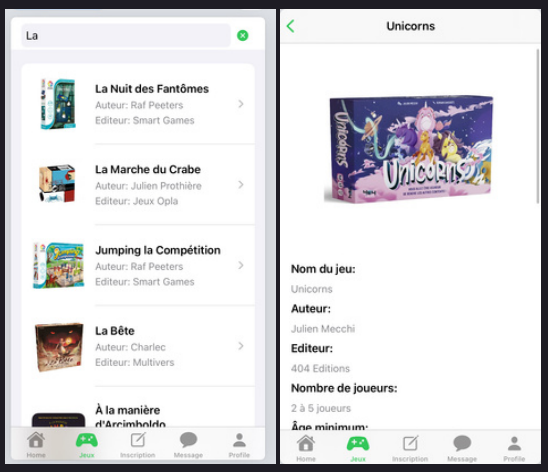
Architecture Globale

Nous allons présenter l'architecture globale de l'application. L'application comporte une barre de navigation située en bas, qui est une TabView. Chaque TabItem est en réalité une NavigationStack englobant une ou plusieurs fonctionnalités de l'application. À l'intérieur de chaque NavigationStack, la navigation est gérée à l'aide d'un routeur. Il est à noter que lorsque l'utilisateur n'est pas connecté, il ne peut accéder qu'à la page d'accueil (HomePage). Une fois connecté, les autres éléments de la barre de navigation deviennent accessibles.

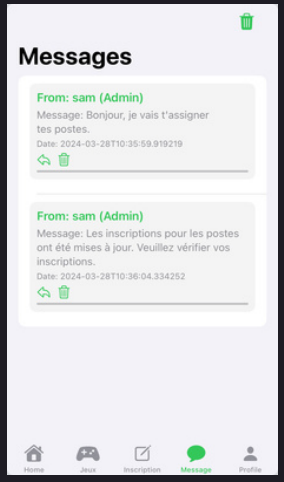
Voici les différentes pages de l'application :



HomePage :
Cette page regroupe plusieurs fonctionnalités. Lorsque l'utilisateur n'est pas connecté, il ne peut accéder qu'à cette page. Par conséquent, deux boutons seront disponibles sur cette page : un pour rediriger vers la vue de connexion (Login View) et un autre pour accéder à la vue de création de compte. Une fois que l'utilisateur se connecte, ces boutons disparaissent et un bouton de déconnexion apparaît à leur place. Il peut également décider, à partir de la page d'accueil, de supprimer toutes ses données personnelles, car l'application est conforme au RGPD.

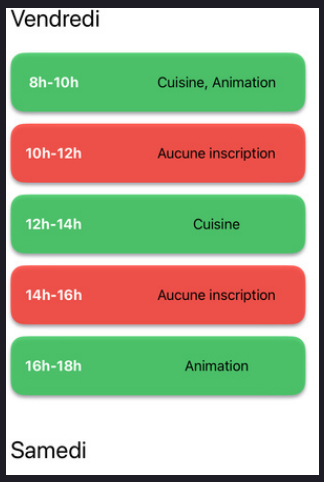


Jeux :
Cette page permet à l'utilisateur de consulter tous les jeux disponibles et de rechercher si un jeu spécifique existe. S'il existe, l'utilisateur peut cliquer dessus pour en voir les détails, tels que la zone bénévole à laquelle il appartient, s'il en appartient à une.



Messages :
Cette page regroupe les notifications et les messages. Chaque utilisateur pourra consulter ses notifications et ses messages, les supprimer s'il le souhaite, et également y répondre s'il le souhaite.

Inscription :
Cette page regroupe plusieurs fonctionnalités. Tout d'abord, lorsqu'un utilisateur y accède, il consulte son planning, où il est inscrit pour chaque créneau de chaque jour. Il peut également voir s'il est flexible sur un créneau. L'emploi du temps est présenté de manière très conviviale et esthétique pour l'utilisateur.

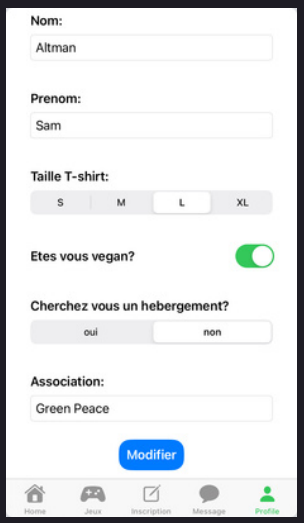


Inscription aux postes :
Pour s'inscrire ou modifier ses inscriptions, c'est très simple : l'utilisateur clique simplement sur le créneau qu'il souhaite, et une liste des postes s'affiche avec ses inscriptions ainsi que le nombre d'inscrits à chaque poste. Pour chaque poste, il peut appuyer sur "Voir plus" pour afficher les descriptions et les référents pour le poste.

Pour s'inscrire ou se désinscrire, nous avons créé une interface très simple et conviviale, une sorte de "sandbox" où l'utilisateur a simplement à sélectionner ou désélectionner les postes qu'il souhaite, puis appuyer sur "Enregistrer" pour s'inscrire.

Inscription aux zones bénévoles :
Lorsque l'administrateur aura inséré les jeux dans la base de données et si l'utilisateur a sélectionné le poste Animation, il sera redirigé vers une page qui l'obligera à s'inscrire à une et une seule zone bénévole. Il pourra également cliquer sur "Voir Jeux" et consulter à nouveau la liste des jeux pour la zone bénévole.

Pour gérer les inscriptions aux zones bénévoles, le processus est exactement le même pour y accéder.



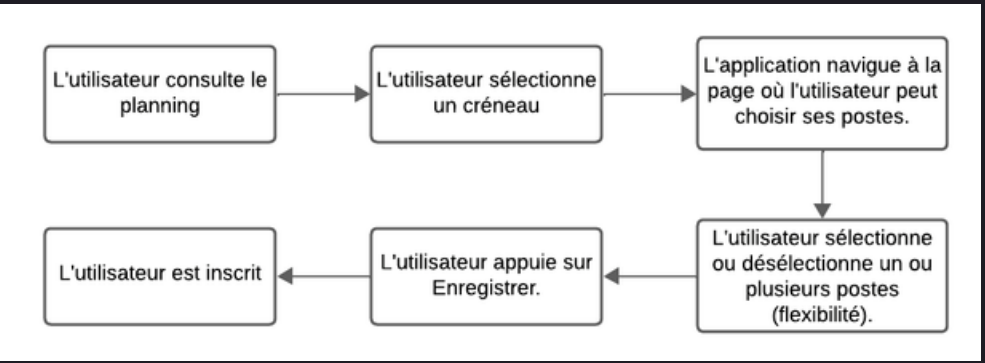
Profil :
Cette page regroupe les informations de profil. L'utilisateur peut mettre à jour ses informations d'une manière très simple.

Exemple de Flow

Nous allons illustrer l'architecture pour l'inscription aux postes. Notez bien que nous traitons ici seulement le cas des inscriptions aux postes et non aux zones bénévoles, afin de simplifier le schéma. Toutefois, l'inscription aux zones bénévoles suit la même logique. Nous nous sommes inspirés du design pattern MVVM ; nous avons choisi d'implémenter un VVM (View-ViewModel) car rajouter un modèle à part serait inutile, les actions que nous effectuons étant très simples. En effet, en consultant divers tutoriels sur internet, on remarque que la plupart des implémentations de MVVM se limitent souvent à un VVM, sans toujours le mentionner. En évitant d'ajouter un modèle supplémentaire, qui alourdirait l'architecture, nous adhérons au principe du KIS (Keep It Simple), ce qui améliore la maintenabilité du code.

Nous avons également décidé de découper les différents composants de l'affichage et de créer des composants réutilisables pour améliorer la maintenabilité et permettre une personnalisation plus poussée de l'UI et de l'UX.

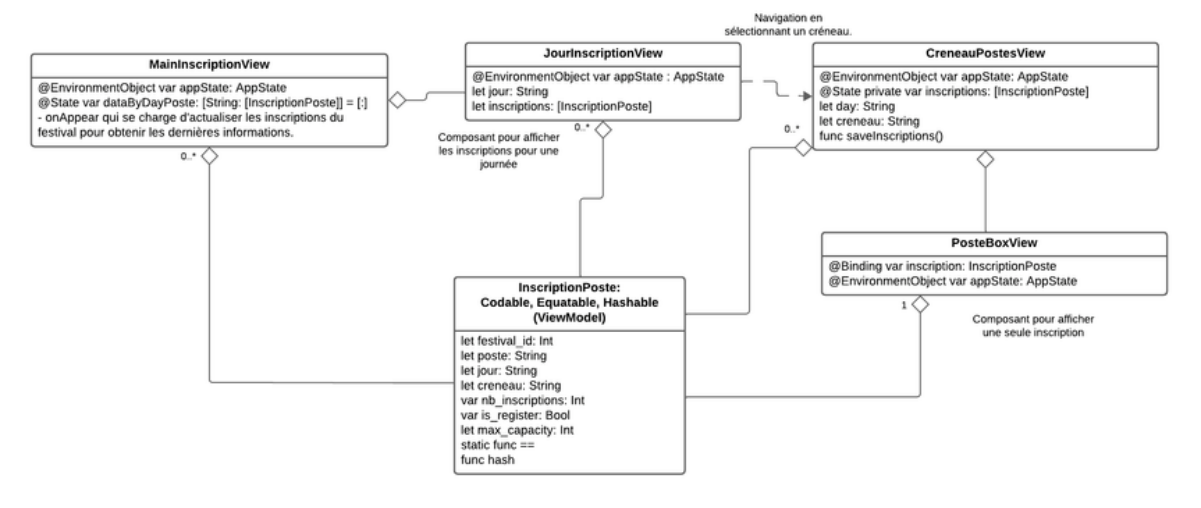
Pour le contexte, voici un déroulé des actions que l'utilisateur effectue :



Nous avons implémenté une interface "sandbox" pour l'utilisateur, c'est-à-dire que le bénévole peut sélectionner et désélectionner autant de postes qu'il souhaite, et quand il a fini, il suffit d'appuyer sur "enregistrer" pour confirmer les inscriptions.

Nous avons aussi mis en place un routeur qui nous permet de naviguer à travers l'application.

Voici comment nous avons agencé nos classes et structs :



Utilisation de struct

Nous avons utilisé une struct pour InscriptionPoste, car après avoir étudié la question, il était plus pertinent d'utiliser une structure plutôt qu'une classe, pour plusieurs raisons :

- Nous n'avons pas besoin d'héritage.
- Les structures sont plus efficaces à utiliser car elles ne sont pas stockées sur le tas (heap), et donc il n'est pas nécessaire d'utiliser un garbage collector.
- Pas de risques de fuite de mémoire.
- Moins de risques de bugs.
- Plus de simplicité.

Ainsi, nous utilisons des propriétés annotées avec @State et des liaisons (@Binding) dans notre application.

Sources de recherche personnelles :

Apple:
https://developer.apple.com/videos/wwdc2015

Microsoft:
https://learn.microsoft.com/en-us/dotnet/standard/design-guidelines/choosing-between-class-and-struct?redirectedfrom=MSDN

Article:
https://medium.com/macoclock/swift-struct-vs-class-performance-29b7be73d9fd

Référence du cours d'IG3:
https://airspeedvelocity.net/2015/07/26/linked-lists-enums-value-types-and-identity/