Compte-rendu 14 : réunion du 28/01/2025

SIForms : Application de saisie de formulaires

Auteurs :
Adèle DESMAZIERES
Léa EL ABBOUD
Kevin XU

Destinataires :
Paul GUYOT
Julien KERLIDOU
Cédric BESSE
Lamia LARAQUI

Responsable : **Adèle DESMAZIERES**

Type : **Compte-rendu 14 : réunion du 28/01/2025**

Date : **28/01/2025**

Nombre de pages : **1**

Statut : **Final**

Adresse : typst.app/PISTL

Version	Date	Modification	Rédacteur
v.0.1	28/01/2025	Rédaction du CR	Adèle DESMAZIERES

Ordre du jour

1. Application mobile

Présentation de notre avancement :

- Connexion http qui marche (problème était lié à l'émulation, donc ne pas trop perdre de temps dessus)
- Changement de design du choix de langue
- Bug : email case sensitive, c'est un bug à régler du côté backend, donc on ne s'en occupe pas car ce n'est pas prioritaire du tout.

Mode offline : Il faut pouvoir utiliser l'application quand même en offline. Si on a aucun formulaire préchargé ni aucun modèle, on ne peut rien faire donc afficher la page sans wifi. Sinon, faire une snackbar dans l'application pour mentionner qu'on est offline, mais pas une page entière qui bloque l'utilisation. La page « offline » est affichée si on ouvre l'app sans wifi, du coup on ne peut pas se connecter mais on peut accéder à nos propres formulaires en local. Offline on ne peut pas changer son mdp ni sa langue, mais on peut remplir des formulaires et les soumettre en asynchrone, même si on perd la connexion pendant l'utilisation.

Page des formulaires : Afficher la liste des formulaires, mettre un bouchon pour le moment. Nos formulaires sont enregistrés en local. A la connexion, récupérer les formulaires du user.

2. Serveur

Edition du profil dans le serveur.

3. Rendering des formulaires

Bibliothèque solution : eclipsesource/jsonforms, jsonform.io

Cette bib gère déjà la partie validation du schema et a déjà des UI (voir le site).

Il faut brancher le renderer de notre choix sur cette bib. Elle en propose des exemple sur github, mais aucun n'est compatible avec React Native. Donc aller voir le vanilla renderer dans `src/renderer.ts` et l'éditer.

Exemple pour TextCell : balises html dans </input ... > à remplacer par des composants ReactNative. Partir de vanilla renderer pour faire un React Native Renderer. Ecrire un minimum d'items nécessaires aux formulaires de Smart Impulse (par exemple, pas besoin de sliders). Voir la bibliothèque AndroConsis/react_native_json_schema_form pour s'inspirer de leur renderer, mais à modifier car elle est trop vieille.

Evaluation de la difficulté :

- Bouton : facile.
- Layout : plus dur, voir la bib de AndroConsis.

Plannification : Adèle mentionne que si pas le temps de faire ça, on fera un bouchon.

4. Plannification

Prochaine réunion mardi 04/02/2025 à 10h.