Compte-rendu 7 : réunion du 29/11/2024

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 7 : réunion du 29/11/2024**

Date : **29/11/2024**

Nombre de pages : **2**

Statut : **Final**

Adresse : typst.app/PISTL

Version	Date	Modification	Rédacteur
v.0.1	29/11/2024	Rédaction du CR	Adèle DESMAZIERES

Ordre du jour

1. Présentation de notre STR

1.1. Objets métier

Entrée de formulaire : juste les réponses. Pas les questions.

One to many : référence document vers formulaire. Mettre les attributs côté enfant.

Modèle de formulaire : identifiant de nom « name » pour rassembler les version d'un même modèle ensemble.

Django : permet de mettre les relations de tous les côtés

Cohérence : retirer les attributs supplémentaires, il faut que ca soit cohérent d'une classe à l'autre. Ajouter data pour Form et name pour Model.

Photo : mettre attribut URL. Ajouter le statut d'envoie des photos. Renommer Photo en « Document ». Côté serveur : stocker le chemin vers le fichier. Côté mobile : identifiant de la photo dans l'appareil.

Attributs : manque logique business sur l'envoi des informations, l'ajouter sur l'envoi des documents. Ajouter des dates. Date de création d'une instance de formulaire, date d'envoi, etc.

Y a-t-il vraiment une différence entre le serveur et la base de données ? Rassembler ces deux sous-systèmes en un seul ?

Sous-systèmes : séparer l'application en plusieurs sous-systèmes. Application mobile : pas forcément les mêmes objets métiers, par exemple pas besoin de savoir à qui sont les choses. De plus le serveur n'a pas besoin de savoir si le formulaire est envoyé.

Puis faire CRUDE pour Routes API REST. Flask / Django plugin ? Django REST Framework. Le CRUDE permet de déterminer les routes à mettre sur les objets métiers.

En général un objet Session est utilisée dans l'API REST. Le rajouter ici.

Bien différencier ce qui est données de la base de données, et données locales de l'application.

JWT ?

(Chaque form doit avoir un lien URL qui expire ?) Y réfléchir, même si ce n'est pas la priorité. Générer le lien à partir de l'id du formulaire par exemple.

Diagramme de séquence QRCode : faut-il un objet intermédiaire « Formulaire pré-rempli » entre le formulaire et le modèle ?

1.2. Déploiement

Pas besoin de mettre à coté les doc non-relationnel, directement le mettre dans PostgreSQL : JSONbinary. Photos dans un fichier à côté du serveur.

1.3. Tests d'intégration

C'est le cahier de recettes, très important, vérifier par le client à la fin.

Pas besoin de diag de séquence. Plutot scénario de tests. Utiliser des instanciations de composants. Nous on a une IHM, donc utiliser le retour de l'IHM plutot que les diag de séquence ? Diag de séquence pas logique pas rapport à l'envoi du formulaire.

TYPO : envoi pas envoie dans diag.

Tests importants : pas le temps, mais si connexion coupée ou instable. Soumission en plusieurs étapes : envoi de la réponse puis des photos.

Envoi des photos séparément.

Interactions app/serveur : Si le serveur refuse des entrées après un envoi asynchrone, que se passe-t-il ? Définir les contraintes : réponses obligatoires, types, taille de la réponse. Cela est décrit dans la grammaire du modèle et est vérifié dans l'application. Il est possible que le serveur refuse quand même la réponse, dans ce cas on doit réfléchir à ce qu'il se passe.

2. Planning

Cette réunion a dépassé les 30 minutes prévues, donc on prévoit plutôt 1h pour la prochaine. Elle sera le mardi 10 décembre de 9h à 10h.