

STBE

SIForms : Application de saisie de formulaires

Auteurs :

Adèle DESMAZIERES

Léa EL ABOUD

Kevin XU

Destinataires :

Paul GUYOT

Julien KERLIDOU

Cédric BESSE

Lamia LARAQUI

Responsable : **Adèle DESMAZIERES**

Type : **STBE**

Date : **30/01/2025**

Nombre de pages : **38**

Statut : **Initial**

Adresse : typst.app/PISTL

Historique

Version	Date	Modification	Rédacteur
v.0.1	11/10/2024	Génération du modèle de plan	Adèle DESMAZIERES
v.0.2	12/10/2024	Rédaction de la définition du projet	Kevin XU
v.0.3	15/10/2024	Reformulation et corrections de typo	Adèle DESMAZIERES
v.0.4	16/10/2024	Rédaction des use case du technicien	Kevin XU
v.0.5	19/10/2024	Diagrammes des use case front et back	Adèle DESMAZIERES
v.0.6	22/10/2024	Contraintes et risques	Léa EL ABOUD
v.0.7	22/10/2024	Comparaison des technologies	Adèle DESMAZIERES
v.0.7	23/10/2024	Diagrammes de séquences	Adèle DESMAZIERES
v.0.8	25/10/2024	Diagramme du frontend mis à jour	Adèle DESMAZIERES
v.0.9	31/10/2024	Diagramme du WBS	Kevin XU
v.0.10	01/11/2024	Mise à jour des use case front	Léa EL ABOUD
v.0.11	02/11/2024	Diagramme de PERT	Adèle DESMAZIERES
v.0.12	02/11/2024	Mise à jour des diagrammes de séquences	Léa EL ABOUD
v.1.1	04/11/2024	Diagramme de Gantt	Adèle DESMAZIERES
v.1.2	04/11/2024	Présentation de la maquette	Léa EL ABOUD
v.1.3	21/11/2024	Modification de la maquette	Léa EL ABOUD
v.1.4	22/11/2024	Corrections mineurs (lexique, typos, bibliographie, Gantt)	Adèle DESMAZIERES
v.1.5	23/11/2024	Mise à jour du diagramme de PERT	Adèle DESMAZIERES
v.1.6	27/11/2024	Lexique	Adèle DESMAZIERES
v.1.7	17/01/2025	Gantt mis à jour	Adèle DESMAZIERES
v.1.8	30/01/2025	Bibliographie mise à jour	Adèle DESMAZIERES

Table des matières

1. Définition du projet	3
1.1. Objectifs du projet	3
1.2. Présentation de Smart Impulse	4
1.3. Périmètre de SIForms	4
1.4. Interactions entre SIForms et Smart Impulse	5
2. Architecture fonctionnelle du système	5
2.1. Découpe en composants	5
2.2. Choix des technologies	6
2.2.1. Solution pour le frontend	6
2.2.2. Technologie de l'application mobile	6
2.2.3. Technologie du serveur	6
2.2.4. Technologie des données	7
2.2.5. Format des entrées des formulaires	7
2.3. Synthèse des choix des technologies	7
2.4. Sous-systèmes	8
2.4.1. Application mobile pour le frontend	8
2.4.2. Serveur pour le backend	20

2.5. Interactions entre les systèmes	21
2.5.1. Inscription	21
2.5.2. Authentification	22
2.5.3.	23
2.6. Interactions entre les cas d'utilisation	23
3. Contraintes et risques	23
3.1. Les types de contraintes	23
3.1.1. Contraintes fonctionnelles	23
3.1.2. Performances et planification	24
3.1.3. Ressources	24
3.1.4. Techniques	24
3.1.5. Mise en exploitation de SIForms	24
3.2. Analyse des risques	24
3.2.1. Risques liés aux contraintes fonctionnelles	24
3.2.2. Risques liés aux performances	25
3.2.3. Risques liés aux ressources	25
3.2.4. Risques liés aux contraintes techniques	25
3.2.5. Risques liés aux contraintes procédurales	26
4. Prototypage	26
4.1. Maquette sur Figma	26
5. Planification	32
5.1. Work Breakdown Structure (WBS)	32
5.2. Tâches	32
5.2.1. Conception	33
5.2.2. Réalisation	33
5.2.3. Tests	33
5.2.4. Déploiement	34
5.2.5. Gestion	34
5.3. Program Evaluation and Review Technique (PERT)	35
5.4. Diagramme de Gantt	36
6. Annexes	37
6.1. Lexique	37
6.1.1. Termes fonctionnels	37
6.1.2. Termes techniques	37
6.1.3. Technologies	37
6.2. Bibliographie	38
6.2.1. Sources pour la comparaison des technologies	38
6.2.2. Documentation des technologies	38

1. Définition du projet

Le projet vise à développer une application mobile **SiForms**, permettant à des techniciens électriques de remplir des formulaires dynamiques et de les soumettre de manière asynchrone. L'outil doit simplifier la saisie d'informations par les techniciens, assurer le transfert sécurisé des données en asynchrone, et simplifier la gestion des comptes utilisateurs.

1.1. Objectifs du projet

Les formulaires permettent d'informer les équipes de Smart Impulse sur les caractéristiques des sites d'intervention et sur leurs appareils électriques. L'utilisateur doit pouvoir saisir les informations nécessaires pour y répondre, et y attacher des photos. Il peut enregistrer un formulaire en cours de remplissage, reprendre son édition plus tard, et l'envoyer une fois complété. Il peut également consulter l'état de ses formulaires, et consulter ses réponses précédentes. De plus, les formulaires peuvent être traduits dans l'application.

La pose du compteur Smart Impulse se fait souvent dans des lieux sans connexion Internet. Or la solution actuelle est un ensemble de questionnaires Jotform qui ne permettent pas l'envoi asynchrone. C'est pourquoi l'outil développé doit permettre de remplir et soumettre l'envoi des formulaires hors-ligne, et délayer l'envoi jusqu'au rétablissement de la connexion Internet.

Les réponses sont envoyées à un serveur qui les stocke et les exploite pour générer automatiquement un compte-rendu d'intervention. L'outil doit garantir la sécurité des données utilisateur surtout lors du transit et éventuellement lors du stockage.

Enfin, une extension à faire s'il reste du temps consiste à analyser les photos transmises par les techniciens pour vérifier leur pertinence avant transfert et stockage, et pour vérifier la pose correcte du compteur afin d'automatiser une procédure manuelle.

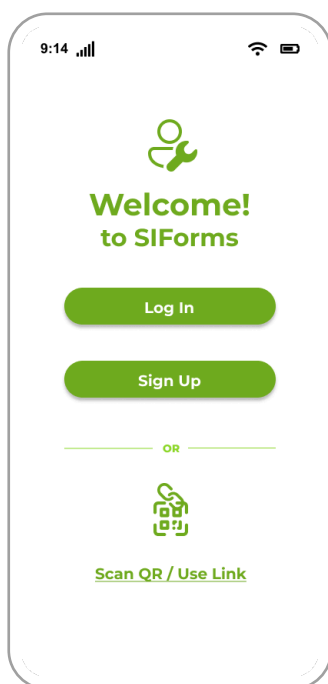


Fig. 1. Écran d'accueil de la maquette de l'application. Les images de maquette présentes dans ce document ne sont pas contractuelles, mais servent à donner une idée du produit final au lecteur.

1.2. Présentation de Smart Impulse

Smart Impulse est une entreprise spécialisée dans l'optimisation des consommations électriques des bâtiments tertiaires, en France et à l'étranger. C'est une ancienne startup devenue une entreprise de 80 employés.

Smart Impulse permet à des gestionnaires de bâtiments de visualiser et réduire leur consommation électrique. Pour cela, la consommation électrique des bâtiments est mesurée avec le Smart X, un compteur électrique installé chez le client, par un technicien partenaire de Smart Impulse ou par un technicien mandaté par le responsables du site. Puis, Smart Impulse propose des plans de d'économie d'énergie au gestionnaire des bâtiments.

Les techniciens réalisant les interventions doivent envoyer des informations techniques sur le site à Smart Impulse. La forte croissance des équipes de Smart Impulse a induit la nécessité d'automatiser ce processus, qui était auparavant au format papier puis avec un gestionnaire de formulaires externe.

L'outil de questionnaire actuel présente des limites, c'est pourquoi Smart Impulse souhaite développer une application de saisie de formulaires. Elle doit simplifier la collecte des réponses du côté de Smart Impulse, et fluidifier le processus de saisie et d'envoi pour les techniciens.



Fig. 2. Smart X

1.3. Périmètre de SIForms

L'outil à réaliser doit permettre aux techniciens de répondre à des formulaires dynamiques même hors-ligne, puis de les soumettre au serveur, c'est-à-dire lancer la recherche de connexion pour l'envoyer. Le transfert des données a effectivement lieu lorsque la connexion est rétablie, sans intervention supplémentaire de l'utilisateur.

Il existe pour le moment trois modèles de formulaires, et l'outil doit permettre l'ajout de nouveaux modèles. Les questions d'un formulaire ne doivent donc pas être codées en dur dans le logiciel. Les formulaires sont dynamiques, c'est-à-dire que la réponse à certaines questions conditionne l'affichage des questions suivantes, et certains groupes de questions peuvent être répétés un certain nombre de fois qui dépend des réponses précédentes.

L'application doit proposer de traduire les 1formulaires, grâce aux traductions fournies par Smart Impulse. La traduction de l'interface de l'application n'est pas requise, et sera donc développée en anglais pour permettre une meilleure accessibilité aux techniciens non-francophones.

Un formulaire est soit en cours de saisie de réponse, soit dans un des états suivants :

- **brouillon** si la réponse est partiellement saisie et enregistrée localement,
- **en attente de connexion** si la réponse a été soumise par l'utilisateur mais pas encore transférée par manque de connexion,

- **en cours d'envoi** si une partie de la réponse a été transférée au serveur mais pas complètement,
- **envoyé** si toute la réponse a été envoyée au serveur.

Les données doivent être transférées de manière sécurisée. Le serveur est une maquette temporaire, qui sera remplacée par le serveur de Smart Impulse après la livraison du projet. Donc, le stockage peut éventuellement être sécurisé, mais ce n'est pas essentiel au projet.

Lorsque le serveur reçoit une réponse, il stocke les documents attachés et le contenu de la réponse, et il génère un compte-rendu d'intervention au format PDF. Ce document récapitule l'intervention et est envoyé par mail au technicien, pour servir de trace.

Le premier accès à l'application par un technicien se fait par à un lien envoyé à l'équipe projet d'installation. La personne chargée de l'intervention technique clique sur le lien, télécharge l'application, crée un compte, puis accède au formulaire pré-rempli.

Hors de la portée du projet : Réaliser un éditeur de formulaires ne fait pas partie du projet. De plus, le serveur à réaliser est une mockup (maquette), qui sera à terme remplacé par le serveur de Smart Impulse.

Extensions : S'il reste du temps pour le projet, on pourrait analyser les photos transmises par traitement d'image, afin de vérifier la pertinence des photos pour économiser en transfert de données et en stockage. On pourrait également vérifier que le boîtier est correctement installé (c'est-à-dire avec sa flèche vers le bas) pour automatiser un travail actuellement manuel réalisé par les employés de Smart Impulse.

1.4. Interactions entre SIForms et Smart Impulse

SIForms interagira directement avec le serveur maquette de Smart Impulse. Lorsqu'un technicien reçoit un code d'accès ou un lien temporaire, il sera dirigé vers un formulaire pré-rempli, basé sur les informations spécifiques du site. Si l'application n'est pas installée, le lien renverra d'abord vers une page de téléchargement, puis vers le formulaire après installation. Une fois rempli, le formulaire sera synchronisé avec la base de données dès que l'appareil sera en ligne. Un PDF sera généré et envoyé à l'utilisateur ainsi qu'à Smart Impulse, avec un lien vers le dossier partagé contenant les documents et photos. L'outil gèrera également les soumissions automatiques en cas de rétablissement de la connexion, assurant ainsi un processus fluide même dans des environnements avec un accès Internet limité.

2. Architecture fonctionnelle du système

2.1. Découpe en composants

Nous avons isolé les composants suivants, nécessaires à la conception de l'outil :

- **Le composant frontend** d'application mobile, permettant à un technicien de saisir et envoyer ses réponses à des formulaires.
- **Le composant backend**, composé d'un serveur permettant d'envoyer des formulaires vierges ou pré-remplis et d'enregistrer les réponses dans une base de données accessible par Smart Impulse. Le composant backend est un composant « bouchon » qui sera remplacé à terme par un serveur de Smart Impulse.

2.2. Choix des technologies

2.2.1. Solution pour le frontend

Application mobile ✓	Progressive Web App ✗	Site web ✗
<ul style="list-style-type: none"> • permet l'accès hors-ligne • simplifie la soumission asynchrone • simplifie la mise à disposition des formulaires précédents 	<ul style="list-style-type: none"> • rien à installer pour les techniciens • fonctionne sur n'importe quelle plateforme avec un navigateur • permet de consulter des formulaires hors-ligne grâce au cache • envoi des données en asynchrone compliqué 	<ul style="list-style-type: none"> • rien à installer pour les techniciens • fonctionne sur n'importe quelle plateforme avec un navigateur • envoi des données en asynchrone compliqué • pas d'accès hors ligne

Pour le composant frontend, nous avons donc choisi de développer une application mobile, qui simplifiera la gestion de l'historique des réponses et l'envoi asynchrone.

2.2.2. Technologie de l'application mobile

Contraintes : L'application doit être cross-plateforme pour Android et iOS. Nous avons donc ignoré le développement natif, pour ne pas devoir programmer deux l'application avec les langages natifs.

ReactNative ✓	Flutter ✗
<ul style="list-style-type: none"> • langages : React et JavaScript • beaucoup de bibliothèques existantes • grande communauté et plein de tutoriels • moins performant mais probablement suffisant • déjà utilisé chez Smart Impulse • on souhaite se former à cette technologie • gestion compliquée des dépendances mais le module expo simplifie cela 	<ul style="list-style-type: none"> • langage : Dart • plus performant • moins de ressources car communauté moins développée mais en croissance • verbeux quand on souhaite rentrer dans les détails

2.2.3. Technologie du serveur

Contraintes : Serveur en Python imposé par Smart Impulse.

Django ✓	Flask ✗	Ruby on Rails ✗
<ul style="list-style-type: none"> framework adapté aux gros projets meilleure sécurité meilleure maintenance format pour les formulaires utilisé à Smart Impulse adapté à PostgreSQL Paul Guyot aimerait réduire l'utilisation de Django à Smart Impulse mais nous l'a quand même conseillé 	<ul style="list-style-type: none"> framework minimaliste prise en main plus rapide environnement flexible (bibliothèques) plus performant moins sécurisé 	<ul style="list-style-type: none"> facile à développer bon environnement test difficile de créer une API avec manque de documentation

2.2.4. Technologie des données

Données relationnelles : PostgreSQL

Nous avons sélectionné PostgreSQL car c'est une base de données (BD) relationnelle déjà utilisée à Smart Impulse, et recommandée par les professionnels. De plus, Django communique très bien avec cette BD.

Données brutes : enregistrées sur serveur local

Les données brutes (contenu des formulaires et documents attachés) seront enregistrées en dur dans le serveur. En cas d'avancement suffisant du projet, nous envisageons une migration des données dans un cloud (comme AWS), offrant une plus grande place de stockage.

2.2.5. Format des entrées des formulaires

JSON ✓	XML ✗	YAML ✗
<ul style="list-style-type: none"> utilisé actuellement par Smart Impulse très répandu lisible par un humain rapide pour requêter un champ spécifique souvent utilisé pour l'échange de données 	<ul style="list-style-type: none"> très répandu souvent utilisé pour l'échange de données très verbeux syntaxe complexe 	<ul style="list-style-type: none"> très lisible par un humain plus souvent utilisé pour de la configuration

Le format des modèles de formulaires est imposé par Smart Impulse comme étant en JSON Schema légèrement modifié pour s'accorder à leurs besoins.

2.3. Synthèse des choix des technologies

- **Application mobile** : ReactNative 0.75 (émulateur : Android Studio)
- **Serveur backend** :
 - Python 3.12
 - Django 4.2
- **Base de données** : PostgreSQL 16
- **Données brutes** :
 - Modèles de formulaire (JSON Schema)

- Entrées de formulaires (JSON)
- Documents attachés (PNG, JPEG/JPG, BMP...)

2.4. Sous-systèmes

2.4.1. Application mobile pour le frontend

2.4.1.1. Architecture et diagramme des cas d'utilisation

Le technicien doit pouvoir créer un compte et s'authentifier à son compte. Il doit pouvoir accéder à des formulaires pré-remplis pour lui transmis par Smart Impulse, mais aussi à des formulaires vierges. Le technicien peut saisir ses réponses dans le formulaire et y attacher des documents photo. Il peut supprimer sa réponse, l'enregistrer en brouillon, ou lancer son envoi. Il peut également sélectionner la langue de traduction des formulaires.

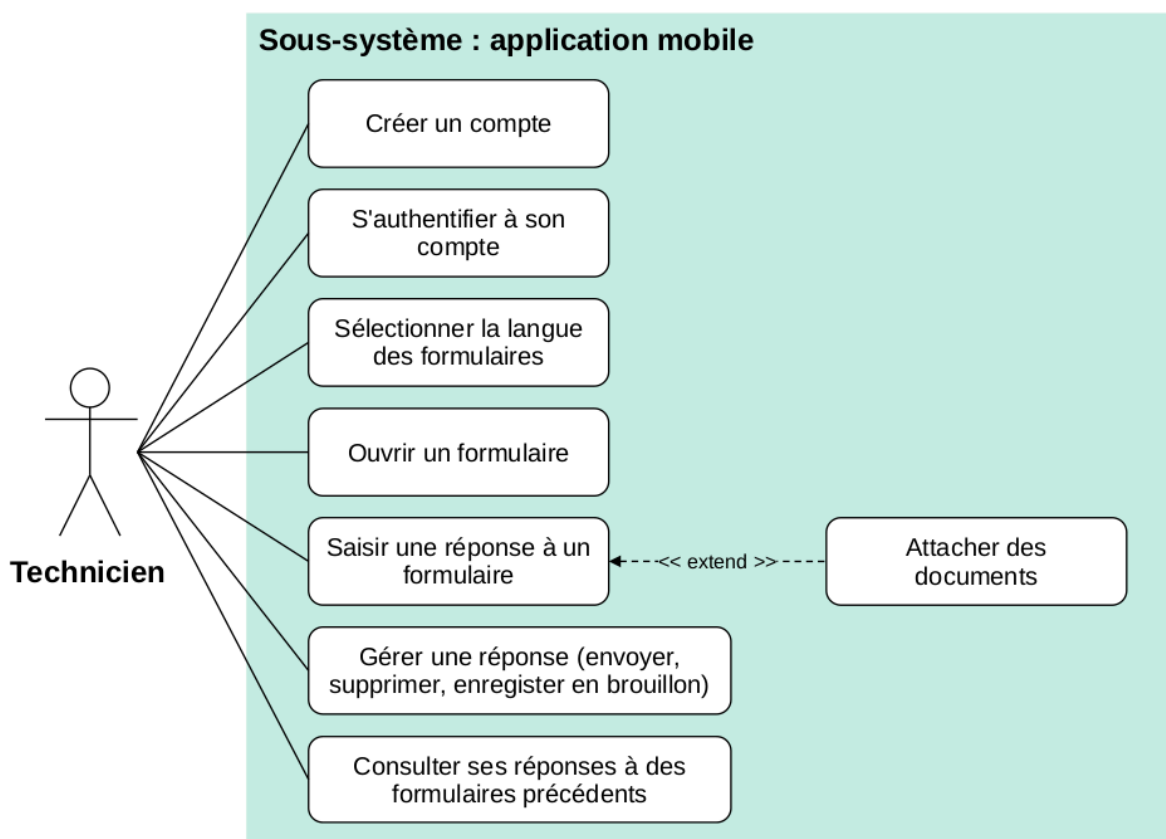


Fig. 3. Diagramme des cas d'utilisation du frontend.

2.4.1.2. Acteurs

Technicien

- **Type** : Humain
- **Rôle** : Chargé de l'installation électrique

2.4.1.2.1. Détail du cas d'utilisation : a. Créer un compte

- **Nom du cas d'utilisation** : a. Créer un compte
- **Acteur principal** : Technicien
- **Acteurs secondaires** : Aucun
- **Objectif** : Permettre à un nouvel utilisateur de créer un compte.
- **Pré-condition(s)** : L'application est installée et l'utilisateur dispose d'une connexion Internet.
- **Hypothèses** : Aucune

- **Scénario :**
 1. L'utilisateur ouvre l'application.
 2. Le système affiche la page d'accueil avec l'option « Log In » et « Sign Up ».
 3. L'utilisateur sélectionne l'option « Sign Up ».
 4. Le système affiche la page d'inscription.
 5. L'utilisateur saisit ses informations personnelles dans les champs : nom, prénom, email, mot de passe.
 6. Le système active le bouton « Create account » une fois que toutes les informations sont remplies.
 7. L'utilisateur clique sur « Create account »
 8. Le système envoie un email de vérification à l'utilisateur, contenant un lien de confirmation.
 9. L'utilisateur reçoit l'email et l'ouvre.
 10. L'utilisateur clique sur le lien dans l'email.
 11. Le système redirige l'utilisateur vers la page de la liste des formulaires, initialement vide.
- **Post-condition(s) :** Le compte de l'utilisateur est créé, un message de confirmation de création de compte est affiché.
- **Exigences non fonctionnelles :** Aucune
- **Exceptions, alternatives :**
 - **Exception 1 :** En SN7
Si l'email saisi est déjà associé à un compte existant, un message d'erreur s'affiche, invitant l'utilisateur à s'authentifier.
 - **Exception 3 :** En SN9
Si l'email de vérification n'arrive pas, l'utilisateur peut demander un renvoi de l'email de confirmation.
 - **Exception 2 :** En SN10
Si l'utilisateur ne clique pas sur le lien du mail de confirmation de création de compte avant une semaine, alors la procédure d'inscription est annulée.

Test de validation : Créer un compte – nominal

- **Nom du test :** a1 – Créer un compte – nominal
- **Use Case correspondant :** Créer un compte
- **Contexte :** L'application est connectée à Internet et l'utilisateur n'a pas de compte.
- **Pré-condition(s) :** L'application est installée, l'utilisateur est sur la page d'accueil et dispose d'une connexion Internet.
- **Entrée(s) :** Informations personnelles valides (nom, prénom, email, mot de passe).
- **Scénario :**
 1. Sélectionner « Sign Up ».
 2. Remplir les champs avec des informations personnelles valides.
 3. Cliquer sur « Create Account ».
 4. Ouvrir sa messagerie électronique.
 5. Cliquer sur le lien dans l'email envoyé par Smart Impulse.
- **Résultat attendu :** Le compte est créé.
- **Post-condition(s) :** L'utilisateur dispose d'un compte et peut s'y authentifier.
- **Moyen de vérification :** Le message de confirmation d'inscription s'affiche. Il est possible de s'authentifier avec les identifiants choisis.

Test de validation : Créer un compte – Compte déjà existant – exception 1

- **Nom du test** : a2 – Créer un compte – Compte déjà existant – exception 1
- **Use Case correspondant** : Créer un compte
- **Contexte** : L'application est connectée à Internet et l'utilisateur n'a pas de compte.
- **Pré-condition(s)** : L'application est installée, l'utilisateur est sur la page d'accueil et dispose d'une connexion Internet.
- **Entrée(s)** : Informations personnelles avec un email invalide (ex. : « email_error »).
- **Scénario** :
 1. Ouvrir l'application.
 2. Sélectionner « Sign Up ».
 3. Saisir l'email « email_error » dans le champ correspondant, et remplir les autres champs avec les informations personnelles.
 4. Cliquer sur « Submit ».
- **Résultat attendu** : Un message d'erreur apparaît, indiquant que l'email est invalide.
- **Post-condition(s)** : Le compte n'est pas créé. Aucun mail n'est envoyé.
- **Moyen de vérification** : Le message d'erreur s'affiche, aucun compte n'est créé, et il est impossible de s'authentifier à l'application avec ce compte.

2.4.1.2.2. Détail du cas d'utilisation : b. S'authentifier à son compte

- **Nom du cas d'utilisation** : b. S'authentifier à son compte
- **Acteur principal** : Technicien
- **Acteurs secondaires** : Aucun
- **Objectif** : Permettre à l'utilisateur de s'authentifier à son compte.
- **Pré-condition(s)** : Un compte utilisateur existe, l'utilisateur est sur la page d'accueil et dispose d'une connexion Internet.
- **Hypothèses** : Aucune
- **Scénario** :
 1. L'utilisateur choisit l'option « Log In ».
 2. Le système affiche la page d'authentification.
 3. L'utilisateur saisit son email et son mot de passe.
 4. Le système active le bouton « Log in » dès que les champs sont remplis.
 5. L'utilisateur appuie sur « Log in ».
 6. Le système vérifie les identifiants.
 7. Le système affiche la liste de ses formulaires.
- **Post-condition(s)** : L'utilisateur est authentifié et redirigé vers la page de la liste des formulaires.
- **Exigences non fonctionnelles** : Aucune
- **Exceptions, alternatives** :
 - **Exception 1** : En SN6
Si les identifiants sont incorrects, un message d'erreur s'affiche. L'utilisateur peut réessayer ou sélectionner la fonction « Forgot password ». Dans ce cas, il saisit son adresse email dans un champs dédié, clique sur « Reset password », puis reçoit un email contenant un lien permettant de réinitialiser son mot de passe. En cliquant sur ce lien depuis son email, il est renvoyé sur une page demandant la saisie d'un nouveau mot de passe et la saisie du même mot de passe, qui permet de changer de mot de passe après confirmation.

Test de validation : S'authentifier à son compte – nominal

- **Nom du test** : b1 – S'authentifier à son compte – nominal
- **Use Case correspondant** : S'authentifier à son compte
- **Contexte** : L'utilisateur a déjà un compte valide.
- **Pré-condition(s)** : Le compte utilisateur est déjà créé et activé, l'application est lancée avec une connexion Internet disponible, et l'utilisateur se trouve sur la page d'accueil.
- **Entrée(s)** : Identifiants valides (email et mot de passe).
- **Scénario** :
 1. Sélectionner « Log In » dans la page d'accueil
 2. Entrer les identifiants valides.
 3. Appuyer sur le bouton « Log in »
- **Résultat attendu** : L'utilisateur est authentifié et redirigé vers la page de la liste des formulaires.
- **Post-condition(s)** : L'utilisateur est authentifié et peut utiliser l'application.
- **Moyen de vérification** : L'utilisateur accède aux fonctionnalités après la connexion.

Test de validation : S'authentifier à son compte – Mot de passe incorrect – erreur

- **Nom du test** : b2 – S'authentifier à son compte – Mot de passe incorrect – erreur
- **Use Case correspondant** : S'authentifier à son compte
- **Contexte** : L'utilisateur a déjà un compte valide mais saisit un mot de passe incorrect.
- **Pré-condition(s)** : Le compte utilisateur est déjà créé et activé, l'application est lancée avec une connexion Internet disponible, et l'utilisateur se trouve sur la page d'accueil.
- **Entrée(s)** : Identifiants avec un mot de passe incorrect.
- **Scénario** :
 1. Sélectionner « Log In » dans la page d'accueil.
 2. Entrer un email valide mais un mot de passe incorrect.
 3. Appuyer sur « Log in »
- **Résultat attendu** : Un message d'erreur apparaît, indiquant que le mot de passe est incorrect.
- **Post-condition(s)** : L'utilisateur reste sur l'écran de connexion.
- **Moyen de vérification** : Vérifier que le message d'erreur apparaît et que la connexion échoue.

2.4.1.2.3. Détail du cas d'utilisation : c. Ouvrir un formulaire

- **Nom du cas d'utilisation** : c. Ouvrir un formulaire
- **Acteur principal** : Technicien
- **Acteurs secondaires** : Aucun
- **Objectif** : Permettre à l'utilisateur d'accéder à un formulaire, soit pré-rempli en utilisant un code (QR Code ou lien) , soit vierge sans code.
- **Pré-condition(s)** : L'utilisateur doit être connecté à l'application, disposer d'une connexion internet et se trouve sur la liste des formulaires
- **Hypothèses** : Aucune
- **Scénario** :
 1. Il appuie sur le bouton « New site ».

2. Le système propose trois options: « Scan QR Code (avec le bouton « Scan ») », « Use Link (avec un bouton) » ou « No code ».
 3. L'utilisateur choisit « No code ».
 4. Le système affiche la liste des différents types de formulaires.
 5. L'utilisateur choisit un type de formulaire
 6. Le système affiche le formulaire vierge à remplir
- **Post-condition(s)** : L'utilisateur accède au formulaire souhaité (vierge ou pré-rempli selon le choix).
 - **Exigences non fonctionnelles** : Aucune
 - **Exceptions, alternatives** :
 - **Alternative 1** : En SN3
 1. L'utilisateur clique sur le bouton « Scan » pour scanner un QR code.
 2. Le système active la caméra pour scanner le QR code.
 3. L'utilisateur scanne son code.
 4. Le système redirige l'utilisateur vers le formulaire pré-rempli.
 - **Alternative 2** : En SN3
 1. L'utilisateur entre un lien dans le champ « Link ».
 2. L'utilisateur appuie sur le bouton.
 3. Le système redirige l'utilisateur vers le formulaire pré-rempli.
 - **Exception 1** : En A1
Si le QR code ne peut pas être scanné ou est invalide, un message d'erreur s'affiche.
 - **Exception 2** : En A2
Si le lien est invalide, un message d'erreur s'affiche.

Test de validation : Ouvrir un formulaire avec code valide – nominal

- **Nom du test** : c1 – Ouvrir un formulaire avec code valide – nominal
- **Use Case correspondant** : Ouvrir un formulaire
- **Contexte** : L'utilisateur est connecté et souhaite accéder à un formulaire pré-rempli.
- **Pré-condition(s)** : L'utilisateur est connecté, dispose d'une connexion Internet, possède un code d'accès valide et se trouve sur la liste des formulaires.
- **Entrée(s)** : Code d'accès valide.
- **Scénario** :
 2. Cliquer sur « Nouveau site ».
 3. Choisir de remplir le code d'accès (via QR code ou lien)
 4. Entrer le code d'accès valide.
- **Résultat attendu** : L'utilisateur accède au formulaire pré-rempli.
- **Post-condition(s)** : L'utilisateur peut visualiser et remplir le formulaire.
- **Moyen de vérification** : Vérifier que le formulaire est ouvert avec les données correspondantes.

Test de validation : Ouvrir un formulaire – Code d'accès invalide – erreur

- **Nom du test** : c2 – Ouvrir un formulaire – Code d'accès invalide – erreur
- **Use Case correspondant** : Ouvrir un formulaire
- **Contexte** : L'utilisateur est connecté mais utilise un code invalide.
- **Pré-condition(s)** : L'utilisateur est connecté, dispose d'une connexion Internet, possède un code d'accès invalide et se trouve sur la liste des formulaires.

- **Entrée(s)** : Code d'accès invalide.
- **Scénario** :
 1. Cliquer sur « New site ».
 2. Choisir de remplir le code d'accès (via QR code ou lien).
 3. Entrer un code d'accès invalide.
- **Résultat attendu** : Un message d'erreur apparaît, indiquant que le code est invalide.
- **Post-condition(s)** : L'utilisateur reste sur l'écran de saisie du code.
- **Moyen de vérification** : Vérifier que le message d'erreur apparaît et que le formulaire n'est pas accessible.

Test de validation : Scanner un QR code valide – nominal

- **Nom du test** : c3 – Scanner un QR code valide – nominal
- **Use Case correspondant** : Ouvrir un formulaire
- **Contexte** : L'utilisateur est connecté et souhaite accéder à un formulaire en scannant un QR code valide .
- **Pré-condition(s)** : L'utilisateur est connecté, dispose d'une connexion Internet et se trouve sur la liste des formulaires.
- **Entrée(s)** : QR code valide.
- **Scénario** :
 1. Cliquer sur « New site ».
 2. Choisir l'option « Scan ».
 3. Scanner le QR code valide.
- **Résultat attendu** : Le formulaire pré-rempli correspondant au QR code est accessible.
- **Post-condition(s)** : L'utilisateur accède au formulaire pré-rempli.
- **Moyen de vérification** : Vérifier que le formulaire correspondant est bien ouvert après le scan du QR code.

Test de validation : Scanner un QR code invalide – erreur

- **Nom du test** : c4 – Scanner un QR code invalide – erreur
- **Use Case correspondant** : Ouvrir un formulaire
- **Contexte** : L'utilisateur scanne un QR code invalide.
- **Pré-condition(s)** : L'utilisateur est connecté, dispose d'une connexion Internet et se trouve sur la liste des formulaires.
- **Entrée(s)** : QR code invalide.
- **Scénario** :
 1. Cliquer sur « New site ».
 2. Choisir l'option « Scan ».
 3. Scanner un QR code invalide.
- **Résultat attendu** : Un message d'erreur apparaît, indiquant que le QR code est invalide.
- **Post-condition(s)** : L'utilisateur ne peut pas accéder au formulaire et reste sur l'écran d'initialisation.
- **Moyen de vérification** : Vérifier que le message d'erreur s'affiche et que le formulaire n'est pas ouvert.

Test de validation : Ouvrir un formulaire vierge sans lien ou QR code – nominal

- **Nom du test** : c5 – Ouvrir un formulaire vierge sans lien ou QR code – nominal
- **Use Case correspondant** : Ouvrir un formulaire
- **Contexte** : L'utilisateur souhaite remplir un formulaire vierge sans utiliser de code.
- **Pré-condition(s)** : L'utilisateur est connecté, dispose d'une connexion Internet et se trouve sur la liste des formulaires.
- **Entrée(s)** : Aucun lien ni QR code requis.
- **Scénario** :
 1. Cliquer sur « New site ».
 2. Choisir le modèle de formulaire.
- **Résultat attendu** : L'utilisateur accède à un formulaire vierge.
- **Post-condition(s)** : Le formulaire vierge est ouvert et peut être rempli.
- **Moyen de vérification** : Vérifier que l'utilisateur accède à un formulaire vierge sans pré-remplissage.

2.4.1.2.4. Détail du cas d'utilisation : d. Saisir une réponse à un formulaire

- **Nom du cas d'utilisation** : d. Saisir une réponse à un formulaire
- **Acteur principal** : Technicien
- **Acteurs secondaires** : Aucun
- **Objectif** : Permettre à l'utilisateur de remplir un formulaire avec les informations requises.
- **Pré-condition(s)** : L'utilisateur a ouvert un formulaire.
- **Hypothèses** : Aucune
- **Scénario** :
 1. Le système affiche un formulaire de saisie avec des champs textes ou boutons « Add files »
 2. L'utilisateur remplit les champs texte du formulaire.
 3. Le système valide dynamiquement les champs pendant la saisie.
- **Post-condition(s)** : Les données saisies sont enregistrées en tant que brouillon ou prêtes pour soumission.
- **Exigences non fonctionnelles** : Aucune
- **Exceptions, alternatives** :
 - **Alternative 1** : En SN2
 1. L'utilisateur sélectionne le bouton « Add files ».
 2. Le système affiche un sélecteur de fichiers acceptant les formats documents ou photos.
 3. L'utilisateur choisit un ou plusieurs fichiers.
 4. Le système affiche le nom du document dans le champ associé.
 - **Exception 1** : Si un champ obligatoire n'est pas rempli, un message d'erreur s'affiche pour informer l'utilisateur.

Test de validation : Saisir une réponse à un formulaire – nominal

- **Nom du test** : d1 – Saisir une réponse à un formulaire – nominal
- **Use Case correspondant** : Saisir une réponse à un formulaire
- **Contexte** : L'utilisateur est connecté et a ouvert un formulaire vierge ou pré-rempli.
- **Pré-condition(s)** : L'utilisateur a un formulaire ouvert.
- **Entrée(s)** : Informations requises pour le formulaire.

- **Scénario :**
 1. Remplir les champs du formulaire.
 2. Ajouter des photos ou documents, si nécessaire.
- **Résultat attendu :** Le formulaire est correctement rempli et validé sans erreurs.
- **Post-condition(s) :** Le formulaire est prêt pour la soumission ou sauvegarde en brouillon.
- **Moyen de vérification :** Vérifier que les données sont enregistrées et qu'il n'y a pas d'erreurs lors de la saisie.

Test de validation : Saisir une réponse à un formulaire – Champ obligatoire manquant – erreur

- **Nom du test :** d2 – Saisir une réponse à un formulaire – Champ obligatoire manquant – erreur
- **Use Case correspondant :** Saisir une réponse à un formulaire
- **Contexte :** L'utilisateur tente de soumettre un formulaire sans avoir rempli un champ obligatoire.
- **Pré-condition(s) :** Un formulaire ouvert avec des champs obligatoires non remplis.
- **Entrée(s) :** Formulaire partiellement rempli avec des champs obligatoires manquants.
- **Scénario :**
 1. Ne pas remplir tous les champs obligatoires.
 2. Soumettre le formulaire sans remplir un champ obligatoire.
- **Résultat attendu :** Le système affiche un message d'erreur signalant le champ obligatoire manquant.
- **Post-condition(s) :** Le formulaire ne peut pas être soumis tant que les champs obligatoires ne sont pas remplis.
- **Moyen de vérification :** Vérifier l'affichage d'un message d'erreur et l'impossibilité de soumettre le formulaire.

2.4.1.2.5. Détail du cas d'utilisation : e. Gérer une réponse

- **Nom du cas d'utilisation :** e. Gérer une réponse
- **Acteur principal :** Technicien
- **Acteurs secondaires :** Aucun
- **Objectif :** Permettre à l'utilisateur d'envoyer, de supprimer ou d'enregistrer en brouillon sa réponse.
- **Pré-condition(s) :** Le formulaire doit être rempli.
- **Hypothèses :** Aucune
- **Scénario :**
 1. Le système affiche un formulaire rempli.
 2. L'utilisateur appuie sur « Submit » pour envoyer sa réponse.
 3. Le système redirige l'utilisateur vers une page de confirmation de soumission.
 4. L'utilisateur appuie sur « Submit » pour valider son choix.
 5. Le système affiche une notification ou snackbar sur l'application comme confirmation lorsque le formulaire finit d'être envoyé.
 6. L'utilisateur reçoit un email de validation.
 7. Le système redirige l'utilisateur vers la page de la liste des formulaires, où le formulaire apparaît avec son statut de soumission.

- **Post-condition(s)** : Le formulaire est soumis et stocké dans le cloud.
- **Exigences non fonctionnelles** : La connexion doit être stable à un moment donné
- **Exceptions, alternatives** :
 - **Alternative 1** : En SN2
 1. L'utilisateur appuie sur « Delete » pour supprimer sa réponse.
 2. Le système affiche un message demandant la confirmation de suppression.
 3. L'utilisateur appuie sur « Delete » pour confirmer.
 4. Le système affiche un snack bar sur l'application pour confirmer la suppression de la réponse.
 5. Le système redirige l'utilisateur vers la page de la liste des formulaires.
 - **Alternative 2** : En SN2
 1. L'utilisateur appuie sur « Draft » pour enregistrer sa réponse.
 2. Le système affiche un snackbar sur l'application pour confirmer l'enregistrement de la réponse.
 3. Le système redirige l'utilisateur vers la page de la liste des formulaires. Le formulaire y apparaît avec le statut « Draft ».
 - **Exception 1** : En SN4

En cas d'échec de la soumission, le système affiche un snackbar et les champs en erreur (indiqués en rouge), permettant à l'utilisateur de les corriger et de renvoyer le formulaire.
 - **Exception 2** : En A2

L'utilisateur clique sur « Cancel » dans le message de confirmation pour annuler la suppression de la réponse, le système revient au formulaire.

Test de validation : Envoyer une réponse – nominal

- **Nom du test** : e1 – Envoyer une réponse – nominal
- **Use Case correspondant** : Gérer une réponse
- **Contexte** : L'utilisateur a terminé de remplir un formulaire et est prêt à le soumettre.
- **Pré-condition(s)** : Le formulaire est complet et correctement rempli. L'application est connectée à Internet.
- **Entrée(s)** : Formulaire rempli.
- **Scénario** :
 1. Cliquer sur « Submit ».
 2. Vérifier les informations.
 3. Confirmer l'envoi du formulaire.
- **Résultat attendu** : Le formulaire est soumis avec succès et l'utilisateur reçoit une notification de confirmation dans l'application ainsi qu'un email de confirmation
- **Post-condition(s)** : Le formulaire est stocké dans le cloud, l'utilisateur reçoit une confirmation par email.
- **Moyen de vérification** : Vérifier la réception de la notification dans l'application et l'email de confirmation.

Test de validation : Envoyer une réponse - Sans connection Internet – nominal

- **Nom du test** : e1 – Envoyer une réponse – nominal
- **Use Case correspondant** : Gérer une réponse
- **Contexte** : L'utilisateur a terminé de remplir un formulaire et est prêt à le soumettre.

- **Pré-condition(s)** : Le formulaire est complet et correctement rempli mais l'utilisateur ne dispose pas d'une connexion Internet.
- **Entrée(s)** : Formulaire rempli.
- **Scénario** :
 1. Cliquer sur « Submit ».
 2. Vérifier les informations.
 3. Confirmer l'envoi du formulaire en cliquant sur « Submit ».
- **Résultat attendu** : Un snackbar informe l'utilisateur que le formulaire sera envoyé automatiquement lorsque la connexion sera rétablie. Le formulaire est enregistré en attente d'envoi.
- **Post-condition(s)** : Le formulaire est stocké localement en attente de connexion.
- **Moyen de vérification** : Vérifier que le formulaire est marqué comme « Waiting for connection » dans l'application. Vérifier que le message d'erreur s'affiche.

Test de validation : Envoyer une réponse – Envoi partiel

- **Nom du test** : e2 – Envoyer une réponse – Envoi partiel
- **Use Case correspondant** : Gérer une réponse
- **Contexte** : L'utilisateur a terminé de remplir un formulaire et est prêt à le soumettre. La connexion Internet est instable, avec des risques de coupure.
- **Pré-condition(s)** : Le formulaire est complet et correctement rempli.
- **Entrée(s)** : Formulaire rempli.
- **Scénario** :
 1. L'utilisateur clique sur « Submit ».
 2. Vérifier les informations.
 3. L'utilisateur clique sur « Submit » pour confirmer l'envoi du formulaire.
- **Résultat attendu** : En cas de coupure de connexion pendant l'envoi, un message d'erreur s'affiche pour indiquer que l'envoi a échoué. Le formulaire reste enregistré localement et en attente d'envoi pour reprendre automatiquement dès que la connexion Internet est rétablie.
- **Post-condition(s)** : Le formulaire est stocké localement en attente de connexion.
- **Moyen de vérification** : Vérifier que le formulaire est marqué comme « Waiting for connection » dans l'application. Vérifier que le message d'erreur s'affiche en cas de coupure de connexion.

Test de validation : Enregistrer une réponse – nominal

- **Nom du test** : e3 – Enregistrer une réponse – nominal
- **Use Case correspondant** : Gérer une réponse
- **Contexte** : L'utilisateur souhaite enregistrer un formulaire en cours de remplissage pour le reprendre plus tard.
- **Pré-condition(s)** : Un formulaire est en cours de remplissage.
- **Entrée(s)** : /
- **Scénario** :
 1. Cliquer sur « Draft ».
- **Résultat attendu** : Le formulaire est enregistré en tant que brouillon localement sur l'appareil et un snackbar s'affiche.

- **Post-condition(s)** : Le formulaire est récupérable dans la liste des formulaires avec le statut « Draft » ou dans la liste spécifiques des brouillons.
- **Moyen de vérification** : Vérifier que le formulaire apparaît bien dans la liste principale des formulaires avec le statut « Draft ». Vérifier également que le formulaire est visible dans la liste des brouillons.

Test de validation : Supprimer une réponse – nominal

- **Nom du test** : e4 – Supprimer une réponse – nominal
- **Use Case correspondant** : Gérer une réponse
- **Contexte** : Permettre à l'utilisateur de supprimer un formulaire en cours de remplissage ou déjà enregistré.
- **Pré-condition(s)** : L'utilisateur doit être sur le formulaire.
- **Entrée(s)** : Formulaire à supprimer.
- **Scénario** :
 1. Cliquer sur l'option « Delete ».
 3. Confirmer la suppression en cliquant sur « Delete » dans le message de confirmation.
- **Résultat attendu** : Le formulaire est supprimé avec succès, et un snackbar s'affiche pour indiquer que la suppression a bien eu lieu.
- **Post-condition(s)** : Le formulaire n'apparaît pas dans la liste.
- **Moyen de vérification** : Vérifier que le formulaire n'apparaît pas dans la liste. Vérifier également que le formulaire n'apparaît pas dans la liste des brouillons, s'il y était présent.

2.4.1.2.6. Détail du cas d'utilisation : h. Sélectionner la langue des formulaires

- **Nom du cas d'utilisation** : h. Sélectionner la langue des formulaires.
- **Acteur principal** : Technicien
- **Acteurs secondaires** : Aucun
- **Objectif** : Permettre à l'utilisateur de changer la langue des formulaires.
- **Pré-condition(s)** : L'utilisateur est dans les paramètres de l'application.
- **Hypothèses** : Aucune
- **Scénario** :
 1. L'utilisateur choisit l'option « Language » dans les paramètres.
 2. L'utilisateur sélectionne une langue dans la liste disponible.
 3. Le système affiche une demande de confirmation de la langue.
 4. L'utilisateur confirme.
- **Post-condition(s)** : Les formulaires sont affichées dans la langue choisie par l'utilisateur.
- **Exigences non fonctionnelles** : Aucune
- **Exceptions, alternatives** :
 - **Alternative 1** : en SN3, l'utilisateur peut annuler le changement de langue, et revient alors en SN2.

Test de validation : Sélectionner la langue des formulaires – nominal

- **Nom du test** : h1 – Sélectionner la langue des formulaires – nominal
- **Use Case correspondant** : Sélectionner la langue des formulaires

- **Contexte** : L'utilisateur souhaite changer la langue du formulaire.
- **Pré-condition(s)** : L'utilisateur doit être dans les paramètres de l'application.
- **Entrée(s)** : Une langue, l'Anglais.
- **Scénario** :
 1. Choisir l'option « Language » dans les paramètres.
 2. Choisir la langue « Anglais » dans la liste.
 3. Confirmer le changement de langue.
- **Résultat attendu** : Les formulaires sont désormais affichés dans cette langue.
- **Post-condition(s)** : Les formulaires sont affichés dans la langue choisie par l'utilisateur.
- **Moyen de vérification** : Vérifier que les questions d'un formulaire s'affichent dans la langue sélectionnée.

2.4.1.2.7. Détail du cas d'utilisation : i. Consulter sa réponse à un formulaire

- **Nom du cas d'utilisation** : i. Consulter sa réponse à un formulaire.
- **Acteur principal** : Technicien
- **Acteurs secondaires** : Aucun
- **Objectif** : Permettre au technicien de consulter les formulaires qu'il a envoyés ou enregistrés en brouillon.
- **Pré-condition(s)** : L'utilisateur est sur la page listant ses formulaires. Il a déjà accédé à au moins un formulaire.
- **Hypothèses** : Aucune
- **Scénario** :
 1. L'utilisateur clique sur un formulaire envoyé.
 2. Le système ouvre le formulaire sélectionné en lecture seule.
- **Post-condition(s)** : Les questions et réponses du formulaire sont affichées.
- **Exigences non fonctionnelles** : Aucune
- **Exceptions, alternatives** : **Alternative 1**: En SN1
 - L'utilisateur clique sur un formulaire enregistré en brouillon.
 - Le système ouvre le formulaire non soumis pour consultation.

Test de validation : Consulter sa réponse à un formulaire – nominal

- **Nom du test** : i1 – Consulter sa réponse à un formulaire – nominal
- **Use Case correspondant** : Consulter sa réponse à un formulaire
- **Contexte** : L'utilisateur est connecté et se trouve sur la liste des formulaires.
- **Pré-condition(s)** : L'utilisateur est sur la liste des formulaires et dispose d'un formulaire envoyé ou enregistré.
- **Entrée(s)** : Un formulaire appartenant à l'utilisateur.
- **Scénario** :
 1. Sélectionner l'option consulter associée au formulaire.
- **Résultat attendu** : Le formulaire est affiché et l'utilisateur peut consulter les questions et réponses.
- **Post-condition(s)** : La réponse est affichée et consultée par l'utilisateur.
- **Moyen de vérification** : Vérifier que le formulaire s'affiche correctement avec les réponses enregistrées.

2.4.2. Serveur pour le backend

2.4.2.1. Architecture et diagramme des cas d'utilisation

L'employé de Smart Impulse doit pouvoir ajouter des formulaires vierges à la liste accessible par les techniciens. Il doit également pouvoir envoyer un formulaire pré-rempli à un technicien spécifique. Il peut recevoir la réponse à un formulaire, et également recevoir le compte-rendu d'intervention par e-mail.

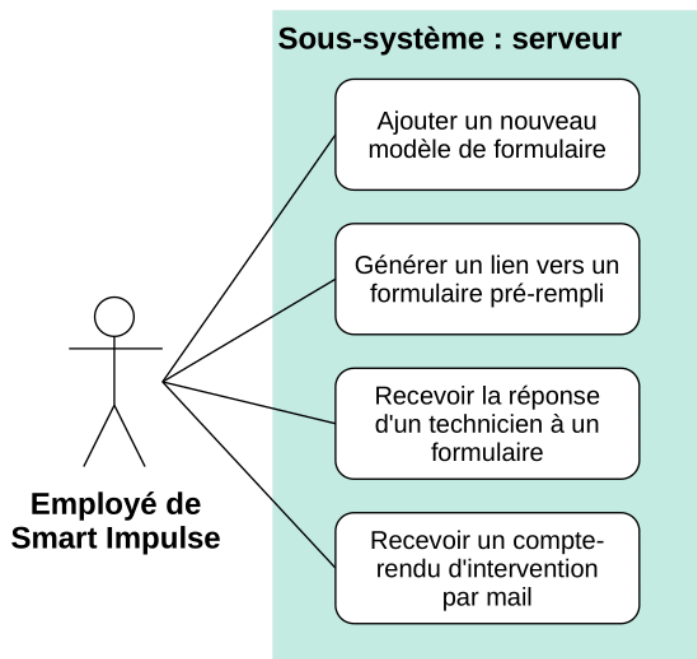


Fig. 4. Diagramme des cas d'utilisation du backend.

2.4.2.2. Acteurs

Employé de Smart Impulse :

- **Type** : Humain
- **Rôle** : Administrateur du serveur

2.4.2.3. Détails des cas d'utilisation du serveur

Le serveur est un composant bouchon du projet, nous ne devons donc pas développer d'interface homme-machine pour ce composant pour interagir avec. Voici quelques lignes expliquant comment les fonctionnalités minimales seront assurées.

Pour ajouter un nouveau modèle de formulaire, l'administrateur de SIForms devra enregistrer ses données brutes dans le serveur et ajouter son chemin dans la base de données relationnelle.

Pour générer un lien vers un formulaire pré-rempli, l'administrateur de SIForms devra utiliser une ligne de commande qui donne le lien associé à un formulaire pré-rempli spécifique s'il existe déjà, ou le crée sinon.

L'administrateur pourra de plus consulter les données brutes des réponses reçues par le serveur.

Enfin, il pourra utiliser une ligne de commande pour générer un document PDF de compte-rendu d'intervention à partir d'une réponse à un formulaire, et l'envoyer à une adresse email spécifique.

2.5. Interactions entre les systèmes

2.5.1. Inscription

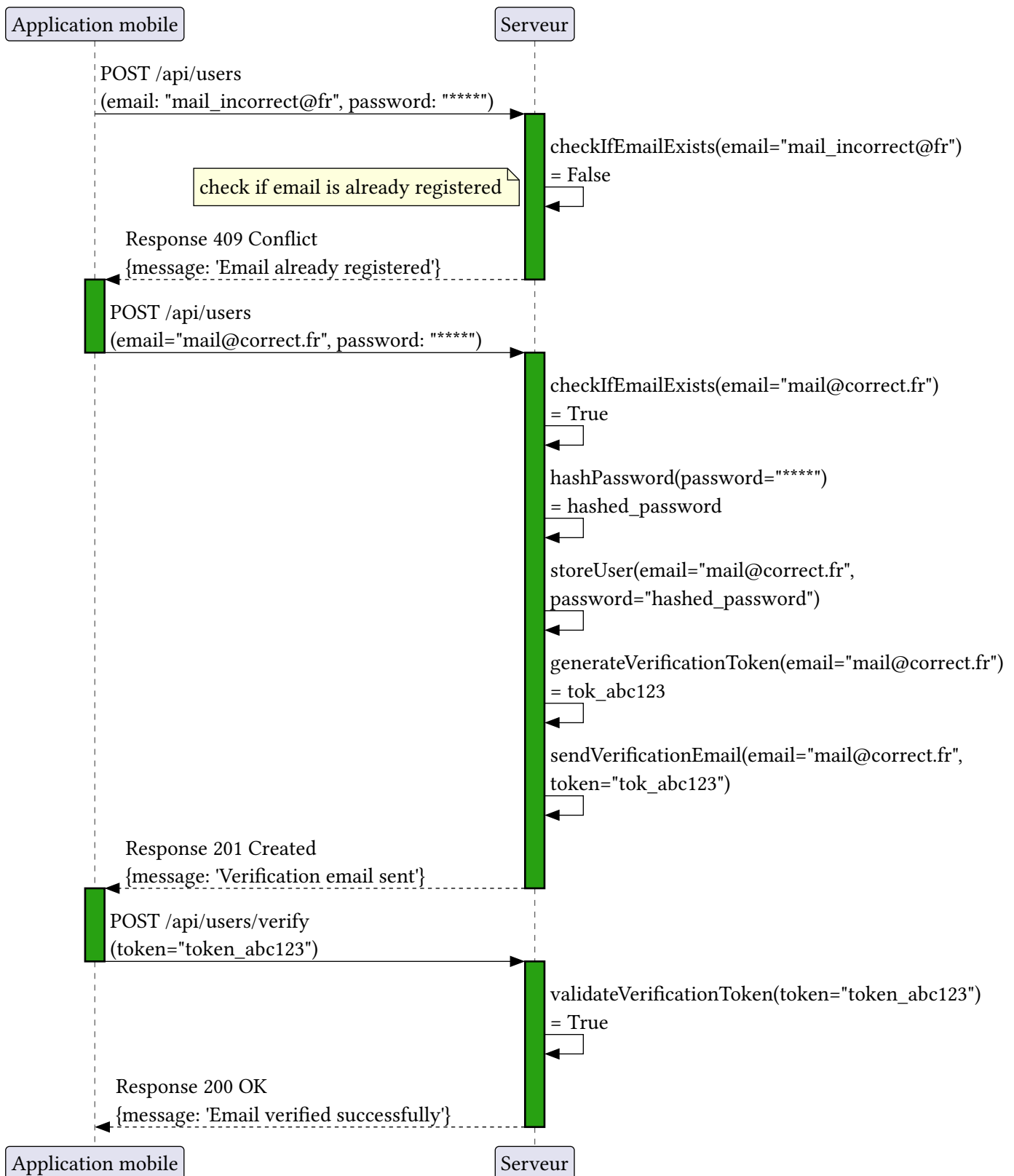


Fig. 5. Diagramme de séquence pour s'inscrire. Un token de session est généré par le serveur lors de l'authentification de l'utilisateur et envoyé à l'application mobile. Ce token est ensuite envoyé dans chaque message venant de l'application, permettant au serveur de vérifier l'identité de l'envoyeur.

2.5.2. Authentification

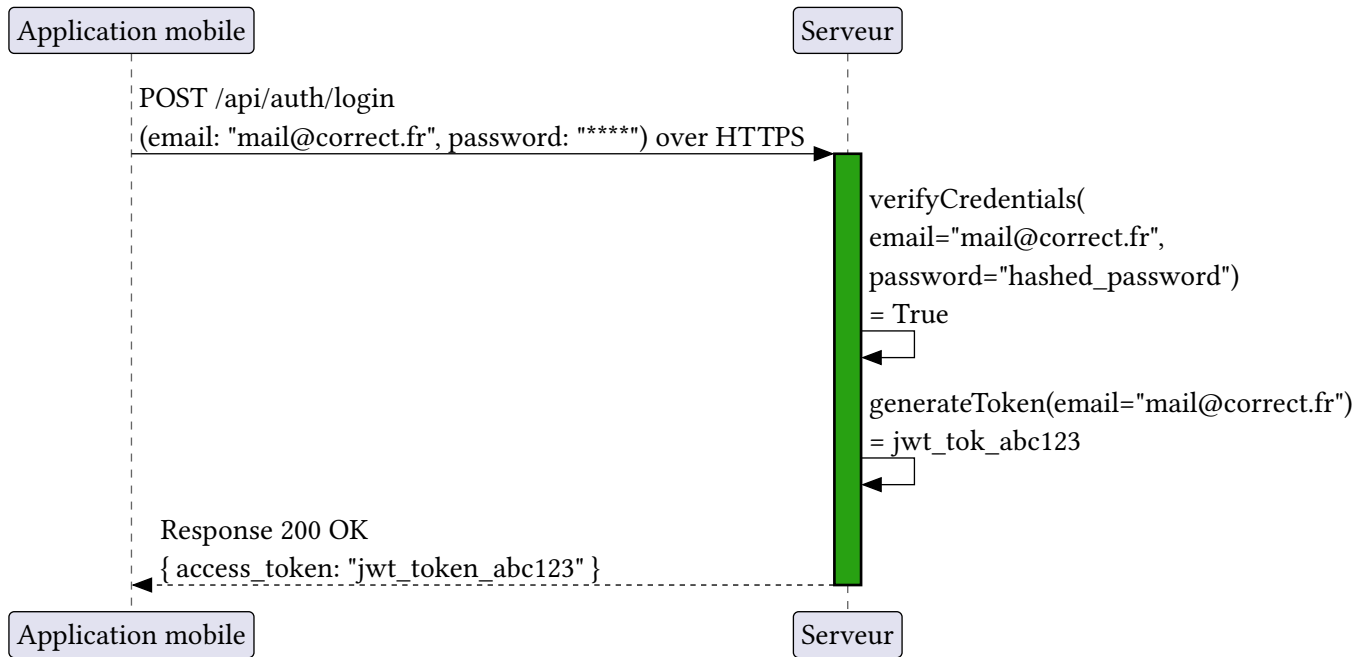


Fig. 6. Diagramme de séquence pour s'authentifier.

2.5.3.

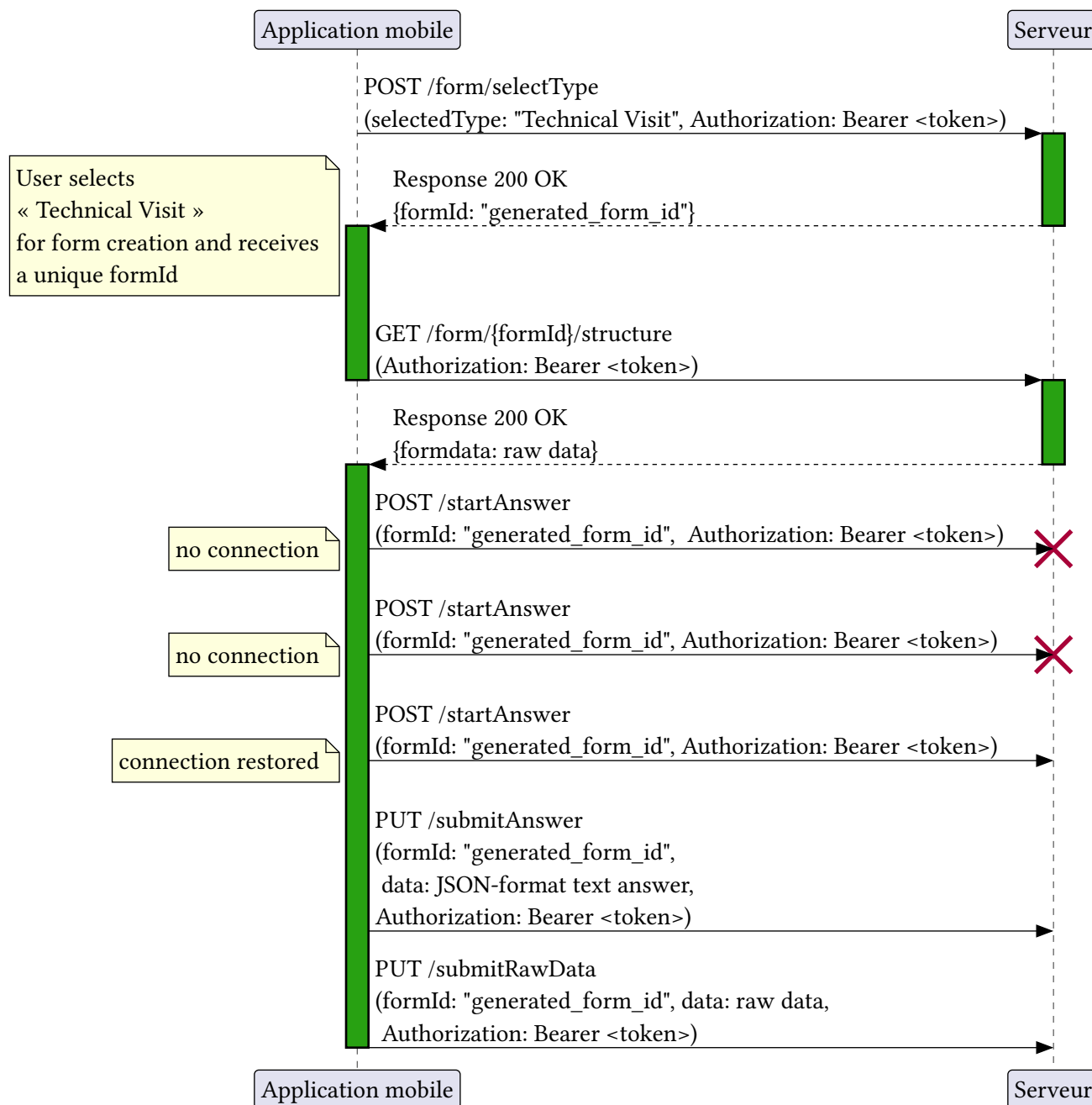


Fig. 7. Diagramme de séquence pour répondre à un formulaire.

2.6. Interactions entre les cas d'utilisation

3. Contraintes et risques

3.1. Les types de contraintes

3.1.1. Contraintes fonctionnelles

Les contraintes fonctionnelles de l'application SIForms incluent la nécessité pour les techniciens de s'adapter à l'utilisation de la nouvelle application. En effet, certains des utilisateurs ne l'utiliseront qu'une seule fois, donc sa prise en main doit être rapide grâce à une bonne ergonomie.

Il devront s'authentifier, ce qui exige une connexion Internet et peut poser problème dans les zones sans réseau. L'accès aux formulaires pré-remplis via des liens ou QR codes dépend également d'une connexion, limitant l'usage hors ligne.

De plus, la mise en place de fonctionnalités hors ligne, telles que le remplissage et la soumission de formulaires, peut entraîner des contraintes supplémentaires, notamment la sauvegarde de données localement sur les appareils des techniciens et leur synchronisation une fois la connexion rétablie. La validation des données doit se faire en temps réel pour éviter les erreurs.

Par ailleurs, les modèles de formulaires doivent être envoyés par le serveur, car ils peuvent être modifiés par Smart Impulse.

3.1.2. Performances et planification

L'application doit garantir des temps de réponse inférieurs à 3 secondes dans les différentes actions d'authentification. Elle doit être robuste pour fonctionner en mode hors ligne sans perte de données tout en offrant une synchronisation fiable dès que la connexion est rétablie.

3.1.3. Ressources

Sur le plan matériel, il sera essentiel de disposer d'appareils variés pour tester l'application, en particulier pour iOS, qui requiert des outils spécifiques comme un Mac pour les tests en conditions réelles.

3.1.4. Techniques

Le projet comporte plusieurs contraintes techniques. Tout d'abord, la nécessité d'apprendre rapidement Django et ReactNative pour ne pas ralentir le développement du projet.

Ensuite, la mise en œuvre d'un système asynchrone pour permettre la soumission hors ligne des formulaires et leur synchronisation présente des défis techniques, notamment pour garantir que les formulaires remplis hors ligne sont correctement soumis dès que la connexion est rétablie, sans perte de données. Il est également crucial de s'assurer que les informations soumises soient toujours cohérentes et valides.

En termes de sécurité, l'application devra garantir la protection des données des utilisateurs, tant pendant le stockage local que lors du transit, en appliquant des techniques de cryptage appropriées. Il faudra aussi s'assurer de la compatibilité multi-plateforme (iOS et Android) avec des performances suffisantes et des apparences cohérentes.

3.1.5. Mise en exploitation de SIForms

Les appareils des techniciens doivent répondre aux prérequis techniques (Android 8.0+ et iOS 14+), ce qui pourrait nécessiter une mise à jour ou un remplacement de certains terminaux.

Bien que l'application fonctionne hors ligne, une connexion Internet est indispensable pour l'installation, la création de compte initiale et le téléchargement d'un formulaire.

Les données utilisateurs (formulaires des Jotform) pourront éventuellement être migrées manuellement depuis l'ancien système.

3.2. Analyse des risques

3.2.1. Risques liés aux contraintes fonctionnelles

- **Nom du risque** : Dépendance à la connexion pour l'authentification et à l'accès aux différents types de formulaires.

- **Description** : Pour accéder à son compte et aux formulaires, les techniciens doivent s'authentifier ou utiliser un code d'accès nécessitant ainsi une connexion Internet. Cependant, les techniciens travaillent dans des zones sans réseau.
- **Causes** : Mauvaise couverture réseau dans la zone où les techniciens travaillent et nécessité d'une connexion Internet pour l'authentification, l'utilisation du code d'accès (QR code ou lien) et l'accès aux formulaires.
- **Conséquences** : Si le technicien n'est pas connecté avant d'entrer dans une zone sans réseau, il ne pourra pas accéder à son compte. Si le technicien est déjà authentifié mais doit utiliser un QR code ou un lien dans une zone sans réseau, il ne pourra pas accéder aux formulaires pré-remplis.
- **Moyens de prévention** : Les techniciens doivent se connecter obligatoirement avant d'entrer dans leur zone de travail.
- **Gravité** : Haute, car ce risque peut entraîner des retards dans le travail des techniciens.

3.2.2. Risques liés aux performances

- **Nom du risque** : Problème de gestion de fichiers photos volumineux.
- **Description** : Si les photos téléchargées dans le formulaire sont de grande taille, cela peut entraîner des lenteurs dans le traitement, l'envoi et le stockage des fichiers.
- **Causes** : Les photos de grande taille prennent du temps à être téléchargées surtout avec une connexion Internet lente.
- **Conséquences** : Temps de téléchargement et soumission des formulaires très longs. Les techniciens ne peuvent pas réussir à soumettre leurs formulaires.
- **Moyens de prévention** : Compression des images côté client afin de réduire la taille des photos sans perte de qualité.
- **Gravité** : Haute, car cela peut entraîner des ralentissements et peut affecter la capacité à soumettre les formulaires.

3.2.3. Risques liés aux ressources

Pas de risques.

3.2.4. Risques liés aux contraintes techniques

- **Nom du risque** : Changement de version du formulaire entre la sauvegarde et la soumission.
- **Description** : Un formulaire sauvegardé en brouillon par un technicien peut ne plus correspondre à la version actuelle du formulaire sur le serveur si ce dernier a été mis à jour avant la soumission.
- **Causes** : Le formulaire a été modifié sur le serveur après que le technicien l'a sauvegardé en tant que brouillon.
- **Conséquences** : Lors de la tentative de soumission, des erreurs pourraient survenir en raison de différences entre la version en brouillon et la nouvelle version du formulaire. Cela pourrait empêcher la soumission ou nécessiter des modifications de la part du technicien.
- **Moyens de prévention** : Implémenter un versionnage des formulaires. Chaque template possède un numéro de version. Si une nouvelle version est déployée, le technicien n'a pas besoin de refaire le formulaire puisque ces réponses sont conservées pour la version initiale.
- **Gravité** : Moyenne à haute, car cela peut entraîner une perte de temps pour le technicien si le formulaire doit être modifié ou complété à nouveau.

3.2.5. Risques liés aux contraintes procédurales

Pas de risques liés aux contraintes procédurales.

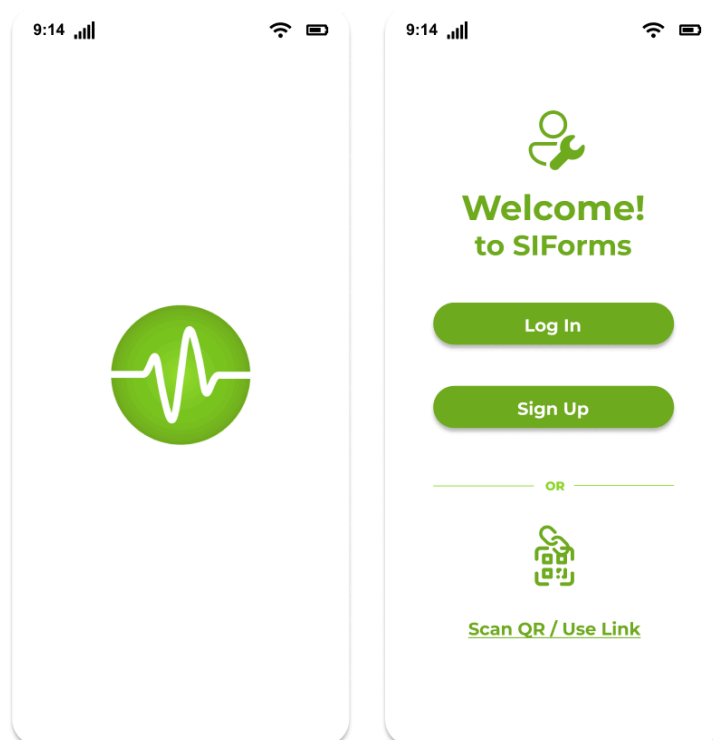
4. Prototypage

4.1. Maquette sur Figma

Lien vers la maquette entière: <https://www.figma.com/design/icz5FlRxcwlkh4gO3RfjRc/SmartImpulse?node-id=0-1&t=XvvCQ8KwxacrBNly-1>

Écran splash : Cet écran splash s'affiche lors de l'ouverture de l'application, avec le logo de Smart Impulse.

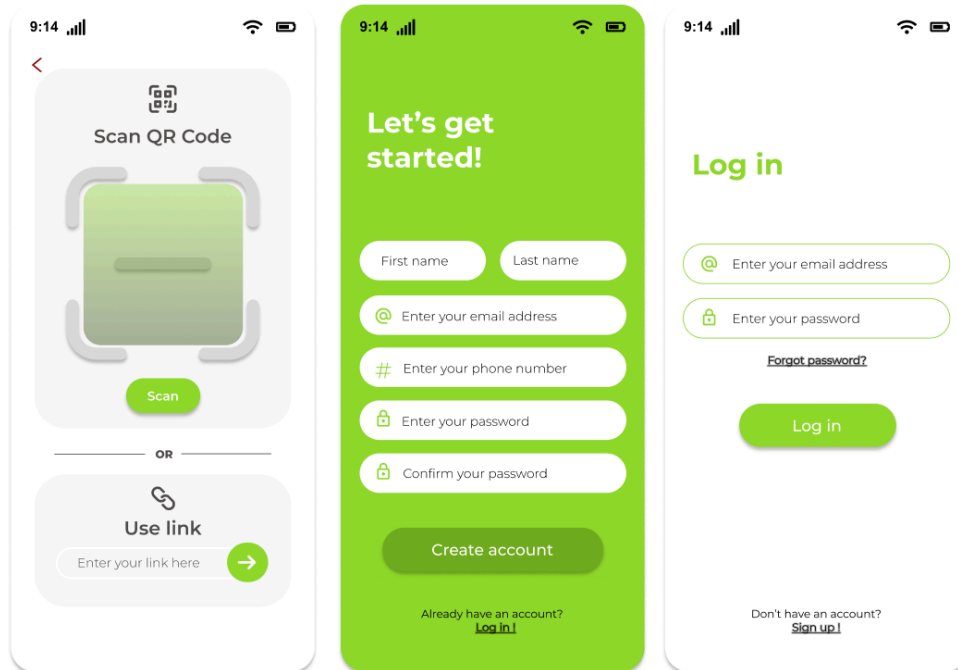
Écran d'accueil : L'utilisateur a deux options principales : « Log In » pour se connecter et « Sign Up » pour créer un compte. Il existe également une option pour scanner un code QR ou utiliser un lien direct afin d'accéder rapidement aux formulaires pré-remplis.



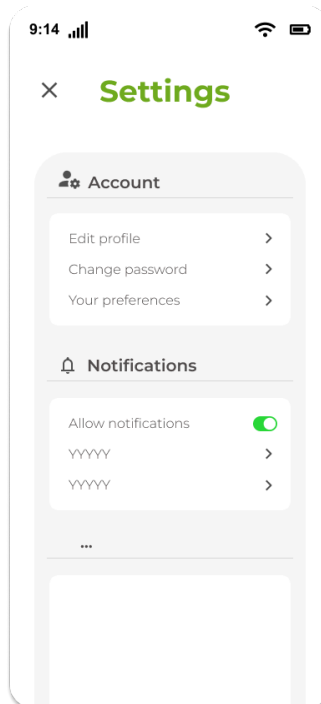
Écran de connexion : Si l'utilisateur a déjà un compte, il peut directement se connecter en entrant son adresse e-mail et mot de passe. Il y a également une option pour récupérer un mot de passe oublié et une option de création de compte pour les nouveaux utilisateurs.

Écran de création de compte: L'utilisateur crée un compte en fournissant son prénom, nom, adresse e-mail, mot de passe et numéro de téléphone.

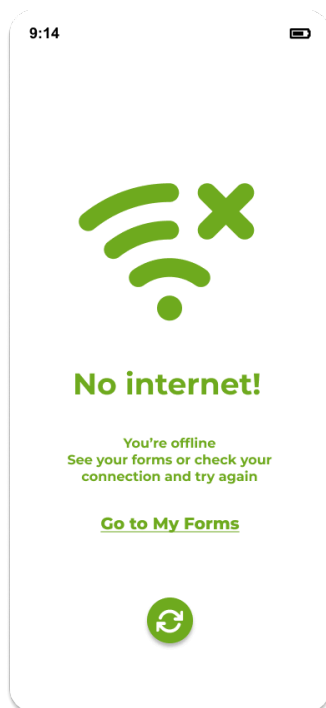
Écran de scan QR: Permet à l'utilisateur de scanner un code QR ou d'entrer manuellement un lien pour accéder à un formulaire spécifique. Après le scan, l'utilisateur est invité à se connecter ou à créer un compte. Une fois authentifié, l'utilisateur est dirigé vers le formulaire pré-rempli.



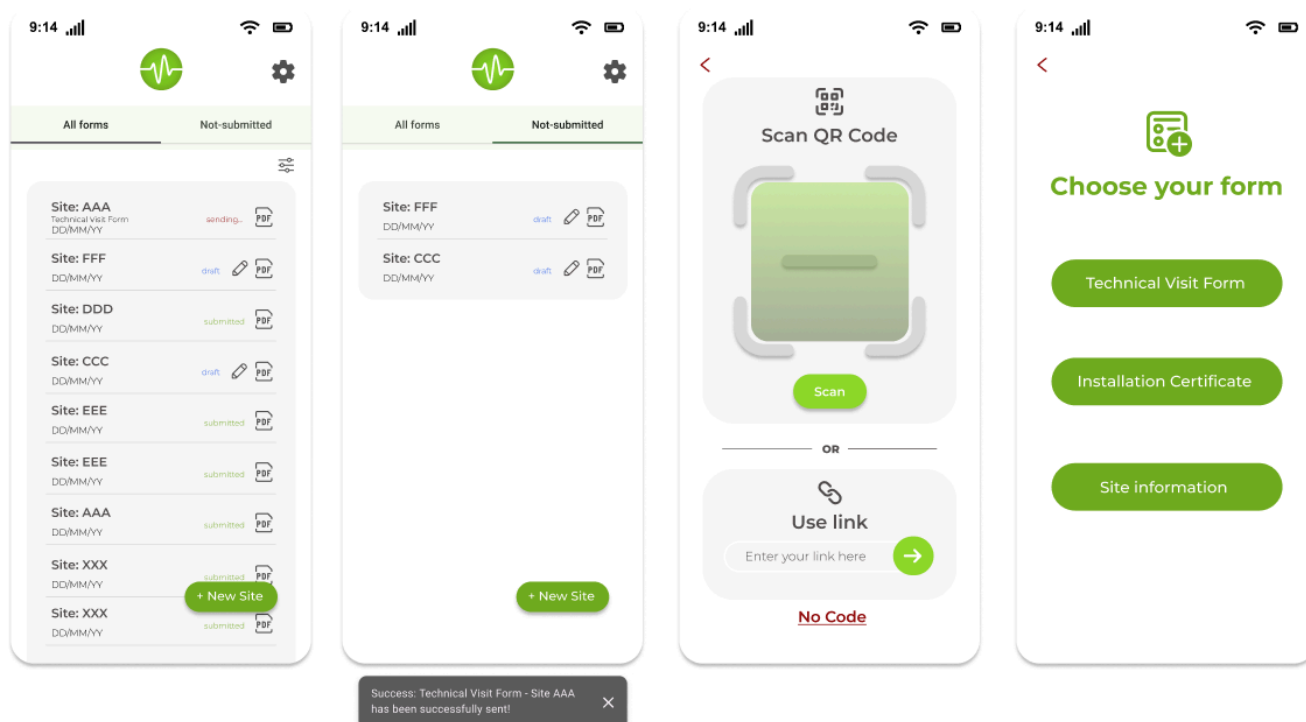
Écran de paramètres : Permet aux utilisateurs de gérer les informations (Détails du profil...) et les préférences (Langues des formulaires...) de leur compte ainsi que leurs notifications.



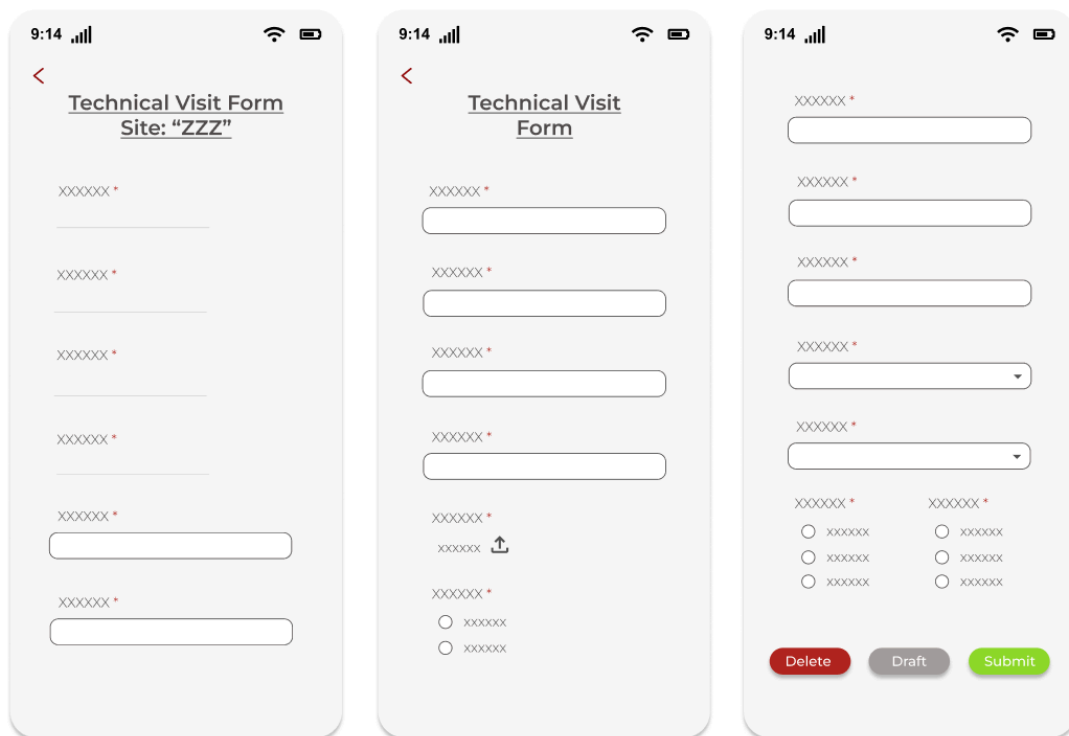
Écran hors-ligne : Informe l'utilisateur qu'il est hors ligne et lui permet d'accéder aux formulaires déjà enregistrés ou de réessayer la connexion lorsqu'elle sera rétablie.



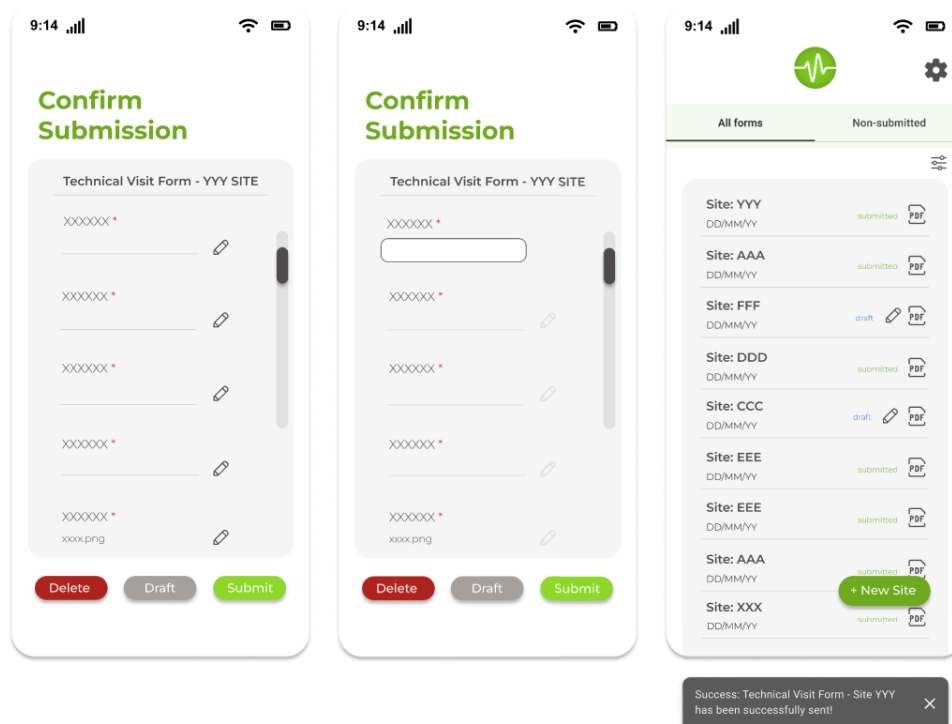
Écrans des formulaires : La page principale affiche deux onglets : un pour tous les formulaires « All forms » et un pour les formulaires non soumis « Not submitted ». En cliquant sur « New site », l'utilisateur est dirigé vers une page de scan QR ou de saisie de code, avec l'option No Code pour accéder directement au choix du type de formulaire.



Écrans de remplissage d'un formulaire : L'utilisateur répond à un formulaire (pré-rempli pour un site ou vierge). Il peut le sauvegarder en tant que brouillon, le soumettre, ou le supprimer.



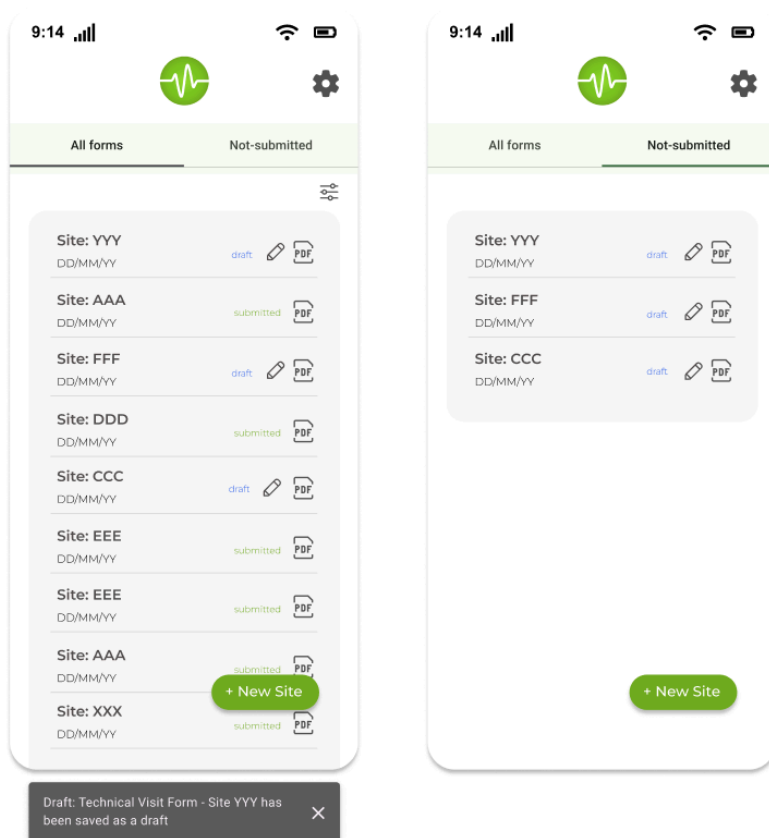
Écrans de confirmation et soumission d'une réponse : L'utilisateur confirme sa réponse à un formulaire en appuyant sur « Submit ».



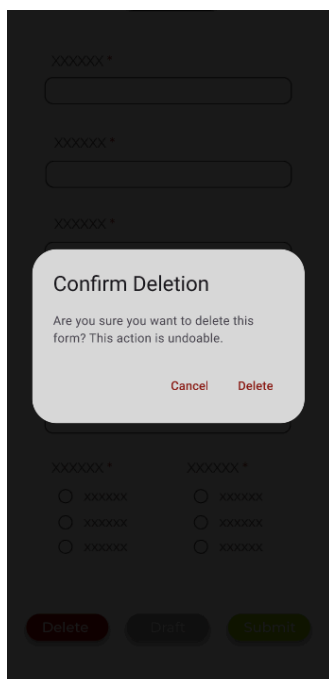
Écran de formulaire invalide : L'utilisateur doit corriger les erreurs pour pouvoir soumettre sa réponse.



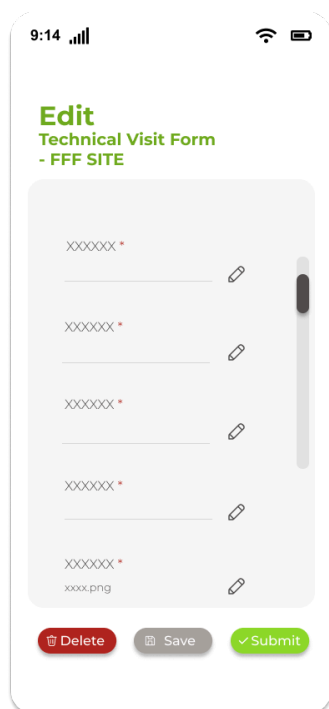
Écrans d'enregistrement d'une réponse : L'utilisateur appuie sur « Draft » pour enregistrer le formulaire.



Écran de suppression d'une réponse : L'utilisateur appuie sur « Delete » pour supprimer sa réponse à un formulaire. Une confirmation de suppression est ensuite demandée.

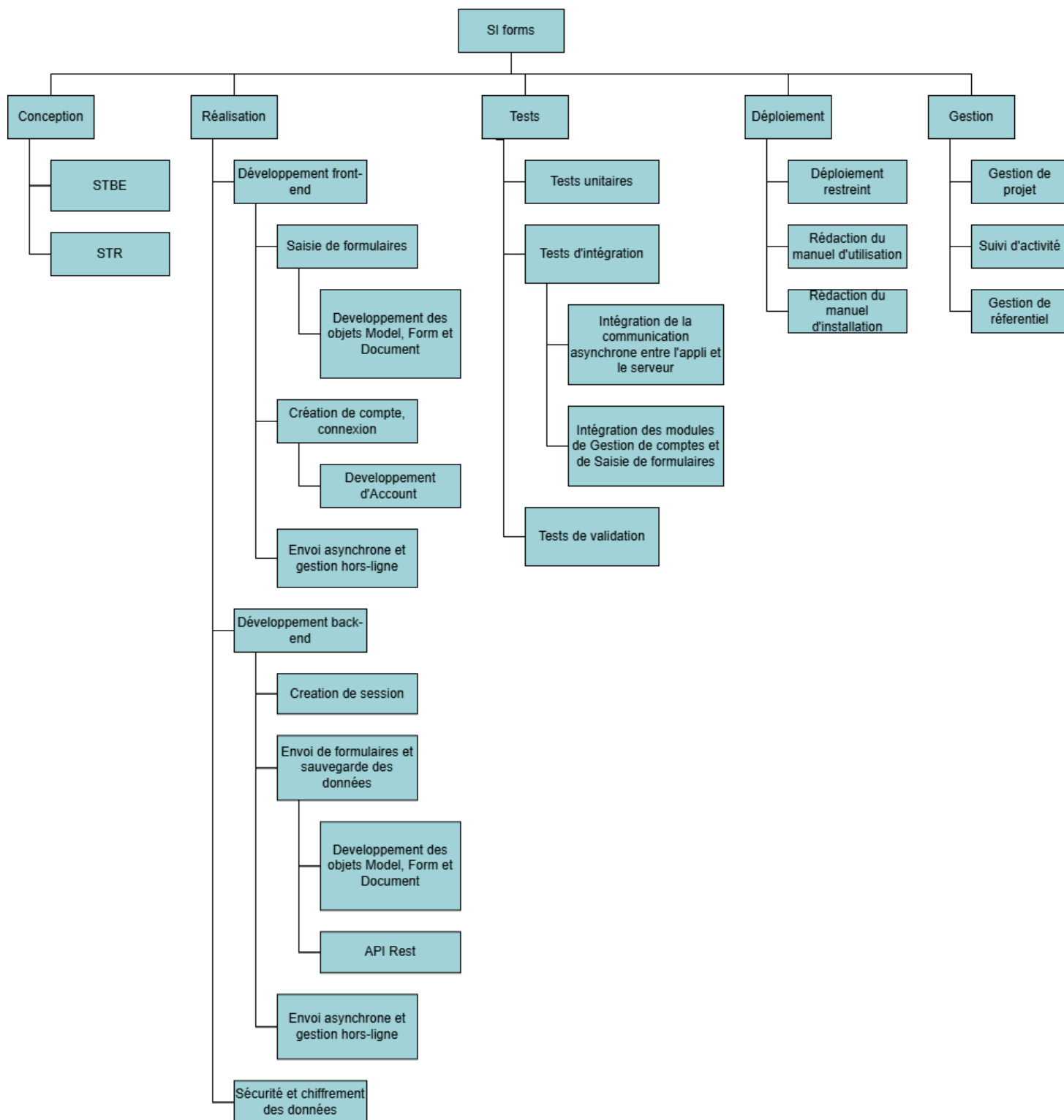


Écran de modification d'une réponse enregistrée : L'utilisateur peut modifier une réponse enregistrée.



5. Planification

5.1. Work Breakdown Structure (WBS)



5.2. Tâches

Détail des tâches à réaliser pour le projet. Le temps estimé est en jours par personne.

5.2.1. Conception

Nom de la tâche	Responsible	Description	Temps estimé	Ressources
STBE	Adèle	Documenter les exigences fonctionnelles et non fonctionnelles du système.	34 j/p	Typst, Figma, LibreOffice, GoogleSheets, Graphviz
STR	Léa	Décrire l'architecture, les technologies et les méthodes d'implémentation du système.	7 j/p	Typst

5.2.2. Réalisation

Nom de la tâche	Responsible	Description	Temps estimé	Ressources
Saisie de formulaires	Léa	Frontend : Implémenter des formulaires pour saisir les informations des utilisateurs.	5 j/p	React Native, JSON
Création de compte, connexion	Adèle	Frontend : Ajouter des fonctionnalités de création de compte et de connexion.	4 j/p	React Native, base de données
Envoi asynchrone et gestion hors-ligne	Kevin	Permettre l'envoi asynchrone des formulaires du serveur vers l'application, et une utilisation hors-ligne.	9 j/p	React Native
Sauvegarde des données	Léa	Backend : Gérer la sauvegarde des réponses aux formulaires.	3 j/p	SQL
Sécurité et chiffrement des données	Kevin	Implémenter des mesures de sécurité pour protéger les données.	3 j/p	Bibliothèque Django

5.2.3. Tests

Nom de la tâche	Responsible	Description	Temps estimé	Ressources
Tests unitaires	Léa	Tests de chaque module individuellement.	4 j/p	/
Tests d'intégration	Kevin	Vérifier l'interaction entre les différents modules	5 j/p	/
Tests de validation	Adèle	Valider que le produit final respecte les exigences utilisateurs	2 j/p	/

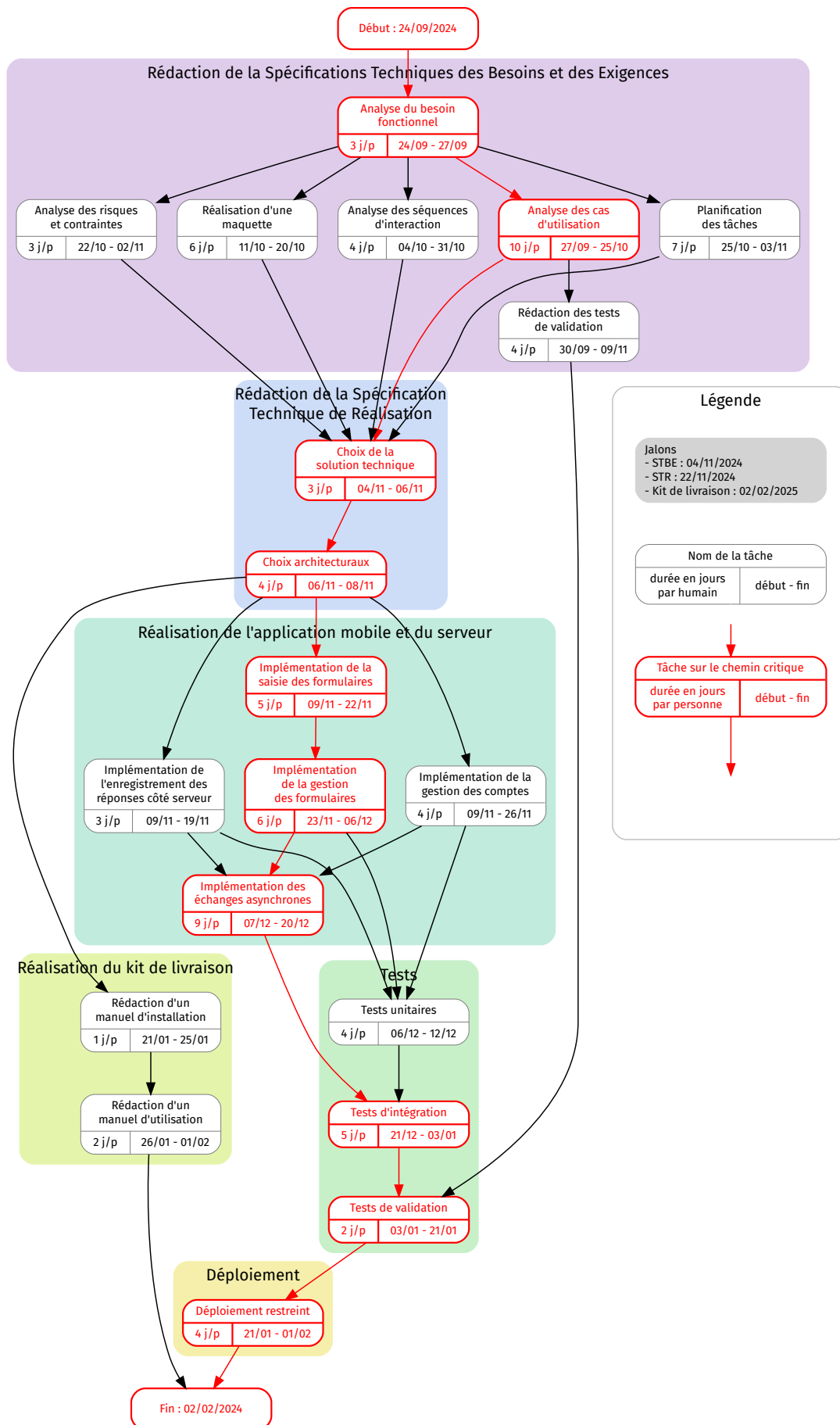
5.2.4. Déploiement

Nom de la tâche	Respon- sable	Description	Temps estimé	Ressources
Déploiement restreint	Léa	Déploiement limité pour les tests finaux en conditions réelles.	4 j/p	/

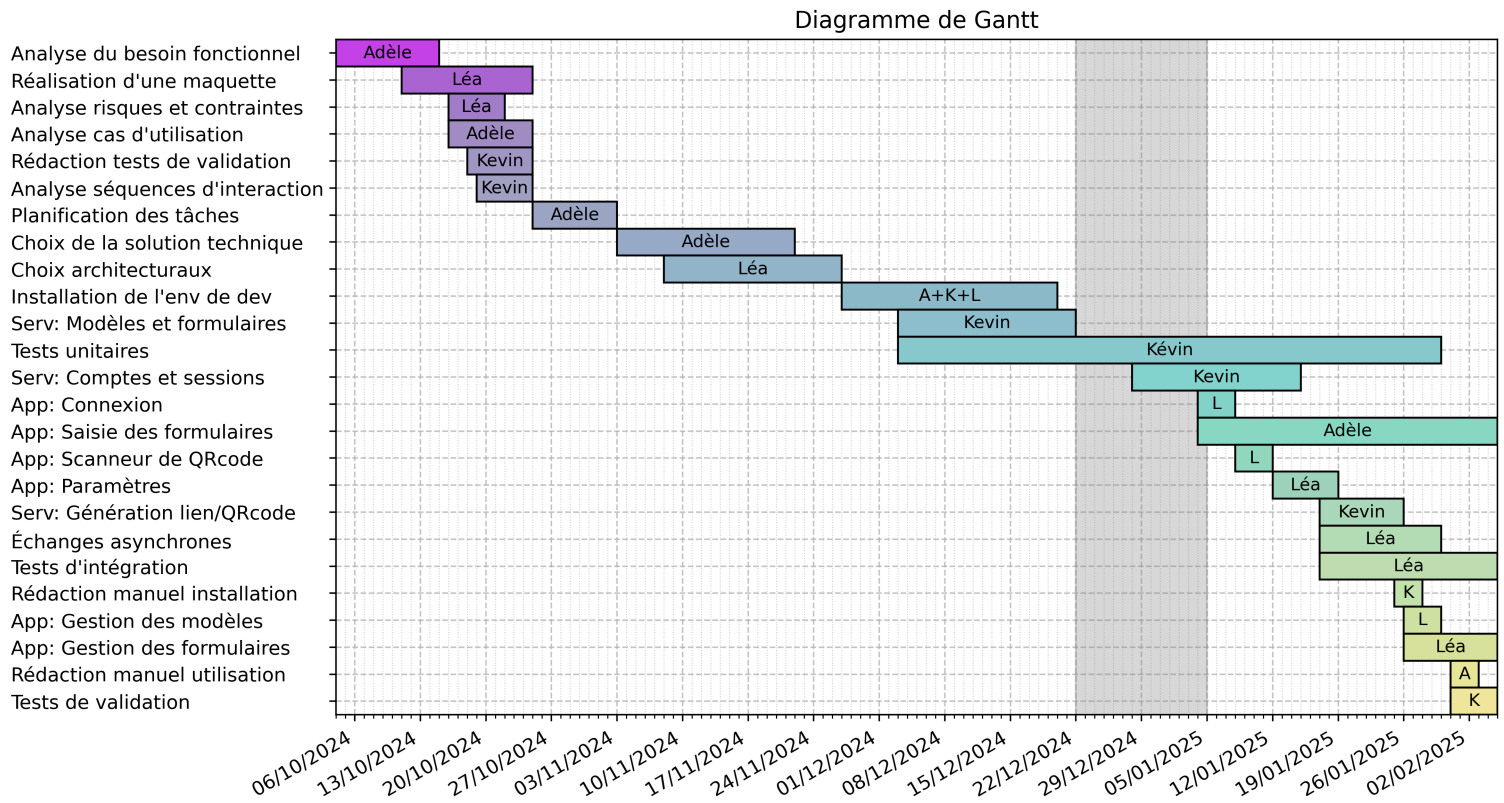
5.2.5. Gestion

Nom de la tâche	Respon- sable	Description	Temps estimé	Ressources
Gestion de projet	Adèle	Suivi de l'avancement du projet et gestion des équipes.	/	Discord
Suivi d'activité	Kevin	Assurer le suivi régulier des activités de développement.	/	/
Gestion de référentiel	Léa	Gérer les versions des documents et des codes sources.	/	Google Drive, Typst, Github

5.3. Program Evaluation and Review Technique (PERT)



5.4. Diagramme de Gantt



6. Annexes

6.1. Lexique

6.1.1. Termes fonctionnels

- **entrée de formulaire** : Ensemble de réponses à un formulaire.
- **formulaire** : Ensemble de questions à réponses textuelles ou cases à cocher (à choix multiples ou unique) et des champs d'insertion de documents photos.
- **formulaire dynamique** : Formulaire dont le nombre et le contenu des questions change en fonction des réponses aux questions précédentes. Par exemples :
 - La question "Quelle sont les caractéristiques de la prise électrique ?" est affichée si l'utilisateur a répondu "Oui" à la question "Y a-t-il une prise électrique ?".
 - La question "Quel est la taille de l'armoire électrique ?" est affiché 5 fois quand l'utilisateur a répondu "5" à la question "Combien y a-t-il d'armoires électriques ?".
 - Le bloc de questions 12 à 15 s'affiche 3 fois si l'utilisateur a répondu « 3 » à la question 11.
- **localisation de l'application** : Processus d'adaptation de l'application à une langue spécifique.
- **modèle de formulaire** : Un modèle de formulaire correspond à un ensemble spécifique de questions, formant un formulaire. Deux formulaires du même modèle ont les mêmes questions et les mêmes règles qui régissent l'apparition de ces questions (cf. "formulaire dynamique"). Il existe pour le moment 3 modèles de formulaires :
 - le formulaire de visite technique,
 - le formulaire d'installation,
 - le formulaire d'informations du site d'intervention.
- **site d'intervention** : C'est le lieu d'installation du compteur électrique Smart X. Parfois raccourci en « site ».

6.1.2. Termes techniques

- **authentifier** : Démarrer une session d'utilisation avec son compte utilisateur.
- **backend** : La partie d'une application qui gère la logique métier, les bases de données et les interactions côté serveur, invisibles pour l'utilisateur.
- **framework** : Ensemble de composants logiciels permettant de simplifier le développement logiciel en définissant une base d'architecture.
- **frontend** : La partie visible d'une application ou d'un site web avec laquelle l'utilisateur interagit directement, comprenant l'interface utilisateur.
- **snackbar** : Élément d'interface graphique informant l'utilisateur. Court message affiché dans l'application, dans une pop-up temporaire au premier plan.

6.1.3. Technologies

- **Dart** : Un langage de programmation développé par Google, principalement utilisé pour le développement d'applications avec le framework Flutter. C'est un langage orienté objet, optimisé pour la performance, qui supporte la compilation en code natif et en JavaScript pour les applications web.
- **Django** : Un framework web Python complet et structuré, conçu pour faciliter le développement rapide et sécurisé d'applications web, avec des fonctionnalités intégrées pour la gestion des bases de données, l'authentification, et le routage.

- **Flask** : Un framework Python très léger pour créer des applications web, connu pour sa simplicité et sa flexibilité, offrant des fonctionnalités de base pour gérer les requêtes HTTP et le routage sans imposer de structure rigide.
- **Flutter** : Un framework de Google permettant de créer des applications natives multiplateformes (iOS, Android, web, desktop) en Dart, offrant une interface utilisateur très réactive.
- **JavaScript** : Un langage de programmation principalement utilisé pour ajouter des fonctionnalités dynamiques et interactives aux pages web, exécuté côté client ou côté serveur.
- **PostgreSQL** : Un système de gestion de base de données relationnelle open-source, supportant les transactions, les requêtes complexes, et les types de données avancés.
- **React** : Une bibliothèque JavaScript développée par Meta pour construire des interfaces utilisateur pour des applications web à pages dynamiques, en utilisant des composants réutilisables.
- **ReactNative** : Un framework de Meta permettant de développer des applications mobiles natives pour iOS et Android en utilisant JavaScript et React.

6.2. Bibliographie

6.2.1. Sources pour la comparaison des technologies

1. [ReactNativ at Airbnb](#), blog-post sur Medium, 19/06/2018
2. [Flutter ou React Native pour développer une application ?](#), blog-post sur Citronnoir, 2021
3. [Flask vs Django: Let's Choose Your Next Python Framework](#), blog-post sur Kinsta, 05/06/2023
4. [10 Advantages of Using Django for Web Development](#), blog-post sur Inoxoft, 10/07/2024
5. [What is Flask? Benefits and uses](#), blog-post sur Mytaskpanel
6. [Une API REST, qu'est-ce que c'est ?](#), blog-post sur RedHat, 02/02/2024
7. [Go-Back-N ARQ](#), Wikipédia FR, article web visité le 09/12/2024
8. [Ruby on rails vs Django](#), Wikipédia FR, article web visité le 13/12/2024

6.2.2. Documentation des technologies

1. [Django documentation](#)
2. [React Native documentation](#)
3. [Python documentation](#)
4. [PostgreSQL documentation](#)
5. [Expo documentation](#)