**SPECIFICATIONS TECHNIQUES**

**Projet :** Mise en place d’un système centralisé de traitement automatique des Workflow.

12-08-2022

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| HISTORIQUE DES VERSIONS | | | | |
| VERSION | APPROUVÉ PAR | DATE DE RÉVISION | DESCRIPTION DU CHANGEMENT | AUTEUR |
| 0.1.0 |  |  | Document initial | DEKI moïse |
| 1.1 |  | 06/09/2022 | Mise à jour | DEKI moïse |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

1. **DESCRIPTION DU PROJET**

Notre suite applicative est l’ensemble de trois applications communicantes entre elles. Elle permettre de mettre en place d’un système centralisé de traitement automatique des WorkFlows.

Ce système permettra au client de soumettre un ou plusieurs formulaires relatifs à nos produits d’une part, elle permettre aussi aux utilisateurs de la banque d’initier le processus de validation d’autre part.

Notons que ce système de validation sera géré par un moteur de workflow.

1. **Architecture globale du projet**

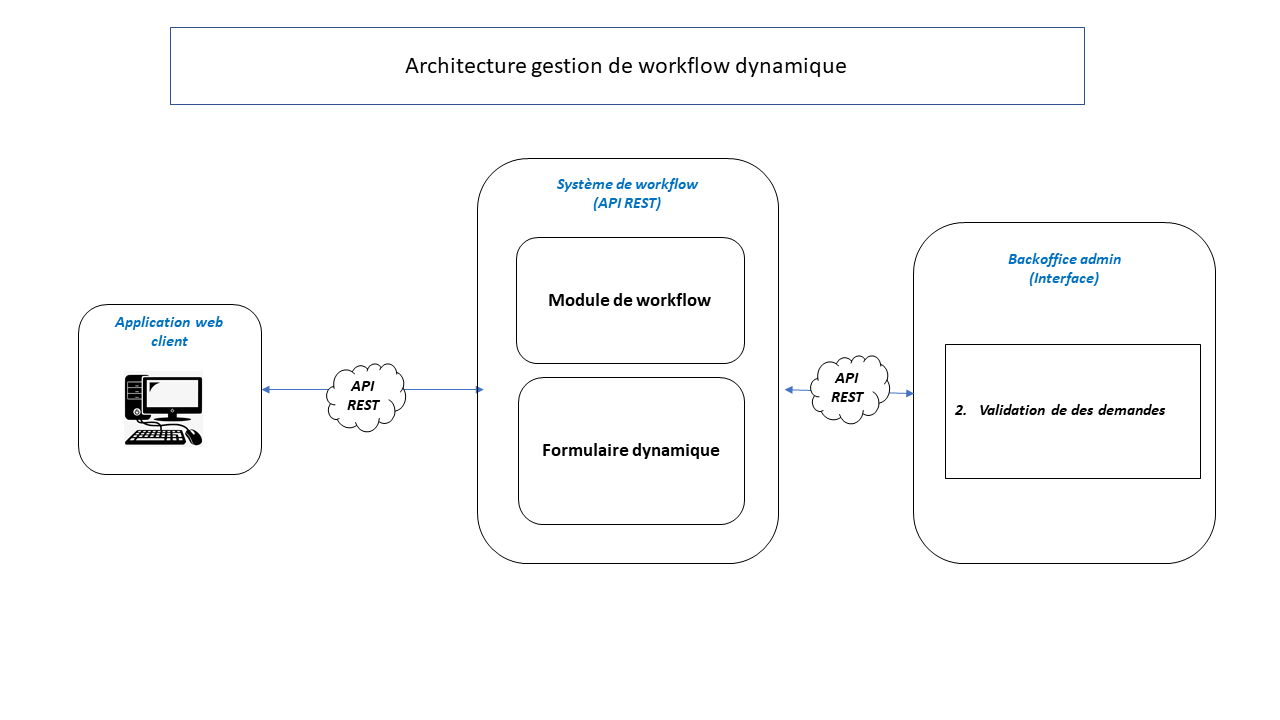
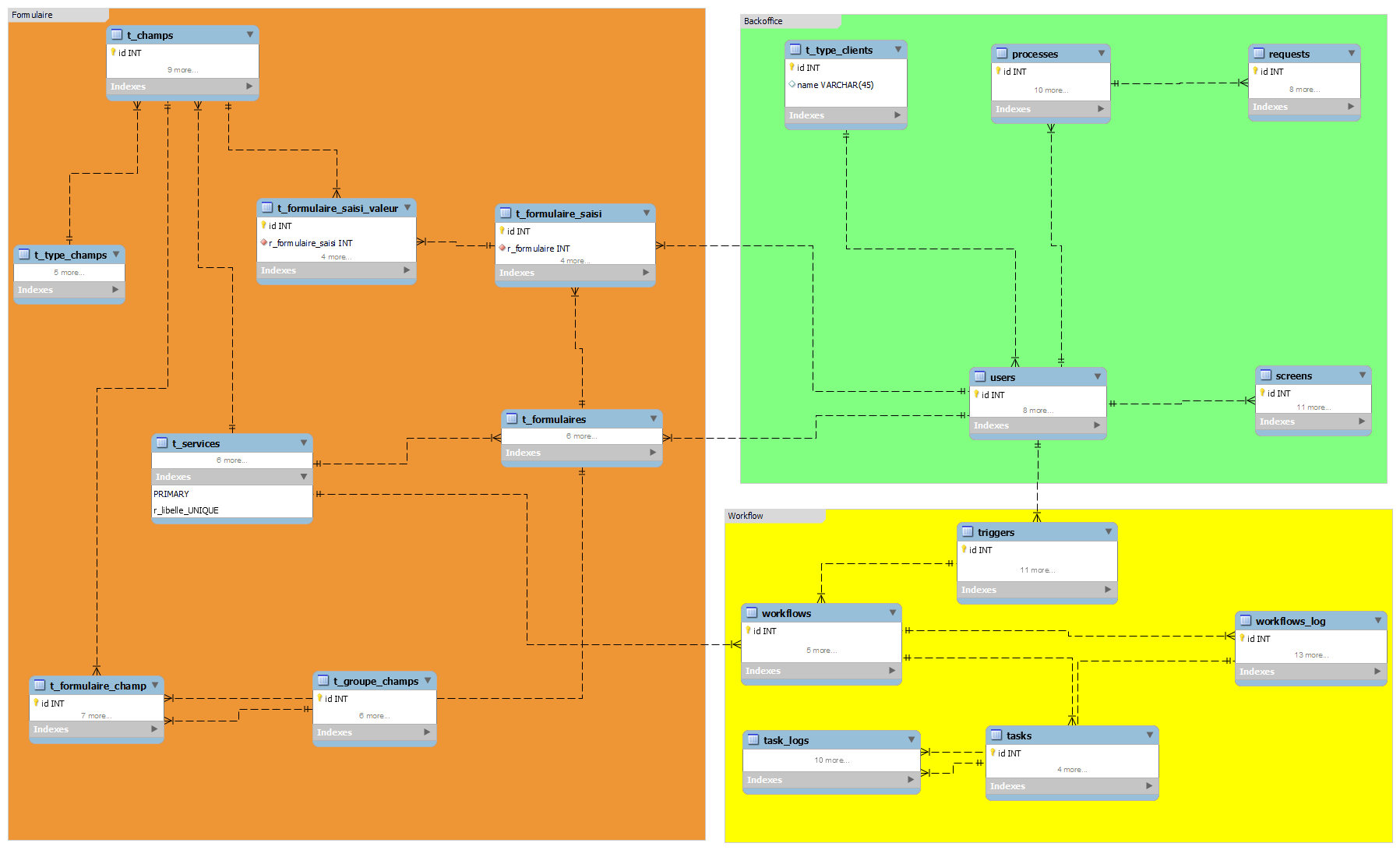


Figure  : Architecture technique

1. **MODELE RELATIONNELLE DES DONNEES**



En fonction des contraintes, il convient de concevoir les applications suivantes :

1. Application de gestion du workflow,
2. Backoffice administration,
3. Application web front-end client
4. **MODELE LOGIQUE DE DONNEES**

t\_type\_clients ( id, name , r\_description, created\_at , updated\_at)

t\_clients ( id , name , email , type , r\_description, created\_at , updated\_at )

users ( id , name , email\_verified\_at , password , remember\_token , created\_at , updated\_at , t\_clients\_id)

triggers ( id , type , name , queueable , data\_fields , conditions , workflow\_id , pos\_x , pos\_y , created\_at , updated\_at , users\_id)

t\_services ( id , r\_libelle , r\_description , r\_utilisateur , created\_at , updated\_at )

workflows ( id , t\_triggers\_id , t\_services\_id , name , created\_at , updated\_at)

workflows\_log ( id , workflows\_id , elemen\_id , elemen\_type , triggerable\_id , triggerable\_type , name , status , message , databus , start , end , created\_at , updated\_at )

tasks ( id , workflows\_id , paren\_id , paren\_type, taskscol)

t\_formulaire ( id , product\_id , users\_id)

t\_groupe\_champs ( id , product\_form\_id)

task\_logs ( id , task\_id , workflows\_log\_id , name , status , message , start , end , created\_at , updated\_at)

t\_formulaires ( id , r\_service , r\_code , r\_libelle, r\_description , users\_id)

t\_type\_champs ( id , r\_libelle, r\_description , created\_at , updated\_at)

t\_champs ( id , r\_service , r\_type\_champ , r\_libelle , r\_description , r\_taille , r\_options JSON NULL, created\_at , updated\_at , t\_champscol)

t\_formulaire\_champ ( id , r\_formulaire , r\_groupe\_champ , r\_champ , r\_rang , r\_statut TINY, created\_at , updated\_at)

t\_formulaire\_saisi ( id , r\_formulaire , r\_client , r\_libelle , created\_at , updated\_at)

t\_formulaire\_saisi\_valeur ( id , r\_formulaire\_saisi , r\_champ , r\_valeur , r\_created\_at , r\_updated\_at)

processes ( id , name , slug , description , archive , bpmn , svg , isValidModel , created\_at , updated\_at , users\_id)

requests ( id , bpmn , data , tokens , status TINY, created\_at , requestscol , updated\_at , processes\_id)

screens ( id , title , slug , description , type , config , computed , custom\_css , watchers , created\_at , updated\_at , users\_id)

1. **CHOIX TECHNOLOGIQUES**
2. **Application de gestion du workflow**

L’application de gestion du workflow sera développée en PHP et utilisera le Framework Laravel (V8)

Le système de workflow à en son sein trois principale fonctionnalités :

1. **Le module de configuration du workflow**

Le module de configuration de workflow se base sur package **42coders workflow**, il permettra de configurer les workflows.

Les fonctionnalités du module **42coders workflow** qui seront utilisés sont :

1. Les triggers
2. Les tâches
3. **Le module de génération du formulaire dynamique**

Le module de génération des formulaires sera développé d’une part en PHP avec le Framework pour l’exposition des API et d’autres part en javascript avec le Framework Angular.

1. **Le module de gestion des APIs**

Le module de gestion des APIs, exposera les routes aux application client et backoffice. Entre autres la gestion des accès, la gestion dynamique des formulaires, la gestion des workflows…

1. **Backoffice administration ou système centralisé**

L’interface backoffice permettra la configuration des workflows ainsi que la validation les demande soumis par le client.

La technologie utiliser pour la conception de l’application est Angular (V13), qui est un Framework JavaScript et présente l’avantage d’être un SPA.

Le système centralisé comprend donc deux applications :

1. **L’application de gestion ou d’administration**

L’application de gestion sera accessible via une page web. Sont principale rôle sera de d’administrer toute l’application de workflows, c’est-à-dire créer :

* Créer les utilisateurs,
* Valider les accès utilisateurs et clients,
* Gérer les rôles et les permissions,
* Configurer les formulaires dynamiques
* Configurer les workflows
* Etc.

1. L’API

L’API fait office d’interface entre le système centralisé et les application web client et web administration.

Son rôle principal est d’exposer les routes attendues par les applications web client et web administration.

L’API est privée et uniquement accessible par un client authentifié (Bearer).

Le route du format de l’API est du JSON.

**NB** : *notons que l’échange des données via l’API se fait tout en cryptant les données qui transitent sur le réseau.*

*Le module de cryptage utilisé est* ***crypto-js*** *qui utilise une clé de cryptage asymétrique définie par le développeur.*

Exemple de retour de l’API :

**Exemple de réponse crypté de l’API**

|  |  |  |  |
| --- | --- | --- | --- |
| Route | Verbes | Format de retour | Retour fourni par l’API |
| /oba/utilisateurs | GET | JSON | **{**  "ct": "8VWRAMWQKFe8ologXuz+ZsKCkYyXtOU=",  "iv": "1c37f566c205324c4550a0d27146a5e0",  "s": "a0708e236f939338"  **}** |

**Exemple de réponse non crypté de l’API**

|  |  |  |  |
| --- | --- | --- | --- |
| Route | Verbes | Format de retour | Retour fourni par l’API |
| /oba/utilisateurs | GET | JSON | **[**  **{**  **"id": 31,**  **"r\_nom\_produit": "Crédit auto",**  **"r\_description": "Crédit auto",**  **"r\_status": true,**  **"r\_created\_by": 31,**  **"r\_updated\_by": 31,**  **"created\_at": "2022-08-24T17:33:02.000000Z",**  **"updated\_at": "2022-09-06T13:59:05.000000Z",**  **"deleted\_at» : null**  **}**  **]** |

1. **Les application web Front-end**

La technologie utiliser pour la conception de ces application est Angular (V13).

Le front-end comprends deux applications :

1. **L’application front-end client**

Il est accessible après authentification (Bearer).

Il aura deux tâches principales :

* Générer le formulaire dynamique et le présenter sur la vue,
* Permettre au client de soumettre le/les formulaires au backend

L’application frontend client est destiné aux clients, il permettra aux clients de soumettre des données via les formulaires dynamiques configurés depuis le système centralisé.

L’échange des donnés est chiffré et se fait via API exposé par le système centralisé.

1. **L’application front-end d’administration et de validation de workflows**

Il est accessible après authentification (Bearer).

Cette application à pour tâche principale de valider les workflows tout en respectant le ou les règles de gestion qui lui sont attribués.

L’application est destinée uniquement aux utilisateurs d’Orange Bank Africa.

1. Sécurité

Les données échangées via API seront chiffrées via le module **Cryptojs-aes-php** du côté l’API et **Cryptojs** du côté de Angular.

L’Accès aux ressources se fera via un token

***Sources :***

<https://workflows.42coders.com/>

<https://github.com/42coders/workflows>