le Cairn
Monnaie locale & citoyenne

# Le plan de tests de l'application web B2B

## Contents


**1 Présentation du projet** **2**
- 1.1 Architecture logicielle . . . 2
  - 1.1.1 La solution logicielle de gestion de paiements : Cyclos . . . 2
  - 1.1.2 Architecture globale . . . 5

**2 La mise en place des tests** **5**
- 2.1 Jeu de données . . . 5
- 2.2 Modèle de charge . . . 6
- 2.3 Type de tests . . . 6
- 2.4 Les scénarios fonctionnels . . . 6

**3 Résultats et bilan des tests** **8**

**Introduction**

Ce document a pour objectif la présentation des besoins du processus de tests. On rappellera d'abord le projet et ses objectifs, avant de décrire l'architecture technique puis les types de tests à réaliser ainsi que les scénarios cibles. Pour ce faire, on s'appuie sur le cahier des charges. Le but du processus de tests est de cibler les points sensibles de l'architecture d'un point de vue Utilisateur (temps de réponse, erreurs), Métier(nombre d'utilisateurs virtuels, processeurs), et Technique(consommation de ressources, nombre de requêtes en base de données Symfony/Cyclos). Nous nous concentrerons sur les fonctionnalités accessibles par le plus grand nombre des parties prenantes (les professionnels / éventuellement les groupes locaux)

# 1 Présentation du projet

**Résumé**

La monnaie locale du bassin de vie grenoblois, le cairn, d'association éponyme, cherche à étendre son réseau d'acteurs et à diversifier ses possibilités d'utilisation. Disponible à ce jour sous forme de coupons-billet uniquement, l'objectif est la mise en place d'un outil de monnaie dématérialisé B2B full web, permettant l'ouverture du réseau à des professionnels n'acceptant pas le liquide ou les très petits montants(inférieurs à 1 cairn).

L'outil doit assurer la plupart des services bancaires classiques - transaction, consultation de compte, conversion entre devises - mais doit aussi, à terme, faciliter et améliorer l'analyse de la vitesse d'émission, de circulation de la monnaie ainsi que ses circuits préférentiels.

## 1.1 Architecture logicielle

### 1.1.1 La solution logicielle de gestion de paiements : Cyclos

**Présentation** La solution logicielle retenue pour la gestion des paiements se nomme Cyclos. Elle répond à l'ensemble des spécifications concernant la gestion des paiements décrites dans le périmètre fonctionnel , mais aussi aux objectifs et au modèle économique potentiel(long terme) ainsi qu'à la possibilité de partage et/ou interopérabilité avec d'autres monnaies locales. Logiciel de paiement en ligne faisant partie du réseau des Organisations de commerce social (STRO), Cyclos propose de nombreuses fonctionnalités utiles à plusieurs types d'institutions : banques, systèmes barters, monnaies locales... Elle compte plus de 1500 systèmes de paiements à son actif.

Il existe plusieurs versions de Cyclos disponibles à ce jour, avec des avantages et des inconvénients différents :

- Cyclos 3 est une version libre, mais n'est plus maintenue
- Cyclos 4 est une version propriétaire, dont il est possible de récupérer la base de données pour exploitation, maintenue à jour par la société.

**Architecture logicielle**

L'architecture logicielle de Cyclos est séparée en deux grandes configurations : système et comptes.

La configuration système concerne l'agencement des utilisateurs de la plateforme. On en distingue principalement trois natures :

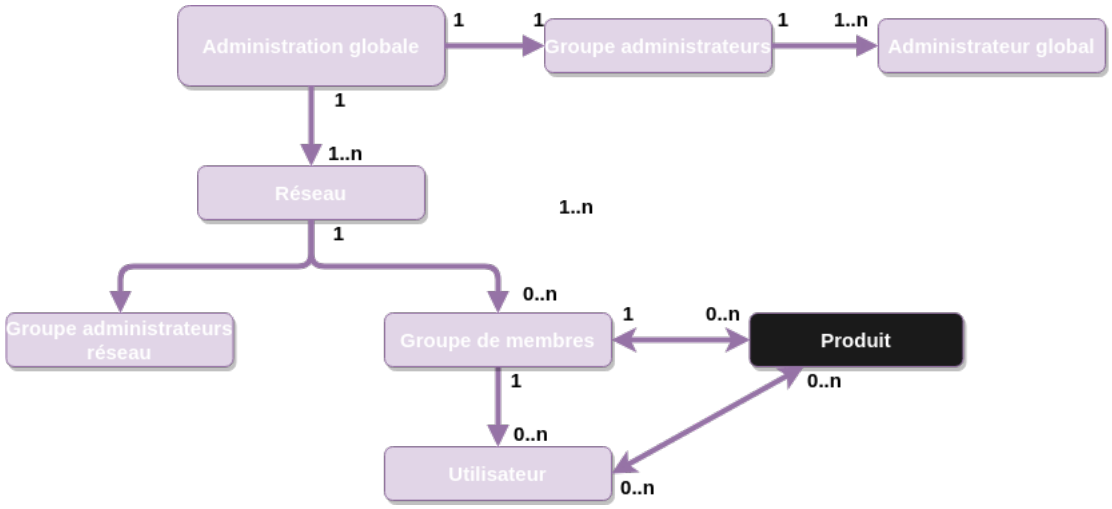


Figure 1: Cette figure montre une version simplifiée de la configuration système de Cyclos

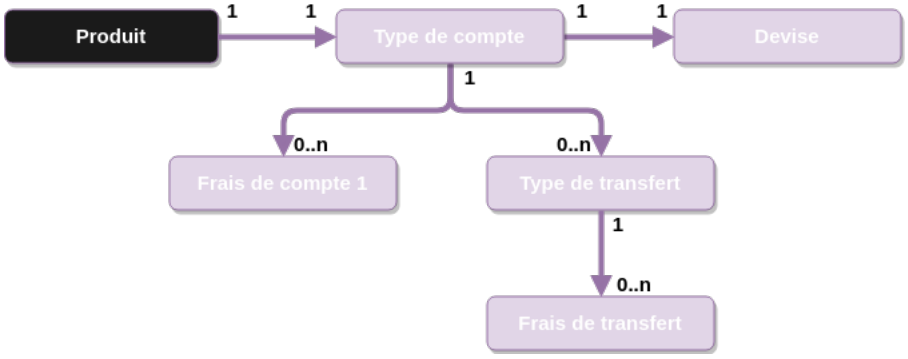


Figure 2: Cette figure montre une version simplifiée de la configuration des comptes de Cyclos

- les administrateurs globaux : ils peuvent créer, accéder, interdire différents réseaux et modifier la configuration de l'ensemble du système -types d'authentification et leurs paramètres, types d'accès, ...
- les administrateurs réseaux : un réseau est un environnement indépendant dans le système. Il est composé de membres qui vont s'échanger des unités de compte. Ainsi, le rôle des administrateurs réseaux est de gérer les utilisateurs et leurs droits, ainsi que la gestion de la configuration de comptes - les devises et les types de comptes.
  Un administrateur global est un administrateur réseau, pour tous les réseaux.
- les membres : c'est la nature d'utilisateurs avec le moins de droits dans le système. Ils ne peuvent administrer que leur propre espace membre.

La nature de chaque utilisateur est donc définie par le type de groupe auquel il appartient.

La configuration des comptes concerne l'activité d'échanges inter-comptes. Les *devises* sont les différentes unités de compte transférables entre les comptes des membres. Les *types de comptes* sont les éléments centraux de la configuration, chacun étant associé à une devise. De même que pour les groupes, il existe des *types de comptes* de natures système et membre :

- les types de comptes "membre" : même si, dans la configuration, on définit un unique

type de compte, chaque membre en aura une instance propre s'il possède un *Produit* le permettant.

- les types comptes de nature système : contrairement aux types de comptes "membre", il n'y en a qu'une instance de chaque dans le système, accessible par les administrateurs réseaux. Il s'agit en fait des comptes de l'organisation. Tous ont logiquement accès aux mêmes comptes.

Chaque type de compte peut contenir des frais de compte et des *types de transferts* décrivant les flux d'unités de compte sortant.

Exemple: le réseau contient deux types de comptes membre $A$ et $B$. Les *types de transferts* de $A$ sont l'ensemble des flux de $A$ vers un autre type de compte. Ainsi, si le *type de transfert* de $A$ vers $B$ n'est pas défini, il sera impossible d'effectuer un virement d'une instance de $A$ vers une instance de $B$.

Le lien entre la configuration système et la configuration de compte est la classe *Produit*(en noir sur la figure 2), définissant, pour chaque utilisateur ou groupe d'utilisateurs, les permissions et les règles d'accès à la plateforme, à ses utilisateurs et à ses comptes. Cette classe joue donc un rôle central dans le système. Plusieurs instances de la classe *Produit* peuvent être assignées à un même groupe d'utilisateurs dits "membres". Les administrateurs ont leur propre *Produit administrateur* qui définit leurs droits dans le réseau. Ce produit est modifiable par les administrateurs globaux. Ainsi, c'est dans cette classe qu'on spécifie si les administrateurs d'un réseau peuvent accéder au compte de l'organisation, ou s'ils peuvent effectuer un virement à la place d'un membre de leur réseau, entre autres.

Enfin, le dernier élément qui semble indispensable à la compréhension de l'architecture logicielle de Cyclos est le système de canaux. Un canal est simplement un moyen d'accéder à Cyclos.
Le moyen le plus simple se nomme *web principal*. Il s'agit d'accéder à l'interface web de Cyclos dans son navigateur, de se connecter et d'accéder directement à son propre espace membre, avec l'interface utilisateur de base.
Il en existe d'autres, comme les *services web*, *application mobile* ou les *opérations par sms*. Pour chacune des instances de *Type de transfert* par exemple, il est possible de définir les canaux autorisés. Ainsi on peut autoriser les paiements d'adhérents vers le compte de l'Organisation via les services web, mais les interdire via une application mobile. Cela est valable pour d'autres classes. Tout est donc question de choix de configuration.

Il s'agit dans la Fig. 2 et dans la partie ci-dessus de présenter une version simplifiée d'un système bien plus complexe, chaque classe contenant son propre lot d'options et de configuration.

**Les services web** Les services web est le canal d'accès à Cyclos utilisé par l'application. Ils renvoient des données à l'application au format `stdClass`, la classe `Objet` la plus basique de PHP.

Cyclos propose deux interfaces de services web permettant de tirer avantage de leurs fonctionnalités existantes : une API REST et une WEB-RPC. Plus orienté "opérations", le WEB-RPC va envoyer une requête contenant le nom du service en cours, ainsi que l'action demandée, et

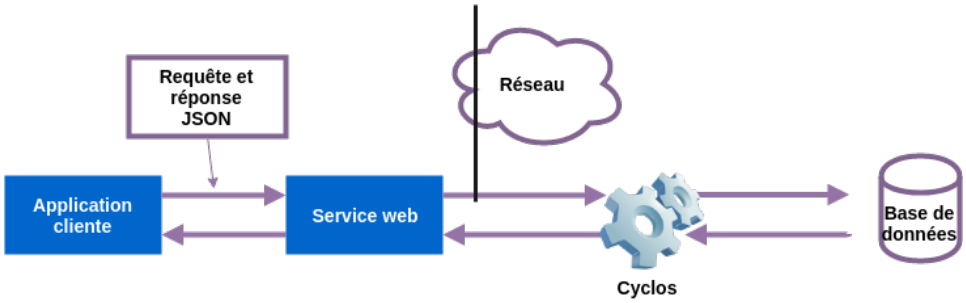


Figure 3: Cette figure est un schéma descriptif de l'utilisation d'un service web

un identifiant vers la ressource concernée. Contrairement à la philosophie REST, le client PHP est un ensemble de services(des classes PHP) : UserService, GroupService, ... Leurs méthodes correspondant à des opérations.

### 1.1.2 Architecture globale

A partir de l'analyse effectuée dans les sous-sections précédentes(**??** et **??**), un workflow simplifié du fonctionnement de l'application web peut être présenté comme suit.

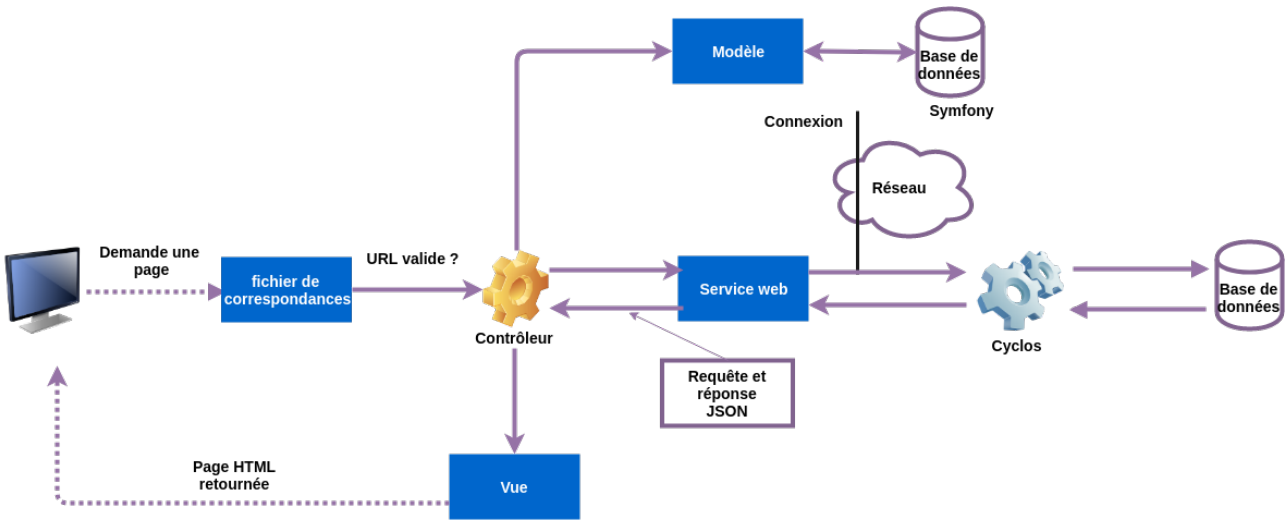


Figure 4: Cette figure décrit de façon simplifiée l'architecture globale de l'application web

# 2 La mise en place des tests

## 2.1 Jeu de données

Deux approches possibles : copier les données de production / créer les bases de données à partir de rien (génération à partir de tableurs etc.). L'extraction des données de production n'est pas particulièrement intéressante étant donné que l'application n'a pas pour objectif d'évaluer des "préférences" des utilisateurs, excepté la possibilité d'enregistrer des bénéficiaires. De plus, il nous faut des garanties sur les valeurs de certains éléments (mots de passe, carte de sécurité). Construire les bases de données **ex-nihilo** semble donc être l'option la plus appropriée, en prenant garde à respecter la cohérence des bases de données Cyclos / Symfony.

## 2.2 Modèle de charge

## 2.3 Type de tests

**Tests fonctionnels** Une fois les tests fonctionnels validés, les tests de performance de l'application peuvent être réalisés.

**Test de charge**

**Test de performance**

**Test de dégradation**

**Test de montée en charge**

**Test de stress**

## 2.4 Les scénarios fonctionnels

**Inscription**

- Accès à la page d'inscription
- Écran de choix du type d'inscription ( pro / particulier )
- Saisie du formulaire d'inscription et soumission

**Connexion**

- Accès à la page de connexion
- Saisie du login/email, du mot de passe et soumission
- Accès à la page d'accueil

**Consultation de compte**

- Accès à l'aperçu des comptes
- Accès aux opérations d'un compte
- Filtrage des opérations par date et/ou montant et/ou mot-clé
- Téléchargement du RIB Cairn
- Accès au téléchargement du relevé de comptes
- Saisie du formulaire (sélection de compte + format + intervalle de temps) et soumission
- Récupération du relevé de compte

**Gestion(ajout / suppression) des bénéficiaires**

- Accès aux fonctionnalités liées aux virements
- Accès à la gestion des bénéficiaires
- Ajout d'un bénéficiaire
- Validation de la carte de sécurité Cairn
- Saisie du formulaire et validation
- Accès à la gestion des bénéficiaires
- Suppression et validation de la suppression du bénéficiaire

**Réalisation d'un virement**

- Accès aux fonctionnalités liées aux virements
- Accès aux virements uniques à effectuer
- Virement vers un bénéficiaire enregistré(recherche de bénéficiaire en plus)
- Saisie et soumission du formulaire de virement (montant faible pour pouvoir le répéter souvent)
- Validation du virement

**Consultation des virements**

- Accès aux fonctionnalités liées aux virements
- Accès aux virements uniques à gérer
- Voir le détail d'un virement
- Téléchargement de l'avis d'opération du virement
- Récupération du document au format PDF

**Consultation du profil**

- 
- 

**Modification du profil**

- Accès à mon profil
- Modification de mon profil
- Validation de la carte de sécurité Cairn
- Saisie du formulaire et validation
- Accès à mon nouveau profil

**Modification du mot de passe**

- Accès à mon profil
- Modification de mon mot de passe
- Validation de la carte de sécurité Cairn
- Saisie du formulaire et validation

**Activation / Blocage d'un utilisateur**

- Accès au profil d'un utilisateur
- Blocage de l'utilisateur
- Validation de la carte de sécurité Cairn
- Confirmation de l'action
- Activation de l'utilisateur
- Confirmation de l'action

# 3 Résultats et bilan des tests

Préciser le contexte technique du plan de test : production / préproduction / dédié . Il est préférable que l'environnement de test soit identique à l'environnement de production.
Trois vues de résultats : Utilisateur (temps de réponse, erreurs), Métier(nombre d'utilisateurs virtuels, processeurs), et Technique(consommation de ressources, nombre de requêtes en base de données Symfony/Cyclos).

Le livrable sera un dossier contenant des sous-dossiers pour chaque scénario de tests. Chacun de ces sous-dossiers contient des fichiers au format CSV pour chaque type de tests. Enfin, chaque fichier CSV contient des tableaux de données filtrables par 'action métier', contenant des données utiles à chacune des vues énoncées ci-dessus.