**Cycle de vie d’un projet** Version 1.0

* Directeur Générale des études technologiques: Faical Yannick P. Congo
* Responsable Multimédia et Design: Dody Baruk
* Responsable Réseaux et Liens serveurs: Elson Sekle
* Directeur Financier et Ressources Humaines: Maimounata Tiendrebeogo

# Contents

[Contents](#h.tpe1uj6fpvzf)

[1. Preface](#h.i6dkd55uymqp)

[1.1. A propos de ce guide](#h.l0k72tf7g22w)

[1.2. Audience cible](#h.pkjeckfz9ssf)

[1.3. Historique de revision](#h.jllrsyn5vbi8)

[2. Analyse et conception](#h.5uqfiu3lklj2)

[2.1. Analyse-](#h.r6t5804z26ky)

[2.1.1. Backend - Web service](#h.173uapiwo85w)

[2.1.2. Frontend - Client](#h.dc8mshnu8gsg)

[2.2. Conception](#h.ks8c4glcvolo)

[2.2.1. Diagramme des actions backend](#h.52u26efuircj)

[2.2.2. Diagramme des actions frontend](#h.rzv3bc6r72yz)

[2.2.2. Definition des modeles des donnees](#h.krox27w9ysmh)

[3. Developement](#h.gq9idm7kgj5j)

[3.1. Implementation-](#h.57bd0is0zmv3)

[3.1.1. Backend](#h.taiyscr6qmk3)

[3.1.2. Frontend](#h.m0n4mcxai0ue)

[3.2. Design](#h.7e3f3lxrm7hc)

[3.2.1. Web](#h.vsuts3g5wghq)

[3.2.2. Natif](#h.7gxgufq7004b)

[4. Test](#h.4wwbvavrf4co)

[4.1. Backend](#h.1iedchgq1x9k)

[4.2. Frontend](#h.gk7j3ikqhthw)

[5. Integration et Recette](#h.x1sx7vkgeh11)

[5.1. Documentation-](#h.ag61l9wq39kr)

[5.1.1. Presentation du produit](#h.yldzj3nbvh3)

[5.1.2. Guide d’utilisation](#h.ugbsctoe02ol)

[5.2. Livraison de l’application](#h.59vwyc5kgnbl)

[5.2.1. Livrable externe](#h.w639g139u4vn)

[5.2.2. Livrable interne](#h.1fxfwj24r5e7)

[6. Mise a niveau](#h.o2toxud3nyvl)

# 

# 1. Préface

Ce document a pour intention de définir les processus standard du cycle de vie d’une infrastructure résultant de la réalisation d’un projet chez Kugri. Les infrastructures concernées dans un premiers sont globalement a deux parties. La première est un service web chargée du liens avec les données et des interactions entre les instances de la deuxième partie qui est un client. Nous avons deux variantes courantes de client. Les clients natifs et les clients web. L’avantage des client web c’est leurs accessibilités indépendantes du système d’exploitation du hôte client parce que dépendante uniquement de la présence d’un navigateur. Les clients natifs par contre jouissent d’un niveau d'accès aux propriétés de l'hôte client que les clients web n’ont pas. Toutes ces deux parties constituent l’infrastructure qui fait globalement remplir un cahier de charge particulier d’un projet informatique chez Kugri.

## 1.1. A propos de ce guide

Ce guide met en place les procédures a suivre dans chaque étape du cycle de développement d’une infrastructure Kugri Co. Il représente la valeur sure de l'établissement des standards de la compagnie.

## 1.2. Audience cible

Ce document est destiné a toutes les personnes des équipes d'étude de projets, de développement, de test et d'intégration des infrastructures chez Kugri Co.

## 1.3. Historique de révision

Version 1.0 - Jeudi 30 Janvier 2014 - Faical Yannick P. Congo

# 2. Analyse et conception

Après l'édition du cahier des charges par le responsable de formalisation des Projets (FYPC),

la phase d'analyse et de conception doivent mener a la définition réelle en terme de composants informatiques de l’infrastructure.

2.1. Analyse

Dans cette étape on décide du nom du domaine du backend et aussi de la nature des composants techniques au niveaux du client. L’analyse doit permettre de connaître les spécificités du backend et du frontend. Il est a décidé ici également quelle est la stratégie du client. Le client voudrait il un domaine personnel ou n’en tiens pas rigueur. Voudrait il que l'hébergement soit gérer ou qu’il s’en occupe lui même. Il est aussi a considere que l’infrastructure fasse partie de Kugri Technologies. Le web service vivra sur notre serveur de production et le client web sur notre serveur principal.

### 2.1.1. Backend - Web service

Dans cette partie on doit déterminer quel configuration de server (Tomcat - JVMHost: 7.95|15.95|35.95) choisir après avoir déterminer le nom de domaine. Il est possible que nous incluons d’autres types de serveurs plus tard: Rails, Glassfish, JBoss, Django, etc… Il est aussi possible que le type de web service soit mise en choix plus tard (REST par defaut). Dans un autre cas on n’aura pas a faire cette etude puisque le web service sera sur notre serveur de production.

### 2.1.2. Frontend - Client

Il est a déterminer dans cette étape le type de client a prendre en compte. Allons nous allé dans le natif, le web ou faire de l’hybride. Dans le cas du Web s’agirait-il d’un landing page ou d’une application web complete. Dans le cas du Natif il s’agit généralement d’un client lourd complet. Il faut cependant choisir les plateformes cibles: Desktop/Server (Linux, Windows, Osx, Unix) ou Mobile (Android, iOs, Symbian, WPhone, Embedded,...).

## 2.2. Conception

Dans cette section nous allons définir les procédures a suivre pour concevoir l’infrastructure d’un projet Kugri Co. A la sortie de la conception les développeurs et designer ont une sorte de feuille de route sur la structuration des composants de chaque parties.

### 2.2.1. Diagramme des actions backend

Dans cette partie le concepteur doit définir les différentes actions et sous actions associées par rapport aux différents intervenants digitaux de la partie web service. En effet cette conception tiens compte globalement du client également sachant qu’il est l’initiateur. Il s’agit cependant d’un point de vue backend.

### 2.2.2. Diagramme des actions frontend

Ce diagramme doit définir ce que le frontend doit avoir comme action et sous actions. Ici on parle de Menu et de SousMenu. Il s’agit ici de définir aussi les différents workflows que le client doit et peut avoir.

### 2.2.2. Définition des modèles des données

Dans cette section partant des deux diagrammes précédents le concepteur doit définir la structure des données au niveaux backend mais également au niveau du client. A l’exemple des clés d'accès pour accès automatique après le premier login. Cette description doit donner une vue claire de ce qui doit être stocker en ligne et ce qui doit l'être localement. C’est vraiment la section qui fait merger les deux précédentes et ajoutent les dépendances entre elles.

# 3. Développement

Dans cette partie, allant des deux diagrammes précédents du workflow clients et des modèles de bases de données deux axes peuvent être démarrer. L’implementation des cores (pure code) et le design des IU(Interface Utilisateur).

3.1. Implementation

Cette partie concerne les standards a suivre en terme de développement. Pour ce faire des sources génériques ont été créées et sont mises a disposition pour respecter un modèle d’implementation propre a Kugri Co. Cette vue permet d’avoir un support de contrôle de qualité et d'efficacité. Dans cette partie nous allons présenté les directions a suivre pour l’implementation au niveau du backend mais également celui du frontend.

### 3.1.1. Backend

Au niveau du Backend la première étape est la création de la base de données a partir des modèles de données de la Section Conception du backend. Apres la création du backend. La prochaine étape est de créer les Structures dans Entity. Les entities sont identiques a une table dans la base de données. A l’exception qu’elles n’ont pas de champ id. Après la création des entity if faut créer les loaders. Ce sont des méthodes du style StringToEntity. Ils permettent de pouvoir charger les données des tables dans une liste chaînée accessible en mémoire. Il faut aussi créer aussi les structures globales pour ces données. Toutes ces premières modifications sont a faire dans la classe AbstractEndpoint qui est héritée par tous les autres endpoints. Après ces implementations préliminaires dans le backend, la prochaine etape est la création des endpoints. Vous avez un exemple dans net.kugri.backend.service.rest.serviceName.endpoint nomme EndpointTemplate. Après la création des endpoints principaux le développeur peut entammer la creation des actions possibles sur le backend dans chaque endpoint. La création des endpoints et des actions est liée bien entendu au diagramme des actions backend. La base EndpointTemplate donne un bon exemple de comment doit se coder une action et comment gérer les données depuis celle-ci. Pendant la création des actions il est également obliger d’introduire la création des enveloppes qui sont les seules moyens dont disposent une méthode pour charger le contenu utile d’une requête (unique paramètre d’une action). Après la création des actions dans les endpoints le webservice a atteint sa maturité première. Bien entendu lorsque plusieurs actions dépendent ont doivent faire recourt a un procédé celui-ci doit être coder dans l’AbstractEndpoint comme une méthode et être utiliser depuis elle. A présent le webservice est prêt a être tester.

### 3.1.2. Frontend

Dans le cas du client la première étape est le développement de la librairie (API) d’acces au web service. Il existe deja une API generique d’acces aux webservices Kugri. En priorite une version javascript doit etre mise disponible. Par ailleurs des versions ObjectiveC, Java brute, C++, Python, Scala, Ruby doivent etre créer par ailleurs et peuvent faire sujet de stages. Après intégration de l’API la prochaine étape est le test du lien. Chaque action endpoint peut etre contacter concernant sont état. Après validation du lien entre le client et le webservice a travers l’API ont peut aller vers la mise en place des donnees propres aux client. Les structures pour stocker les donnees propres au client. Cela peut entrainer la mise en place de coockie, de base donnee, de cache, etc… Apres la mise en place de ceux-ci viens l’implementation du core (controllers). Ils doivent etre fait en restant dans l’esprit du workflow du client. Apres la creation des controllers. Il faut mettre en place les vues en ne tenant pas compte du design de prime abord. L’objectif etant d’avoir quelque chose de testable sans fancy parts. Apres verification des workflows, Entre en scene l’integration du design. L’implementeur du client doit communiquer tous le long ici avec le designer concernant les dimensions des composants des vues. Au fur et a mesure on se fera une idee et des connaissances globales a ce sujet. Apres l’integration le client est pret a etre tester. Le client doit pouvoir etre tester hors connexion et connecte.

## 3.2. Design

Dans cette section nous allons aborde le travail du designer. Il se concentre sur le workflow et la description du Projet en réalité. Mais aussi des types de clients en jeu. De nos jours les clients web doivent pouvoir tourner sur des ordinateurs, des tablettes, des téléphones, … Il faut donc tenir compte des dimensions des composants des vues lors du designer et s'apprêter a avoir plusieurs modèles des vues. Au niveau natif par contre des connaissances solides peuvent être construites au fur et a mesure que nous avons de l'expérience dans le domaine Android, iOS, WPhone ou Symbian.

### 3.2.1. Web

Dans le cas de la necessite d’un client web il y a deux comportements. Le client peut etre evolutif: Landing Page uniquement | Landing Page -> Web App complete | Web App complete uniquement. Chacun d’eux represente un cas de vie du client. Un autre facteur egalement est le besoin d’un nom de domaine ou pas de la part du demandeur du projet. Dans le cas de nom demande, la web app doit etre encapsuler dans notre les apps de notre site principale et donc avoir un design dependant du notre. Il est a noter aussi que dans ce cas le web service sera sur notre serveur de production. Il est possible egalement que le client web soit accessible en allant sur apps.kugri.com et avoir soit le lien servicename.kugri.com ou le lien kugri.com/servicename. Le design d’un client web est tres ligne a l’Analyse meme du projet avant meme de dependre de sa conception.

### 3.2.2. Natif

La force du client Natif c’est de n’avoir rien a voir avec un nom de domaine particulier. Cela parce qu’il tourne comme une app sur l’hote. Tout ce dont le client a besoin c’est de pouvoir se connecter au web service qu’il aient un nom de domaine ou pas. Son design ne depend donc pas du tout de l’Analyse mais uniquement de la conception du projet. Mais il est clair que ce sont les besoins du client qui permettent de savoir si il est necessaire de passer par du Natif ou pas. Aussi le client peut apres etre distribuer sur notre site principales sur apps.kugri.com qui contient une description de tous les projets qu’ont a fait et leur client mobile|web. Le design et la conception d’un client natif necessite cependant plus de rigeur que celui d’un client web parce que d’une plateforme a une autre les contraintes ne sont pas les meme et si les client doivent avoir la meme apparence partout il faut prendre garde.

# 4. Test

Dans la section de test il y a deux mesures tres critiques et varier. Quand l’un peut indiquer le passe d’un emulateur a un appareil physique, l’autre est le passage du web service de l’ordinateur locale du developeur vers notre serveur de test. Avant meme de pouvoir demarrer le test a proprement dit il va falloir preparer l’environement de test lui meme.

4.1. Backend

Dans le cas du backend, tester un web service Kugri se passe en trois etapes. Mais avant le web service doit etre tester localement sur la machine du developeur avec un client REST. Toutes les requetes et reponses doivent etre sauvegarder comme mesure de controller pour etre executer par des scripts pour valider que des modifications n’ont pas alterer certains endpoints. Maintenant en ligne, la premiere etape est la creation de la base de donnee ainsi que la migration des donnes. Ensuite dans le code sources du webservice s’assurer que les access a la base de donnees sont correctes ainsi que les urls d’acces. Apres le web service doit etre deployer et les logs du Serveur d’applications analyser quant a la trace laissee par le deploiement. Une trace sans erreurs est l’objectif rechercher. Apres le deploiement le web service doit etre retester avec le client REST comme dans le cas locales avec toutes les requetes et reponses sauvegarder egalement. Apres validation ont peut certifier que le web service est deploy et fonctionnel sur le serveur de test et pret pour un test reel.

## 4.2. Frontend

Pour le frontend le test se passe egalement sur la machine du developeur. Il doit se faire sans connexion pour voir le fonctionnement offline du client apres il est teste avec le lien avec le web service local actif. Les tests precedents se font localement. Si les deux developpeurs backend et frontend utilisent des machines differentes deux procedes sont possibles soit que l’un des composants (web service ou client) soit mis sur l’une ou l’autre des machines ou soit que le serveur d’application soit accessible sur le reseau locale de l’entreprise. Dans un premier temps la premiere option sera utilisee et la version deploiement du web service chez le developeur frontend est la solution la plus simple. Les developpeurs auront alors des environnements identiques qu’ils l’utilisent completement ou pas. Apres les tests en local, les tests grandeurs natures se font sur les appareils reels qui peuvent etre un telephone, un systeme embarque (les systemes mobiles ont generalement des SDK qui incluent des emulateurs de leur appareil reel pour les developeurs) ou meme un ordinateur ou un serveur mais avec le service en ligne sur le serveur de test.

# 5. Integration et Recette

Apres la validation environement de test deux destins se tracent devant l’infrastructure. Elle peut migrer vers le client originel du projet comme un livrable complet du web service, des versions par plateformes et type demandes du client et aussi de la documentation: un guide utilisateur et une presentation du produit. Pendant et apres la validation des tests les developeurs et le designeurs doivent collaborer dans l’etablissement de cette documentation

5.1. Documentation

Elle represente le point d’entree documentaire du client du produit. Pour prendre connaissance de la solution au projet proposer mais egalement comme un support lors des seances de formation a l’utilisation de l’application.

### 5.1.1. Presentation du produit

Elle doit etre un powerpoint tres bref et explicite du projet et de la solution Kugri proposee. L’objectif etant que a la sortie d’une presentation relle le directeur sache a quoi ressemble la solution et comment elle resoue le probleme qui a ete pose.

### 5.1.2. Guide d’utilisation

Plus complexe et destiner au utilisateur il doit etre simplifier au point de pouvoir etre utiliser comme support de cours par un enseignant. Parce qu’en realite il servira a former le personnel du client a l’utilisation du produit Kugri solution de leur projet.

## 5.2. Livraison de l’application

L’application globale apres la phase de test et validation est prete a etre livree au client. Cela signifit mise accessible. C’est la premiere etape de la livraison. Elle peut se faire de maniere externe a Kugri Co sur des structures propres au client ou sur notre structure a nous. La deuxieme et derniere etape est la livraison des sources ainsi que de la documentation sous forme d’un support CD ou USB fait a cet effet.

### 5.2.1. Livrable externe

Dans le cas d’une livraison externe. L’equipe de Kugri Co doit veiller au deploiement de l’infrastrure de l’application chez le client. Sur une periode de un mois, nous soutrairons en plus des frais du projet. Le client peut aussi demander une structure de livrable moyennant des frais moindre pour sa propre equipe informatique qui va s’en occuper mais qui peut prendre contacte avec nous sur un tarif horaire.

### 5.2.2. Livrable interne

Dans le cas d’un livrable interne, l’infrastructure migre de l’environement de test a l’environement de production completement. Apres cette etape, les sources ainsi que la documentations doit etre fournie au client dans un delais de un mois maximal.

# 6. Mise a niveau

En fonction du type de contrat les mises a niveau peuvent etre payantes, gratuites ou a la charge du client. Dans tous les cas une mise a jour est prevu au moins chaque fois par an. Les mises a jour tiennent compte de l’horaire generalement autour de 3h00 AM pour le client web et le backend mais peut se faire a tout moment dans la journee pour le client mobile. Pour le client desktop, la construction d’un systeme interne de mise a jour doit permettre de le faire sans impacter l’utilisation de l’application pendant plus de 5 mins.

© 2014 Kugri Technologies Co. Tous droits reservés