CONCOURS DE L'AGRÉGATION EXTERNE
ÉCONOMIE ET GESTION
SESSION 2017

ÉPREUVE DE CAS PRATIQUE

Spécialité : Système d'information

Cas Balthazar

Durée de la préparation : quatre heures
Durée totale de l'épreuve : une heure

Vous disposez d'une durée maximale de quarante minutes pour présenter oralement la solution de l'étude qui vous est proposée. Votre exposé sera suivi d'un entretien avec le jury d'une durée maximale de vingt minutes.

La candidate ou le candidat est invité.e à mettre en œuvre des solutions précises et pertinentes permettant au jury d'évaluer la maitrise des principaux concepts mobilisés dans ses solutions.

Balthazar : leader de la transformation numérique

Présentation de Balthazar

Balthazar est un des leaders européens de la transformation numérique et propose des offres variées aux entreprises : édition de solutions métier, intégration de systèmes, conseil, gestion de l'infrastructure matérielle et de l'externalisation des processus d'affaires.
Cette Entreprise du Secteur du Numérique (ESN) accompagne ses clients dans leur processus de transformation et d'utilisation des technologies numériques en combinant innovation, valeur ajoutée et performance aux solutions et services apportés.
Balthazar est le résultat de plusieurs années de fusions acquisitions d'ESN françaises et européennes.
Aujourd'hui Balthazar, dont le siège social se trouve à Paris, compte plus de 37 000 collaborateurs, présents dans plus de 20 pays en Europe et dans le monde.
Les clients de l'entreprise viennent de secteurs d'activités aussi divers que les banques et assurances, le secteur public, l'aéronautique, la défense et la sécurité, le transport, les télécoms et les médias, l'énergie et enfin, la distribution.

Les principaux métiers de Balthazar

Balthazar met à disposition de ses clients une offre complète qui comprend l'intégration de systèmes, l'infogérance avec plusieurs solutions ainsi que des services de conseils stratégiques en management des Systèmes d'Information (SI) et des Technologies de l'Information et de la Communication (TIC).
Cette dernière activité est prise en charge par la branche Business & Capital (B&C) de Balthazar. B&C accompagne les entreprises dans les changements potentiels et les conseille sur les solutions les plus adaptées à leur besoin dans une optique d'amélioration des performances.

Présentation de l'entreprise cliente

RFS est un opérateur de télécommunications français créé en février 1990. En 2016, RFS réalisait 15,2 Mds€ de chiffres d'affaires. Ses activités principales aussi bien pour le grand public que pour les entreprises, sont la téléphonie fixe, internet, la téléphonie mobile. RFS compte 16 500 collaborateurs.

Un projet d'infogérance : facturation pour RFS

RFS fait face à plusieurs problèmes particuliers. D'abord, celui d'un mécontentement récurrent de sa clientèle face aux dysfonctionnements de son application de tarification. Le reproche le plus fort est celui d'une tarification « aléatoire » de la consommation de data pour le grand public. Les consommateurs ne comprennent pas leur relevé de consommation, leur facture et ils la contestent. Ce mécontentement se diffuse aujourd'hui sur les réseaux sociaux et porte grandement atteinte à l'image de marque de l'entreprise. L'autre problème crucial est celui de la concurrence féroce dans ce secteur. Face à ce problème et à une perte de parts de marché que la direction veut enrayer le plus rapidement possible, une stratégie de « chasse au coût » a été engagée. RFS veut gagner la guerre concurrentielle en s'appuyant sur une structure de coût meilleure que celle de ses concurrents.

La Direction des Services d'Information (DSI) envisage de :

- diminuer le SI en nombre d'hommes dans le souci d'une maîtrise des coûts.
- réorienter une partie des salariés du SI vers des tâches à plus fortes valeurs ajoutées.

Dans ce contexte, RFS pense faire le choix de l'externalisation et plus précisément de l'infogérance. Le projet d'infogérance dirigé par Balthazar pour RFS vise à développer et maintenir son logiciel de facturation grand public.
RFS possède plusieurs secteurs d'activités (téléphonie, internet fixe et mobile, et télévision) et plusieurs types de clients :

- Le parc de clients « grand public » (GP) qui correspond à des particuliers pouvant souscrire à des offres fixes,
- Le parc de clients «entreprise» qui reprend les services GP adaptés aux entreprises,
- Des opérateurs mobiles (MVNO) qui utilisent le réseau mobile fourni par RFS,
- Des opérateurs fixes (OPE) qui utilisent le réseau fixe fourni par RFS. Ces opérateurs proposent ensuite des offres de télécommunications à des particuliers via ces réseaux.

Pour gérer l'ensemble de la facturation de RFS, le projet utilise l'application TARIF : celle-ci fournit des outils et traitements spécifiques développés par l'équipe du projet (Java-J2EE et Oracle). TARIF permet de gérer l'ensemble de la facturation des services de télécommunications (génération de la facture, paiement, relance, etc.).

La Tierce Maintenance Applicative pour RFS recoupe plusieurs domaines fonctionnels parfois spécifiques à un type de client:

- Le « provisionning » (Opérateurs et Grand Public) désigne la gestion des comptes. Cette gestion se fait grâce à la mise en application d'actes de gestion fournis par l'application en amont et concerne les données clients nécessaires à la facturation : informations personnelles, offres et options

souscrites, …

- La valorisation (Opérateurs et Grand Public) consiste à calculer le coût d'une consommation, aussi appelée ticket (communication téléphonique, consommation de DATA, …), pour le client associé.
- La facturation (Opérateurs et Grand Public) permet de calculer le coût à payer pour le client et à générer la facture en prenant en compte toutes les informations de paramétrages du compte du client (cycle de facturation).
- L'extraction (Opérateurs, Grand Public) récupère toutes les informations nécessaires pour produire la facture (pdf, papier) du compte opérateur.
- Le paramétrage des offres (Opérateurs et Grand Public) consiste à paramétrer TARIF pour décrire le fonctionnement des offres de RFS (cycle de facturation, montant, libellés et rubrique sur la facture, …).
- L'Interface Homme Machine (IHM) Opérateurs permet de gérer les comptes opérateurs : création de compte, affectation d'offres commerciales à un opérateur (ensemble de prestations autorisées pour un opérateur de RFS donné), gestion des tarifs de chaque opérateur.
- L'encours (Grand Public) permet de fournir les consommations valorisées mais non facturée. Il gère également le déclenchement d'alerte en cas de surconsommation et l'extraction de ces données (encours forfait et surconsommations).

Les autres domaines portent sur le paiement des factures, les rejets de paiements, les remboursements, la relance, le contentieux. La comptabilité comprend essentiellement la journalisation de la facturation et des paiements, et permet également de faire des prévisions de facturation.

Les attentes de RFS

Dans le cadre de cette infogérance, RFS souhaite mettre en place un test comparatif ou « benchmark » annuel qui permettrait à la DSI de comparer la prestation d'infogérance de Balthazar en termes de coûts, de qualité et de délais. Pour cette infogérance, RFS s'interroge sur la facturation. RFS souhaite une facturation forfaitaire pour les coûts récurrents, une facturation simple et exhaustive. En outre, RFS souhaite avoir des indicateurs concernant les dysfonctionnements de l'application de facturation.

RFS souhaite donc évaluer Balthazar sur sa capacité à prévoir et analyser les principaux incidents, mais aussi sur sa capacité de conseils quant aux propositions de services, d'améliorations, d'évolutions applicatives. Cette évaluation mise en place par RFS, évaluation de la qualité d'accompagnement, de service s'appuierait sur un ensemble de tableaux de bord, définis par RFS, et réalisés mensuellement par Balthazar.

Dossier 1 : Infogérance

La direction de RFS s'interroge quant à la pertinence de cette décision d'infogérance.

TRAVAIL À FAIRE	
1.1	Précisez le type d'infogérance demandé par RFS ? En quoi cette infogérance répond aux objectifs de RFS ?
1.2	Que pensez-vous de ce choix en termes d'alignement stratégique ?
1.3	Quelles peuvent-être les difficultés rencontrées dans la mise en place d'une solution d'infogérance pour RFS ?
1.4	Que pensez-vous d'un contrat de service de type « SLA » (Service Level Agreement) pour RFS ?

Dossier 2 : Qualité du code de l'application TARIF

TARIF est une application N tiers (composants clients, composants serveurs), interfacée avec d'autres applications du système d'information, et de forte volumétrie en termes de lignes de code, de fichiers, de classes, de formulaires, de bases de données relationnelles…

La complexité du code se situe majoritairement dans la partie Java - J2EE. La complexité Oracle est moindre. Une synthèse partielle d'un audit qualité réalisé par SEI (Software Engineering Institute) est présentée en annexe. Elle met en évidence plusieurs points améliorables.

TRAVAIL À FAIRE	
2.1	Proposez des éléments concrets permettant de mesurer les deux « facteurs de santé » évoqués : capacité d'évolution et sécurité.
2.2	Explicitez la première des recommandations : systématisez la gestion des exceptions et journalisez les exceptions.
2.3	Proposez concrètement de « bonnes » pratiques de programmation, visant notamment à améliorer la lisibilité du code.

Pour repérer les dysfonctionnements de TARIF, l'équipe de maintenance applicative met en place des tests unitaires.

Pour intégrer le calcul des consommations téléphoniques des comptes clients de RFS, l'application TARIF dispose de classes implémentées en Java.

Les annexes présentent l'implémentation partielle en Java de ces classes, ainsi que les classes techniques (HashMap et tests unitaires).

TRAVAIL À FAIRE	
2.4	Rétro-concevez sous forme de diagramme de classes, les classes impliquées dans la tarification des appels téléphoniques.
2.5	Le calcul de la durée cumulée des appels hors forfait semble erroné. Codez une méthode de test unitaire de la méthode dureeAppelsHorsForfait().

Dossier 3 : OTARIE, un assistant au service de l'équipe maintenance

Dans le cadre du support fonctionnel que Bathazar fournit à RFS, l'équipe support est chargée :

- de la gestion des incidents de l'application TARIF,
- du traitement des demandes de service,
- de l'élaboration et de la diffusion des tableaux de bord de l'activité,
- du maintien en condition opérationnelle des environnements.

Le support concerne ainsi plusieurs domaines métiers, auxquels sont affectés les membres de l'équipe Balthazar : support fonctionnel (SF Niveau 1, Niveau 2), exploitation, demande de service, mise en exploitation…

OTARIE est un ensemble d'outils, développés et maintenus en interne, dont l'objectif est de simplifier les tâches les plus fastidieuses que doivent réaliser les membres de l'équipe support dans leur travail et de centraliser l'information (requêtes SQL, modes opératoires, etc.) dans une base de connaissances commune.

OTARIE permet ainsi d'améliorer le travail des membres en mettant à disponibilité tous les documents, les contacts et les outils de recherche nécessaires pour chaque domaine de la maintenance.
Par exemple, les modes opératoires représentent les démarches à suivre afin de traiter une demande de service et les actions à effectuer pour une tâche récurrente donnée.
Ces modes opératoires sont décrits dans des fichiers format traitement de texte, accessibles par les membres de l'équipe via un disque réseau partagé.

Parmi les tâches à réaliser, les tâches récurrentes sont des traitements à effectuer afin de corriger les données de production erronées dues à des incidents du système.
Le temps que ces incidents soient corrigés, les tâches doivent être effectuées à intervalle régulier. Ces tâches correctives sont concrètement des commandes système et/ou des traitements batchs et/ou des requêtes sur la base de données. Historiquement, les compétences des membres de l'équipe se sont établies dans l'environnement Unix : la migration de serveurs en environnement Windows nécessite de s'adapter.

Une partie du schéma relationnel de la base de données relative aux tâches récurrentes a été élaborée. Il est présenté en annexe. Il fournit un support aux connaissances nécessaires à la bonne exécution des tâches récurrentes.

TRAVAIL À FAIRE	
3.1	Rédigez la requête SQL permettant d'obtenir le nom et le libellé des catégories de commande (Unix) qui n'ont pas d'équivalent (Windows).
3.2	Rédigez la requête SQL permettant d'obtenir les tâches récurrentes et leurs rédacteurs : indiquez le libellé de la tâche, les nom et prénom du rédacteur, le domaine métier du rédacteur.

La codification des catégories de commande a changé. Ces catégories étaient simplement un entier auto-incrémenté, leur identifiant prend maintenant une signification pour un accès plus rapide. Un tableau de correspondance entre les anciens et les nouveaux codes utilisés pour les identifier vous a été communiqué. Les libellés des catégories n'ont pas changé. La base de données doit utiliser ces nouveaux codes dans l'attribut idCateg.

Extrait du tableau de correspondance entre anciens et nouveaux codes :

Ancien code	Nouveau code
01	statReseau
02	extractLog
03	quotaDisque
…	…

TRAVAIL À FAIRE	
3.3	Expliquez précisément le problème que pose le changement des codes catégorie des commandes dans la base de données. Proposez une solution pour effectuer ce changement.

La périodicité des tâches récurrentes étant variable et leur nombre relativement élevé, il devient nécessaire d'aider les membres du support en leur indiquant quand et quelles tâches récurrentes ils doivent effectuer.
Les tâches ont des périodicités variées : par exemple, tous les jours, chaque lundi, chaque 3ème jeudi du mois, etc.
L'outil OTARIE vise ainsi à fournir une planification des tâches à exécuter et à tracer leur exécution.

TRAVAIL À FAIRE	
3.4	Étendez le modèle des données existant, pour permettre d'établir un planning des tâches à exécuter par les intervenants. Le système devra permettre de pointer les tâches non réalisées à l'échéance prévue.

Dossier 4 : Infrastructure réseau

Le réseau étendu de Balthazar est un élément crucial du bon fonctionnement de l'entreprise. Il permet l'accès au système d'information de milliers d'utilisateurs répartis dans le monde.
Les préoccupations des équipes réseau portent sur la disponibilité et la sécurité du réseau, cela dans un contexte d'évolution technique permanente, de besoins de connectivité hétérogènes, de besoins de performance et de couverture géographique grandissante.

Le schéma simplifié du réseau de l'agence d' Issy les Moulineaux est présenté dans le dossier documentaire.

TRAVAIL À FAIRE	
4.1	Expliquez la nature et le rôle des technologies en présence.
4.2	Décrivez et expliquez les bonnes pratiques de sécurisation des équipements d'interconnexion.
4.3	Les segments représentent les services de l'agence : administratif, ressources humaines, comptabilité, etc. Chaque service est dans un réseau virtuel (VLAN) distinct. Proposez un plan d'adressage IP et de configuration des équipements d'interconnexion, permettant aux services d'accéder aux ressources locales propres au service (imprimantes, fichiers, etc.) ainsi qu'aux ressources mutualisées tel que l'accès Internet.

Annexe 1 – les prin cipaux indicateurs de Balthaz ar

Figure 1 - Répartition des collaborateurs de Balthazar

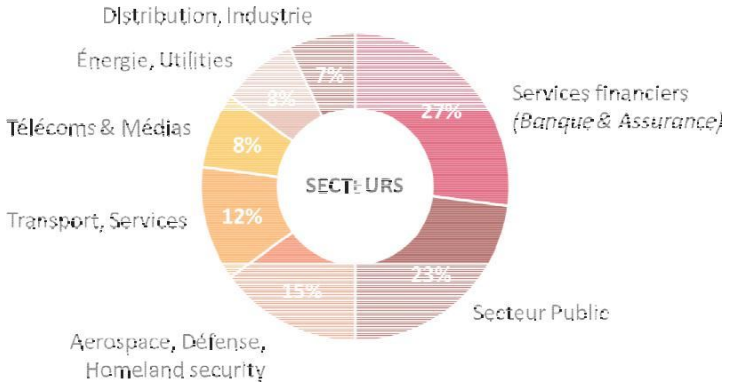


Figure 2 - Répartition de l'activité par secteur

Figure 3 - Principaux client s de Balthazar

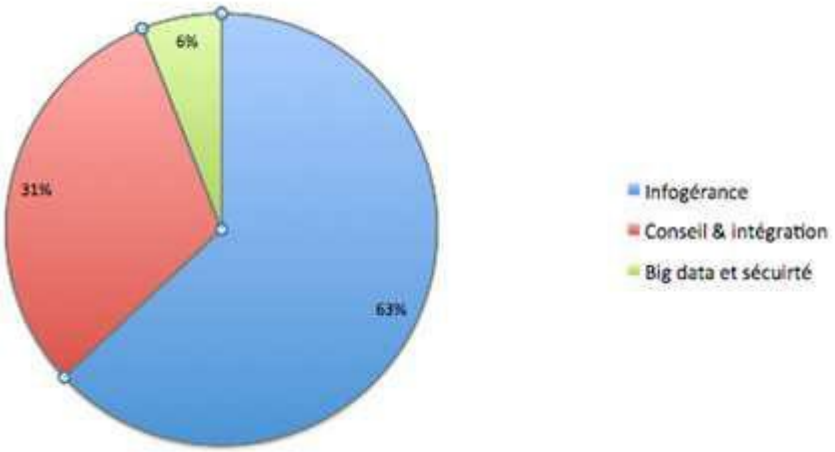
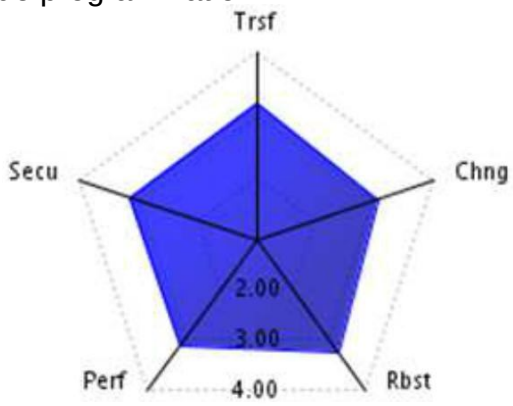


Figure 4 - Répartition chiffr e d'affaires Balthazar T1 2016

Annexe 2 – Audit de code de l'application TARIF

L'audit statique de l'application et de son code s'appuie sur la norme ISO, ISO 9126, qui relève du Génie Logiciel.
Les « facteurs de santé » d'une application étudiés par la norme sont : transférabilité, capacité d'évolution, robustesse, performance, sécurité, maintenabilité, ainsi que l' architecture et les pratiques de programmation.

Transferability	3.16
Changeability	3.05
Robustness	3.24
Performance	3.1
Security	3.13



SEI Maintainability	3.45	3.45
Technical Quality Index	3.15	3.15

Architectural Design	2.69
Documentation	2.74
Programming Practices	3.28

niveau bon : 3.75 < note
Niveau satisfaisant : 3.50 < note < 3.75
Niveau moyen : 3.25 < note < 3.50
Niveau faible : 3 < note < 3.25
Niveau alarmant : note < 3

Bilan de l'audit de l'application – Recommandations
La moyenne des facteurs de santé : 3,19.
Si aucune note, en moyenne, n'est critique, trois points restent à améliorer.

1- Remplacer la gestion « classique » des erreurs en Java par un système de gestion des exceptions.
Envisager la journalisation (log) des exceptions levées
2- Factoriser le code (attention au copier / coller)
La duplication de code augmente les coûts de maintenance.
3- Généraliser la documentation (JavaDoc)
Les classes et méthodes doivent être systématiquement documentées (/** */)
En particulier, les tags @author, @param, @exception, @throws et @return doivent être spécifiés.

Annexe 3 – classes consommation téléphonique

```
public class Appel { // extraits
    public Appel(String id, String numeroAppelé, double coutMinute
                 boolean forfait)   // constructeur
    public double getCoutMinute()   // accesseur
    public boolean estForfaitaire()   // vrai si appel dans le forfait, faux sinon
}

public class LigneTelephonique {
        int             id;
        Date            dateCreation;
        HashMap<Appel, Integer> lignes;

        // constructeur
        public LigneTelephonique(int id, Date d) {
                id              = id;
                dateCreation = d;
                lignes    = new HashMap<Appel, Integer>();
        }

        /**
         * Fournit la liste des appels sous forme de dictionnaire
         * clé : l'instance de Appel => valeur : la durée de l'appel
         * @return      liste des appels
        */
        public HashMap<Appel, Integer> getLignes() {
                return lignes;
        }

        /**
         * Ajoute un appel à la liste des appels
         * @param       unAppel : instance de classe Appel
         * @param       uneDuree : durée de l'appel en minutes
        */

        public void ajouterLigne(Appel unAppel, int uneDuree){

        …
        }

        public void supprimerLigne(Appel unAppel){

        …
        }

        /**
         * calcul de la durée cumulée des appels de la liste, qui ne sont pas dans le forfait
        * @return      durée entière (minutes)
        */
        public int dureeAppelsHorsForfait(){
        …
        }

}
```

Annexe 4 – classes techniques

classe HashMap du framework Java

*/**
La classe HashMap permet de mémoriser un dictionnaire d'éléments (clé, valeur). Toute valeur (de type ValueType) peut être extraite à partir de sa clé (de type KeyType) à une clé présente dans le dictionnaire correspond une et une seule valeur
/

***Class* HashMap <KeyType, ValueType>**

public void put (KeyType key, ValueType value)

// ajoute un élément (key, value). Remplace la valeur existante si
// l'élément existe déjà pour la clé spécifiée.

public ValueType get (KeyType key)

// retourne la valeur correspondant à la clé spécifiée, ou null si la
// Clé spécifiée est inexistante.

public ValueType remove (KeyType key)

// retire du dictionnaire l'élément correspondant à la clé spécifiée // et retourne la valeur qui y était associée. Ne fait rien et
// retourne null si la clé était inexistante.

public Boolean containsKey (KeyType key)

// retourne true si l'élément dont la clé est passée en paramètre est
// présent dans le dictionnaire, false sinon

public Set<KeyType> keySet ()

// retourne un ensemble des clés présentes dans le dictionnaire d'éléments

classe Assert du framework Java

*/**
La classe Assert réunit un ensemble de méthodes statiques d'assertion utiles pour l'écriture des tests. Une méthode d'assertion a pour rôle de vérifier si une affirmation est ou non respectée. Dans le cas où elle ne l'est pas, la méthode déclenche une exception. Si non, elle ne fait rien. */*

***Class* Assert**

public static void AssertEquals(String message, int expected, int actual)

// vérifie si les valeurs entières expected et actual sont égales.
// Si elles ne le sont pas, déclenche une exception de type
// AssertException comportant le message d'erreur spécifié.

public static void AssertEquals(String message, Object expected, Object actual)

// vérifie si les deux objects expected et actual sont égaux.
// S'ils ne le sont pas, déclenche une exception de type
// AssertException comportant le message d'erreur spécifié.

public static void AssertNull(String message, Object obj)

// vérifie si l'objet obj est une référence nulle. S'il ne l'est pas,
// déclenche une exception de type AssertException comportant le message
// d'erreur spécifié.

public static void AssertNotNull(String message, Object obj)

// vérifie si l'objet obj n'est pas une référence nulle. S'il l'est,
// déclenche une exception de type AssertException comportant le message
// d'erreur spécifié.

Annexe 5 – classe de test

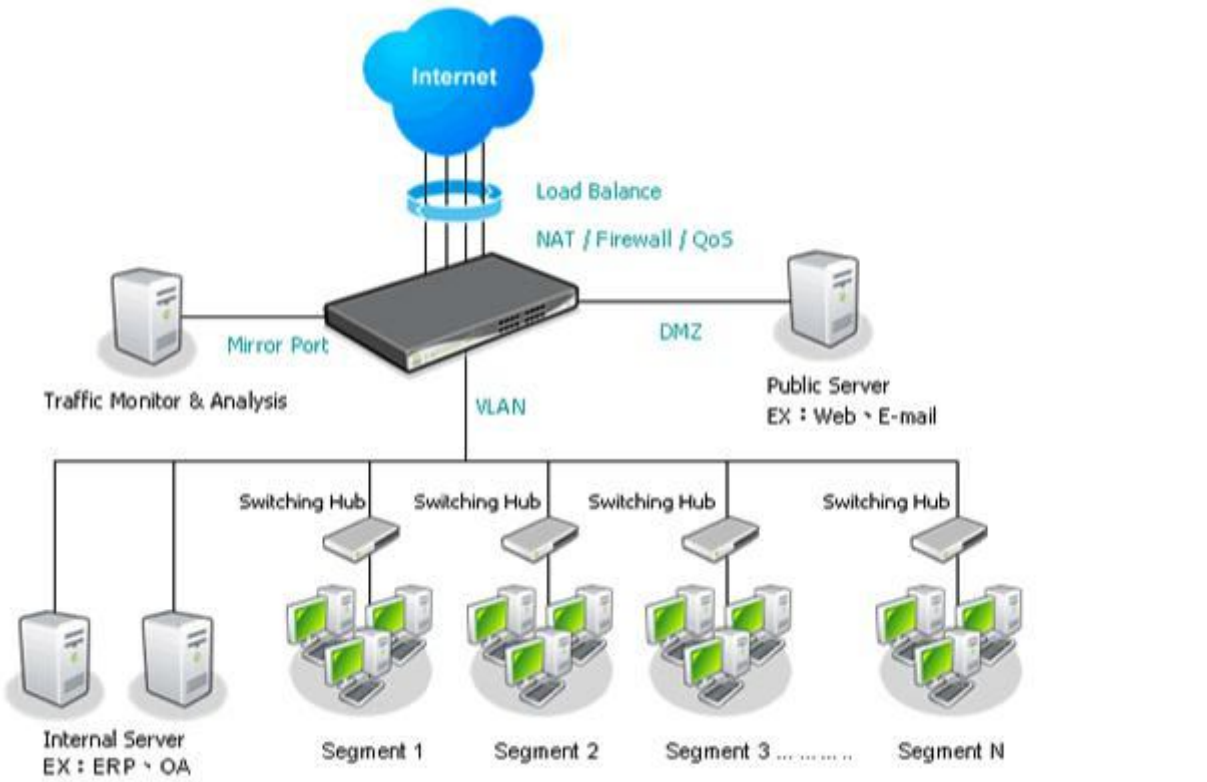
```
/**
 * Classe de test de la classe LigneTelephonique
 * Hérite de la classe TestCase issue du framework de tests unitaires Junit
 * Enchaîne l'exécution de toutes les méthodes commençant par test
*/
public class TestLigneTelephonique extends TestCase {

        public void testAjouterLigneNouvelAppel () {
            // création d'une ligne et d'un appel
            LigneTelephonique lt = new LigneTelephonique(1, new Date());
            Appel unAppel= new Appel("01", "0123456790",0.30, true);// l'appel est dans le  forfait


            lt.ajouterLigne(unAppel, 20); // l'appel a duré 20 minutes
            HashMap<Appel, Integer> lesLignes = lt.getLignes();
            Assert.assertEquals(lesLignes.size(), 1);
            Assert.assertNotNull(lesLignes.get(unAppel));
          int duree = lesLignes.get(unAppel);
            Assert.assertEquals(duree, 20);
}

// suite des méthodes de test
…
}
```

Annexe 6 – réseau d'agence / Issy les Moulineaux



Annexe 7 – OTARIE : schéma relationnel des tâches récurrentes

TacheRecurrente(id , libelle ,commentaire, dateDebut, dateFin, dateModification, idModeOperatoire)

id : clé primaire

idModeOperatoire : clé étrangère en référence à id de ModeOperatoire

Intervenant(id, nom, prenom, idDomaineMetier)

id : clé primaire

idDomaineMetier : clé étrangère en référence à id de DomaineMetier

DomaineMetier(id, libelle)

id : clé primaire

// libellé est à valeur dans { SFN1, SFN2, MEP, Exploitation, DDS…}

ModeOperatoire(id, libelle, fichierMop, idRedacteur,

idTacheRecurrente) id : clé primaire

idRedacteur : clé étrangère en référence à id de Intervenant

idtacheRecurrente : clé étrangère en référence à id de TacheRecurrente

CategorieCommande(categCommande, libelle)

categCommande : clé primaire

// libellé est à valeur dans {processus, réseau, fichiers, journaux}

// il s'agit de classer par catégorie les commandes des systèmes d'exploitation, dont l'emploi peut être indispensable pour réaliser certaines tâches récurrentes : ce sont essentiellement des commandes et leurs options de diagnostique de l'état du système et d'analyse des journaux.

Commande(id, libelle, usage, idCateg)

id : clé primaire

idCateg: clé étrangère en référence à categCommande de CategorieCommande

// référence les commandes Unix nécessaires aux activités de maintenance.

CommandeEquivalent (idCommande, idCommandeEquivalent)

idCommande, idCommandeEquivalent : clé primaire idCommande : clé étrangère en référence à id de Commande idCommandeEquivalent : clé étrangère en référence à id de Commande

// migration de serveurs : équivalents Windows des commandes Unix ci-dessus