Université de Caen
Licence Informatique **Rapport de stage**

Maître de stage : MARÉCHAL David
Tuteur universitaire : Terrier Véronique, BACQUEY Nicolas

Rapport de stage : Kuchikomi

RACINE Pierre-Alexandre

Caen, le 02 Juillet 2013

Remerciements

Je tiens à remercier Mr Maréchal pour m'avoir accepté en stage et m'avoir offert la possibilité de mettre mes connaissances en pratique sur un projet important.

Je remercie également tous les membres de *Nearforge* pour m'avoir permis de tant apprendre et m'avoir transmis une part de leur savoir-faire et de leurs connaissances.

Table des matières

Introduction

0.1 Présentation de l'entreprise

Nearforge est une jeune start-up créée le 31 Janvier 2013. Ses 6 fondateurs sont issus de NXP.

En créant cette entreprise, leur idée était que le NFC était une technologie porteuse. Elle est intégrée dans un nombre croissant de tablettes et de téléphones mais les seuls usages qui en sont fait sont les paiements et les transports. Il pourrait pourtant y avoir beaucoup plus de cas d'usages dans un nombre de domaines très important.

0.2 Présentation du projet

Le projet qui m'a été confié est de créer un émulateur de *Twitter* dont les *tweets* seront écrits par des commerçants et lus par leurs clients. Les clients s'inscriront et s'abonneront par un simple contact de leur téléphone à un champ NFC ou par le scan d'un QRcode.

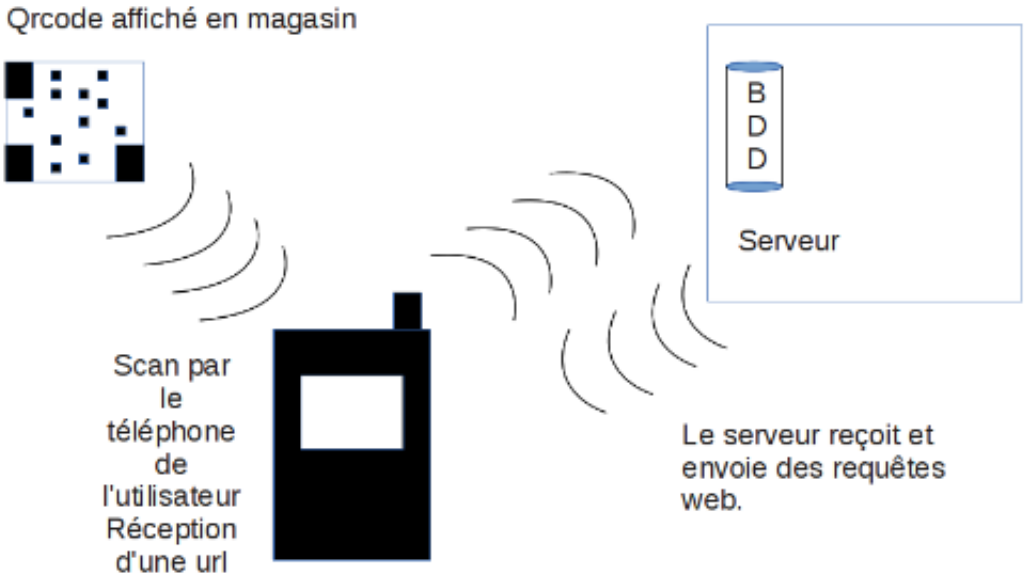
De même, tout commerçant bénéficiera d'une carte avec puce NFC qui, par simple contact avec son téléphone, lui permettra de se connecter directement à son compte et à son interface lui permettant d'écrire ses « kuchikomi » (nom des tweets).

Un administrateur, quant à lui, aura accès à une interface lui permettant de modifier aisément une partie de la base de données.

Toute l'application étant utilisable via un navigateur web (de préférence sur mobile) Le tout sera géré par un serveur web grâce à des scripts PHP. Une grande attention est requise pour la consultation des pages accessibles aux commerçants et aux utilisateurs car la mise en forme du site doit être adaptative (*responsive design*).

0.3 Ma mission dans ce projet

Dans ce projet, mon rôle fut de créer un serveur et d'y placer des scripts PHP capables d'exécuter des requêtes envoyées par des téléphones portables. Je dus également créer et normaliser une base de données répondant aux besoins de l'application. Une partie de l'application est installée sur les téléphones. Un protocole d'échange de données fut normalisé en concertation avec le développeur chargé de cet aspect.



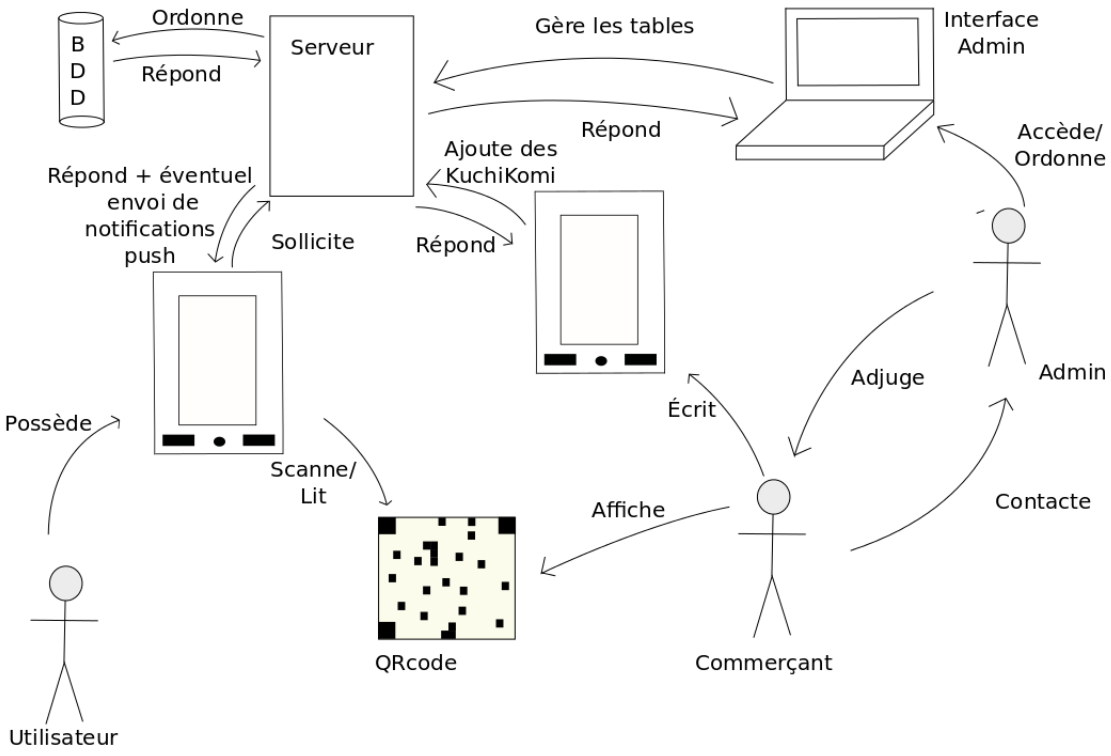
Chapitre 1

Vue d'ensemble

1.1 Attendus et objectifs

À mon arrivée, l'application était à l'état d'idée. Tout fut donc à faire. Le cahier des charges requérant les points suivants :

- Un utilisateur doit pouvoir s'abonner/s'inscrire/se connecter facilement depuis un téléphone soit grâce à un tag NFC soit grâce à un QRcode.
- Un utilisateur peut consulter ses abonnements et leurs fils de « kuchikomi » .
- Un utilisateur peut également voir des infos détaillées d'un commerce pour s'y rendre ou les contacter.
- Un commerçant doit pouvoir se connecter depuis un téléphone et y écrire des textes qui pourront être lus par les clients qui se seraient abonnés chez lui.
- Un commerçant peut également accéder à des statistiques plus ou moins précises sur ses clients.
- Un administrateur doit pouvoir inscrire des commerçants et modifier leurs données publiques. Il aura également accès à des statistiques générales sur les utilisateurs. Ma part du travail consista et consiste encore dans la partie web de l'application. C'est sur cette application web que fut/sera implantée une sur-couche logicielle permettant de simplifier l'utilisation (en particulier la connexion automatique)
- Le site doit être programmé en orienté objet et le code HTML généré doit être valide W3C.



1.2 Outils

Les logiciels utilisés sont :
- *Debian* (à venir)
- MySQL
- PHP
- HTML5/CSS3
- *Inkscape*
- *icescrum*
- Git et son interface *smartgithd*
- *Bootstrap Twitter*
- XML

Le système d'exploitation *Debian* a été choisi pour sa stabilité et sa sécurité. Le serveur qu'il accueillera devant héberger des données de travail qui devront être disponibles à tout moment, ses qualités sont indispensables dans le cas qui nous concerne.

PHP et MySQL ont été choisis pour leur licence libre et leurs documentations conséquentes. De plus, la richesse de leurs API rend le développement plus facile avec ces outils.

Le couple HTML5/CSS3 est choisi car l'application devra envoyer des pages HTML devant être lues sur un grand nombre de terminaux de types très variés. Leur grande compatibilité sur la plupart des appareils les rend tout désignés pour cette utilisation.

Inkscape est un logiciel de dessin vectoriel. Les schémas ainsi créés ne perdent pas de détails lors d'un agrandissement ou d'une réduction. De plus, le très grand nombre de formats d'export rendent ce logiciel très intéressant.

Le logiciel gérant les étapes de développement agile de type SCRUM est *icescrum*. Son interface permet de créer, gérer et planifier aisément chaque étape du développement.

Afin de sauvegarder régulièrement le travail, le superviser et de travailler à plusieurs, le logiciel utilisé est Git avec son interface graphique *smartgithd*.

L'application ayant vocation à être présente sur de très nombreux terminaux, les vues générées doivent s'y adapter au mieux. Pour faciliter le travail, le framework libre *Bootstrap Twitter* a été intégré. Celui-ci propose des classes CSS déjà écrites.

Enfin, le langage XML est un langage de structuration de données. Il permet de structurer des données sous la forme d'une arborescence que sera envoyée aux applications présentes sur les téléphones.

1.3 Présentation de la technologie NFC

La technologie NFC (« Near Field Communication » ou « communication en champ proche ») est une technologie permettant des communications sans-fil à haute-fréquence et à courte portée (pas plus d'une dizaine de centimètres) Le NFC en champ proche hérite une bonne part de ses spécifications de la RFID (« Radio Frequency Identification » ou « radio-identification »), il peut donc communiquer avec d'autres appareils NFC ou d'autres appareils sans-contact (les validateurs du tramway de Caen l'utilisent)

Le NFC dispose d'avantages indéniables. Facile d'utilisation (un seul geste suffit) et sécurisation des données échangées. Comme la distance de transmission est faible, il est difficile de tenter de s'interposer pour capter le signal.

Un téléphone équipé de cette technologie se comportera comme un lecteur et émettra un champ électromagnétique servant à alimenter le tag NFC pour lire ou modifier des données dans la puce.

Il y a plusieurs types d'utilisation possible. Quelques exemples :

- En lecture seule, ce qui permettra de récupérer des informations en ouvrant le navigateur internet (par exemple, les dates d'un spectacle en scannant une afficha)
- Écriture de données puis lecture de ces données avant une nouvelle écriture (par exemple, une chaîne de montage d'objets où seront stockées les informations de ces objets au fur et à mesure de leur création)
- Installer et configurer une application. (par exemple, un gestionnaire de contacts dont les contacts seraient justement inscrits à chaque nouveau tag)

Un nombre de plus en plus important d'appareils mobiles sont équipés de matériel pouvant communiquer en NFC. On peut citer, parmi beaucoup d'autres, les smartphones *Samsung Galaxy S II* ou encore le *Nexus S*. Au total, on estime à 300 millions le nombre de terminaux équipés de cette technologie.

Aujourd'hui, l'accroissement du nombre d'objets NFC permet la communication et les interactions entre utilisateurs et objets mais aussi entre objets créant ainsi un internet des objets.

1.4 Présentation des QRcodes

Les QR-codes (« Quick-Response-code ») sont des codes pouvant être lus et interprétés par un téléphone, une webcam ou tout objet équipé d'un objectif avec un logiciel de reconnaissance.

Apparu en 1999 au Japon, il s'agit d'un format de données ouvert.

Ce code est un fond blanc sur le quel se posent des carrés noirs. Ceux du centre codent le message tandis que les grands carrés situés dans les coins renseignent le sens de lecture.

Techniquement parlant, le QRcode n'a pas de limites d'effets tant qu'une application en comprendra le sens (ouvrir une url dans le navigateur, envoyer un mail ou un SMS, effectuer un paiement, etc...)

1.5 Présentation de Bootstrap Twitter

Bootstrap Twitter est un framework, un kit de composants et d'outils sous licence libre. De fait, il intègre des fichiers CSS et optionnellement javascript (dans le cas d'une utilisation avec jQuery)

Ces fichiers permettent d'utiliser des classes prédéfinies qui seront présentes par défaut et/ou que l'on pourra appeler à volonté. Il intègre également des images d'icônes pour une meilleure convivialité.

La quasi-totalité des balises HTML sont modifiées par les fichiers CSS de ce framework. Le style général de la page web devient alors sobre et élégant.

Ainsi, une page contenant de simples liens peut, grâce à l'ajout de classes dans les balises, devenir ceci :

1.6 Présentation de Git

Afin de sauvegarder et de travailler à plusieurs sur un même projet, le logiciel Git est utilisé. Il s'agit d'un gestionnaire de versions décentralisé. À la base, ce logiciel a été développé par Linus Torvalds.

Concrètement, après avoir modifié son code, on le « commit » (pour créer une nouvelle version) et on le « push » (pour l'envoyer au serveur Git)

Git vérifie alors si il y a incompatibilité avec une ou plusieurs modifications faites par un autre développeur qui aurait fait un *push* entre-temps (par exemple on modifie une ligne alors qu'un collègue l'aurait supprimée)

Si il n'y a pas d'incompatibilités, le fichier est « versionné » . On peut ainsi travailler à plusieurs sur un même projet en gérant les possibles incompatibilités.

Chapitre 2

Étapes de réalisation

2.1 Analyse

Après étude du cahier des charges, je fis une maquette en HTML de toutes les pages de l'application. Je pus ainsi mieux appréhender les besoins de la base de données pour la modéliser peu après.

Ainsi le premier jet de la base de données fut modélisé pour être peu à peu normalisé jusqu'à la troisième forme normale.

Pourquoi normaliser une base de données ?

Normaliser ou pas une base de données se fait selon un critère simple : la proportion de l'écriture par rapport à la lecture.

Une base normalisée est éclatée en unités atomiques dépendantes les unes des autres. Par conséquent, une base normalisée évite la redondance d'informations.

Et plus elle sera normalisée, plus elle sera efficace en écriture et inefficace en lecture (jointures plus fréquentes)

Il a été décidé de normaliser la base de données afin de la rendre plus logique et éviter la redondance d'informations. Toutefois, l'application nécessitera de nombreuses opérations en lecture et relativement peu en écriture. La troisième forme normale devrait parfaitement correspondre à ce qui est et sera attendu.

En annexe : processus de normalisation d'une base de données.

Normaliser une base de données présente avantages et inconvénients.

Parmi les avantages, la base est plus logique, elle offre moins de redondances d'informations donc consomme moins d'espace disque et présentera moins d'incohérences lors de mises à jour qui seront alors simplifiées. Comme une donnée est écrite moins souvent (en théorie une fois), les requêtes d'écriture sont plus rapides.

Les inconvénients d'une normalisation ne sont pas anodins. Comme les informations sont présentes moins souvent, les jointures sont beaucoup plus fréquentes. Ce qui signifie des temps de lecture beaucoup plus longs. De plus, moins de redondances d'informations signifie également moins de sauvegardes. Enfin, la structure de la base est beaucoup plus statique.

Le diagramme de classes fut alors pensé en fonction de la base normalisée pour s'y greffer de manière la plus naturelle.

2.2 Préparation

Les premiers temps servirent à la préparation : étude du cahier des charges, maquette HTML, modélisation bdd, modélisation des classes, préparation de l'environnement de travail (formatage PC + installation XAMPP)

2.3 Codage

Puis vint le codage des premiers scripts.

Le modèle MVC fut choisi pour l'organisation du code. De cette manière, la maintenance de l'application est simplifiée.

D'abord les classes et l'interface de l'utilisateur furent implémentées jusqu'à permettre toutes les actions demandées par le cahier des charges. Puis le même protocole fut appliqué sur l'interface des commerçants. Finalement, un panneau d'administration fut lui aussi créé.

Ces deux dernières parties demandèrent des opérations sql fréquentes et variées (comptage, statistiques simples sur un laps de temps donné, sous-requêtes, etc...)

Une fois tout ceci fait, des discussions sur le protocole d'échange entre l'application côté serveur et l'application côté terminaux.

2.4 Communication serveur/téléphone

2.4.1 Authentification automatique

Le problème d'origine est qu'un utilisateur doit être automatiquement inscrit, connectée et abonné lors de son premier scan de magasin et **sans toucher à aucun bouton ni valider**.

Pour cela, il fallut trouver un biais pour envoyer l'identifiant au serveur tout en ouvrant une page web vers l'application serveur.

Plusieurs méthodes existent :

- les cookies
- les sessions
- la super-globale GET
- la super-globale POST

L'utilisation de cookies rend la sauvegarde des abonnements à la merci d'une suppression. De plus, les vols de sessions cookies rendent la sécurisation des données peu fiables.

Les sessions sont la solution idéale. Malheureusement, elles ne peuvent être implémentées car l'envoi d'un POST par l'application cliente sera considérée comme différente de l'ouverture d'une page web par cette même application. La même application envoyant deux requêtes en moins d'une seconde sera donc perçue par le serveur comme deux utilisateurs distincts rendant ainsi inutile l'utilisation des sessions.

La variable super-globale GET transmet les données dans l'URL. Il n'est donc pas possible de l'utiliser car il suffirait de connaître l'URL pour que quiconque puisse s'identifier comme tel ou tel utilisateur.

Il ne reste donc que la super-globale POST. Malheureusement, comme dit plus haut, l'envoi d'un POST suivi d'une ouverture de page web par l'application ne sera pas perçue par le serveur comme une seule entité.

Le temps manquant, le moins mauvais biais que j'ai pu trouver est d'utiliser l'adresse IP de l'utilisateur tentant de se connecter.

Tout d'abord l'application envoie le POST contenant un identifiant. Le serveur récupère alors la valeur de cet identifiant ainsi que l'adresse IP de l'origine de ce POST.

Si cet identifiant est déjà connu, on met à jour son adresse IP avec celle trouvée.

Si cet identifiant est nouveau, on insère dans la base ce nouvel utilisateur avec l'adresse IP récupérée.

Puis l'application cliente ouvre une *webview* vers l'URL du serveur avec en paramètres les identifiants du commerce qui étaient contenus dans la puce NFC.

Le serveur récupère alors l'adresse IP de l'application appelant la page. Si elle se trouve dans la base, on en déduit alors l'identifiant de l'utilisateur que l'on connectera et abonnera alors avant de le rediriger vers sa liste d'abonnements.

Si l'adresse IP n'est pas dans la base, c'est que le POST émis un instant auparavant s'est mal produit. Rien ne se passe alors.

2.4.2 Notifications automatiques

On souhaite que dès qu'un commerçant aura publié un *kuchikomi*, une notification soit envoyée à tous les abonnés.

La solution à utiliser est le système des notifications PUSH. Malheureusement, le temps pour développer cela dans l'application (téléphone et serveur) manque. Aussi, là encore, un biais fut utilisé pour la version de démonstration.

Ce biais consiste en un appel de l'application (téléphone) au serveur se répétant toutes les 3 secondes. Le fichier appelé par l'application est un script générant du XML. Les données concernées sont la liste des derniers *kuchikomi* écrits par les commerçants où l'utilisateur est abonné.

L'application (téléphone) compare les *kuchikomi* qu'elle a reçu correspond avec ceux qu'elle a en mémoire. En cas de différence(s), l'application émet une notification à l'utilisateur au sujet des nouveaux kuchikomi.

Bien sûr, si il n'y a pas de différence, l'application ne fait rien.

Chapitre 3

Difficultés rencontrées

Mis à part les bugs habituels à corriger, aucune difficulté réelle ne s'est présentée. Les différents outils utilisés possèdent tous des documentations de qualité, ce qui facilite grandement le travail.

Le seule véritable difficulté aura été et est toujours d'implémenter l'authentification automatique des utilisateurs et commerçants. Comme expliqué ci-dessus, le problème fut temporairement contourné grâce aux adresses IP pour la version démonstrative.

Chapitre 4

Ce que j'ai appris

Grâce à ce stage, j'ai pu apprendre des méthodes, des outils, des langages que je ne connaissais pas ou pas suffisamment et m'améliorer dans leur utilisation.

Bootstrap Twitter, le framework CSS, en est le meilleur exemple. J'ai pu avoir l'occasion d'installer et utiliser ce logiciel. Cela m'a sensibilisé à la question du *responsive design* (mise en page adaptative).

J'ai également pu apprendre de façon assez superficielle le langage XML. L'utilisation que j'en ai faite est bien en-deçà de ses possibilités.

Bien entendu, j'ai pu améliorer mes connaissances (dont l'importance variait de l'anecdotique au suffisant) dans d'autres domaines comme le SQL (utilisation massive de sous-requêtes/requêtes imbriquées, de statistiques simples), la programmation orientée objet et le génie logiciel (mise en pratique des cours de cette année).

Avant la fin de ce stage, j'aurai sans doute à faire l'installation, la configuration et la sécurisation d'un serveur LAMPP sur une Debian (en m'aidant de la vaste documentation à ce sujet)

Par la suite, j'aurai à me connecter en SSH à ce serveur ce qui, là encore, constituera un excellent entraînement et une mise en pratique des cours de cette année.

Chapitre 5

Conclusion

5.1 Réalisations

Les demandes telles que formulées dans le cahier des charges originel ont toutes été remplies. En revanche, de nombreuses demandes supplémentaires ont été émises par le client entre-temps

Aujourd'hui, le travail côté serveur est bien avancé, le modèle MVC permettra de modifier aisément le code et et de rajouter des pages supplémentaires.

Un commerçant peut :
- Accéder de façon sécurisée à son compte
- Écrire des *kuchikomi* qui seront immédiatement publiés
- Lire des statistiques inhérentes à son compte (nbre d'abonnés, et le pourcentage d'augmentation sur les 30 derniers jours, nbr de jaime, etc...)
- À la publication d'un kuchikomi, tous les abonnés de ce commerce recevront une notification.

Un utilisateur peut :
- S'inscrire automatiquement en un geste de son téléphone
- Se connecter de façon totalement automatique et invisible et se déconnecter en une touche
- S'abonner à un commerce d'un seul geste et se désabonner en une touche
- Consulter la liste de ses abonnements
- Consulter le fil des *kuchikomi* d'un commerce dont il est abonné
- Consulter un *kuchikomi* en particulier

- Dire si il apprécie un *kuchikomi*
- Consulter des données à propos d'un commerçant (données cartographiques ou de contact)

L'administrateur peut :
- Lire des statistiques générales sur l'utilisation du service
- Ajouter un commerce
- Modifier les données d'un commerce (auquel cas, les champs seront pré-remplis)
- Modifier le bandeau publicitaire

5.2 Travail à faire

Il manque encore certaines choses comme :
- Un utilisateur peut « faire passer » un *kuchikomi* à quelqu'un d'une manière ou d'une autre (SMS, mail, ?)
- Implémenter une authentification automatique propre et sécurisée.
- Refaire de façon adéquate le principe des notifications.

5.3 Évolution possible

Les améliorations possibles sont potentiellement infinies. Des pistes ont déjà été trouvées :
- Refaire entièrement la charte graphique
- À terme, l'application ne doit plus se faire par navigateur mais uniquement par l'application du téléphone qui offrira une *webview*.
- Avant de publier un *kuchikomi*, les commerçants pourraient prévisualiser leur annonce avant de la valider pour publication.
- Les commerçants pourraient modifier ou supprimer leurs *kuchikomi*.
- Les dates des *kuchikomi* pourraient être plus flexibles (période de validité au lieu de dates fixes)
- Les commerçants devraient pouvoir pré-écrire des *kuchikomi* et choisir une date de publication. Une fois cette date atteinte, le *kuchikomi* en question sera automatiquement publié.
- Utilisateurs et commerçants sont connectés automatiquement. Peut-être préféreront-ils s'identifier grâce à un identifiant et un mot de passe.

5.4 Avis personnel

Cette expérience professionnelle m'a permis de mettre en œuvre les connaissances acquises à l'université et d'en apprendre un grand nombre d'autres. Ce projet sur lequel j'ai travaillé et travaille encore est suffisamment imposant pour me permettre d'utiliser des connaissances variées et les connecter ensemble.

J'ai également pu mettre en pratique des méthodes de développement agiles vues en cours.

Chapitre 6

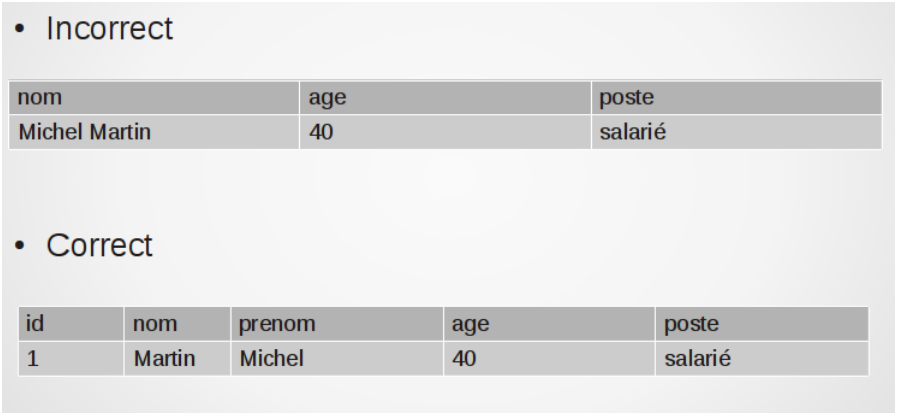
Annexes

6.1 Normalisation d'une base de données

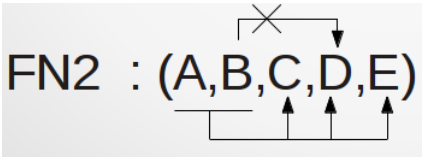
Pour normaliser une base de données, il y a un algorithme bien spécifique à suivre pour chaque forme.

Pour normaliser une base de données vers sa première forme normale, il faut effectuer les tâches suivantes :

- Chaque champ ne peut contenir qu'une seule donnée.
- Une clef primaire par table.



Pour passer à la seconde forme normale, il faut :
- Être de première forme normale.
- Tout attribut non-clef n'est pas une partie de clef.



- Incorrect

Opération bancaire	**N° compte**	**CodeOpe**	**DateOpe**	nom	prenom	libelle	somme
	123456	456789	01.01.13	Dupont	Pierre	essence	50

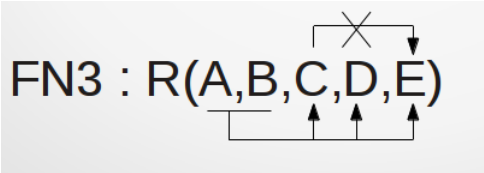
- Correct

Compte	**n°compte**	nom	prenom
	123456	Dupont	Pierre

LIBELLE	**CodeOpe**	libelle
	456789	essence

OPERATION	**n°compte**	**DateOpe**	**CodeOpe**	somme
	123456	01.01.13	456789	50

Enfin, pour passer à la troisième forme normale, il faut :
- Être de seconde forme normale.
- Tout attribut non-clé ne peut dépendre d'un autre attribut non-clef.



- Incorrect

CLIENT	**id**	nom	adresse	chambre	nb_lits
	123456	Martin	26, rue du Labrador	15	1

- Correct

CLIENT	**id**	nom	adresse	id_chambre
	123456	Martin	26 rue du Labrador	15

CHAMBRE	**id_chambre**	nb_lits
	15	1

6.2 Captures d'écran des interfaces

L'interface de l'utilisateur se