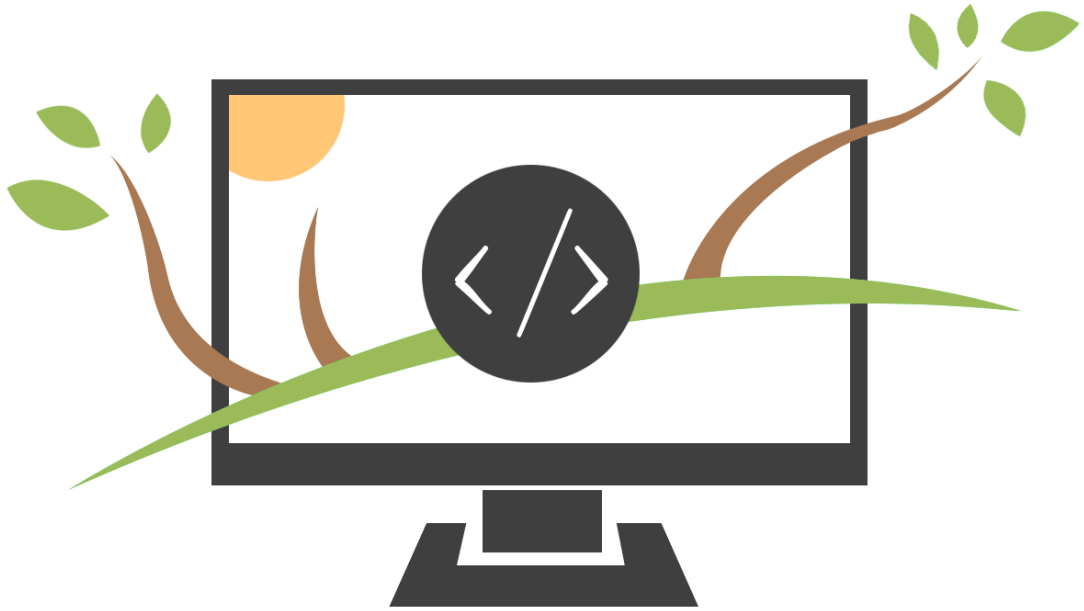
UNIVERSITÉ MONTPELLIER 2
SCIENCES ET TECHNIQUES

I.U.T MONTPELLIER

RAPPORT DE STAGE

Operationnalisation (saisie et restitution)



d'un centre d'informations scientifique

Réalisé par
PAGÈS Andy

Sous la direction de
LE FUR Jean (IRD)

Pour l'obtention du diplôme **DUT Informatique** [2014 – 2015]

REMERCIEMENTS

Je remercie tout d'abord mon tuteur de stage **M. Jean Le Fur**, pour son accueil et son aide précieuse tout au long du stage. Également, pour sa visite et ses conseils je remercie **M. Marc Joannides**, tuteur représentant de l'IUT. Enfin je tiens à remercier **M. Sylvain Piry** pour son temps et ses conseils ainsi que **Emma Artige** pour son aide dans les phases de tests.

SOMMAIRE

TABLE DES FIGURES

INTRODUCTION

Dans le cadre de la seconde année à l'**Institut Universitaire** et **Technologie** de Montpellier, un stage en entreprise est effectué afin de mettre en application les connaissances théoriques acquises durant la formation, ainsi qu'entreprendre un premier contact avec le monde professionnel. Mis en relation à travers l'IUT, j'ai décidé d'effectuer mon stage dans l'établissement **IRD Montpellier**, sous le tutorat de **M. Jean Le Fur**.

L'**IRD** est un établissement qui conduit des recherches sur les milieux intertropicaux depuis cinquante ans. Ses recherches scientifiques sont centrées sur les relations entre l'homme et son environnement dans les régions tropicales et méditerranéennes, dans la perspective d'un développement durable de ces régions. Parmi ces recherches, certaines sont traitées et insérées dans un système appelé le **Centre d'Informations (CI)**.

Objet principal de ce stage, le centre d'informations a été réalisé dans le but de mettre à disposition des informations à des non-informaticiens de manières **intelligente**. A partir d'une base de données structurée, il offre une interface de navigation intuitive à l'aide de **mots-clés**, thèmes, supports et autres type d'associations. Commencé en 2000, le projet Centre d'Informations (CI) est partiellement réalisé par des stagiaires qui se relaient chaque année.

Ce système utilise une base de données à partir de laquelle il crée l'interface de navigation. Cependant, il n'est pas doté d'un **point d'entrée** : il n'existe aucun moyen d'insérer des informations dans le système, en dehors d'insertions directes dans la base de données par un informaticien. Étant destiné à une **distribution libre**, le centre d'informations doit être appréhendable et utilisable par le plus grand nombre d'utilisateurs. Ainsi, une interface spécialisée à l'insertion de ces données est indispensable. On parle ici d'un **masque de saisie**.

L'objectif est donc de fournir une interface **ergonomique** et fonctionnelle capable d'offrir à l'utilisateur une simplicité d'insertion. De plus, ce masque doit permettre une visualisation rapide des données, ainsi que d'éventuelles modifications et suppressions.

GLOSSAIRE

Framework : ensemble de composants logiciels structurels qui permet de créer les fondations ainsi que les grandes lignes de tout ou d'une partie d'un logiciel.

Apache Tomcat : conteneur web libre de servlets.

Servelts : classe Java qui permet de créer dynamiquement des données au sein d'un serveur HTTP.

WampServer : plateforme permettant de faire fonctionner localement des scripts PHP.

Ontologie : ensemble structuré de termes et concepts représentant le sens d'un champ d'informations, que ce soit par les métadonnées d'un espace de noms, ou les éléments d'un domaine de connaissances.

PRÉSENTATION DE L'ENTREPRISE

L'organisme qui m'accueille durant ma période de stage est l'Institut de recherche pour le développement (**IRD**). C'est un organisme français de recherche, original et unique dans le paysage européen de la recherche pour le développement. Privilégiant l'interdisciplinarité, l'IRD centre ses recherches, depuis plus de 65 ans, sur les relations entre l'homme et son environnement en Afrique, Méditerranée, Amérique latine, Asie et dans l'Outre-Mer tropical français. Ses activités de recherche, de formation et d'innovation ont pour objectif de contribuer au développement social, économique et culturel des pays du Sud.

Cependant, le stage se déroule au Centre de Biologie pour la Gestion des Populations (**CBGP**). L'objectif du CBGP est centré sur la biologie de populations et communautés d'organismes qu'il convient de gérer car ils représentent un enjeu majeur pour l'agronomie, les forêts, la santé humaine ou la conservation de la biodiversité. Ce centre regroupe de nombreuses institutions similaires à l'IRD telles que : l'**INRA**, **Cirad** et **SupAgro**.

L'organisation générale du travail au CBGP est donc différente d'un cadre basique de développement informatique. En effet, l'environnement de travail étant concentré sur la biologie avec des objectifs différents, les méthodes étudiées à l'IUT et utilisées dans de nombreuses entreprises ne sont majoritairement pas utilisées au CBGP.

Cependant parmi les projets informatiques existants, certains appliquent tout de même, selon leurs besoins et leurs habitudes de travail, des méthodes de développements similaires aux nôtres. Il s'agit par exemple de méthodes mélangeant à la fois « **agile** » et « **cycle en V** ».

1. CAHIER DES CHARGES

1.1. Présentation du sujet

Le '**Centre d'Informations (CI)**' est une application qui vise à mettre en relation des informations unitaires sur divers domaines de recherche scientifique. L'application élabore à partir d'informations calibrées un ensemble de mots-clés sur lequel est construite une ontologie du domaine. Les trois ensembles **informations** - **mots-clés** - **typologies** conduisent à un réseau sémantique qui peut être utilisé pour naviguer dans le domaine de connaissance.

L'application à déjà été déployée sur quatre domaines de connaissances scientifiques (pêche en Guinée, écosystèmes marins, biologie des rongeurs, étude d'une réserve en Afrique). Elle se présente sous la forme d'une interface web qui offre une navigation simple et intuitive :

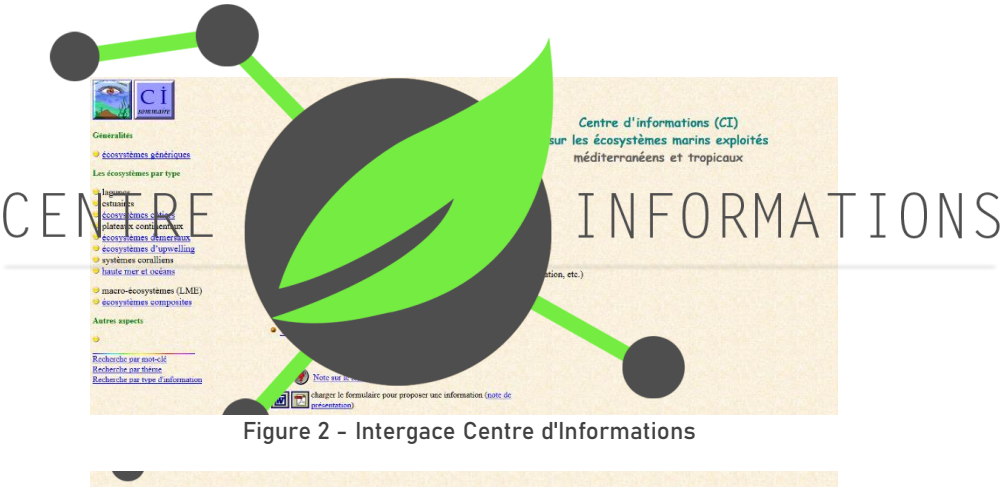


Figure 2 - Intergace Centre d'Informations

Figure 1 - Centre d'Informations : Logo

Le Centre d'Informations s'appuie sur une base de données structurée qui lie les éléments entre eux. Ainsi, un jeu de données complet est constitué d'éléments connectés entre eux de la façon suivante :

La base est centrée autour d'une information. En effet, l'information représente l'élément

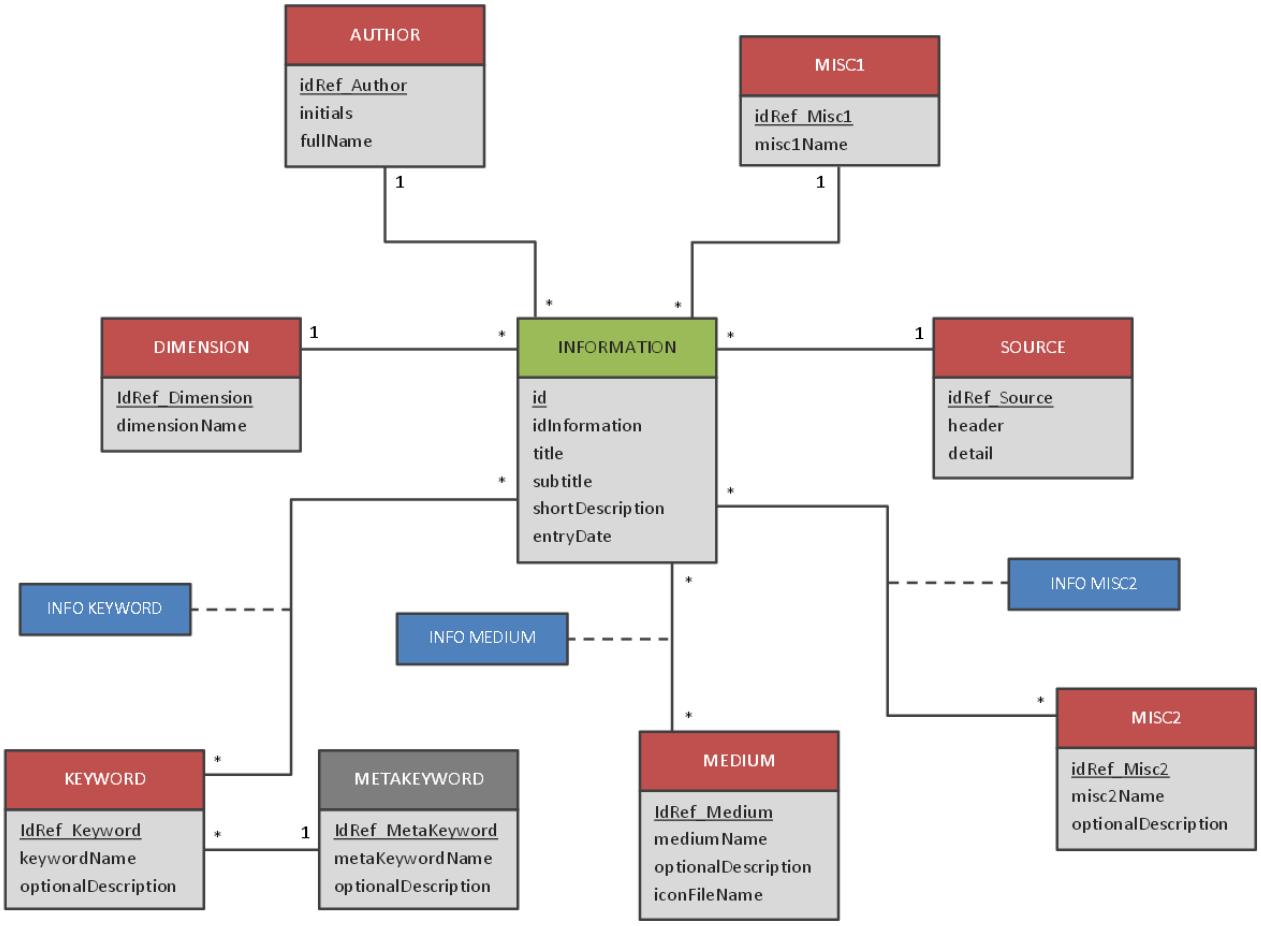


Figure 3 - Structure de données

principal, et est entourée de satellites directs et indirects. Chacune de ces entités représente dans l'interface utilisateur les éléments suivants :

- **Keywords** : Mots-clés ou « tags », ils décrivent une information autour d'un thème,
- **MetaKeywords** : Catégorie d'un mot-clé,
- **Dimensions** : Thème général d'une information,
- **Mediums** : Supports, ce sont des éléments qui accompagnent une information, par exemple une image, un graphique, ou encore un rapport,
- **Sources** : Provenance ou cadre de l'information, par exemple une équipe de recherches,
- **Authors** : Auteur de l'information,
- **Misc1 et Misc2** : ces éléments représentent des axes supplémentaires de connexions. Ils permettent d'offrir à l'utilisateur la possibilité d'aller plus loin dans la catégorisation des informations. Par exemple Misc1 pourrait représenter une marque de voiture pour un utilisateur souhaitant implémenter un centre d'informations sur les voitures.

Sur le Centre d'Informations, il est donc possible de naviguer à travers ces entités. Par exemple, il est possible de visualiser toutes les informations en relation avec le thème « Rongeurs ».

Figure 4 - Navigation par mots-clés

1.2. Problématique

Le module de sortie du Centre d'Informations est donc déjà implémenté et opérationnel. Cependant, son point d'entrée est à revoir : hormis une ancienne version dépassée et peu utilisable, il n'existe **aucun** moyen d'insérer des informations dans une base pour une personne non-informaticienne. Il est donc indispensable de combler ce manque afin d'offrir à tous les utilisateurs une expérience agréable d'utilisation du centre d'informations, à la fois lors de l'insertion que lors de la visualisation.

Objectif : Développer un masque de saisie pour le Centre d'Informations

Avant d'élaborer un cahier des charges, il est nécessaire d'évaluer et analyser le système d'insertion existant, afin d'en tirer les points **positifs** et **négatifs**.

Fonctionnel seulement sous Microsoft Access, l'ancienne interface de saisie se présente sous la forme suivante :

Figure 5 - Ancien masque : accueil

Cette

- **Saisie** : insérer une nouvelle information,
- **Consultation** : visualiser, modifier, supprimer des données.

Points positifs : Bien que dépassée, cette page est claire et intuitive, elle permet d'accéder rapidement aux fonctionnalités du masque.

Points négatifs : L'apparence, l'agencement des éléments ainsi que le bouton « Quitter l'application ».

La fonctionnalité principale du masque est la saisie d'information. Dans l'ancien masque, après avoir cliqué sur « **SAISIE** », elle se présente ainsi :

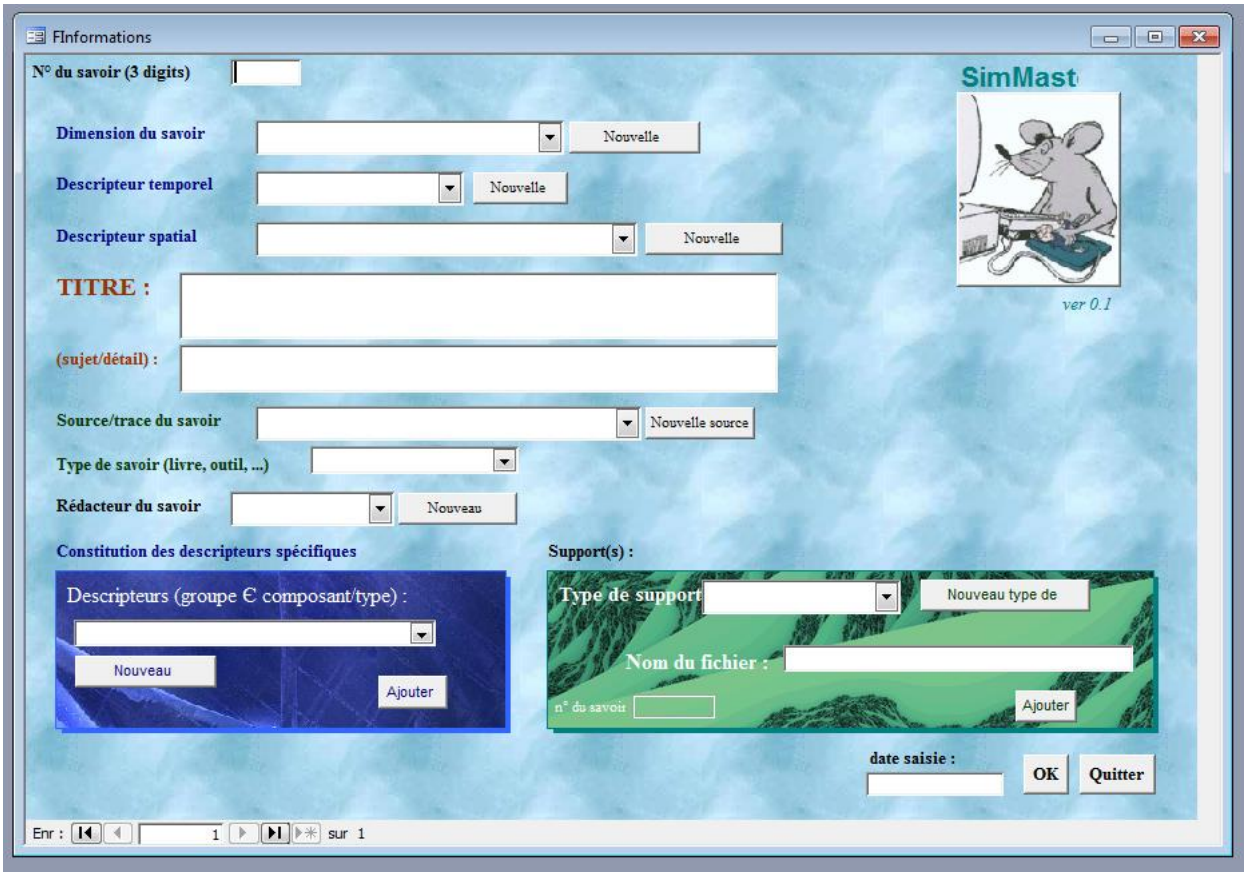


Figure 6 - Ancien masque : saisie

Ce formulaire de saisie est structuré en trois modules :

- Définition de l'**information** : Titre, Description, Auteur…etc.,
- Association des **mots-clés**,
- Association des **supports**.

Points positifs : Formulaire complet et compact, toute la page regroupe les éléments nécessaires à l'ajout d'une information.

Points négatifs : Champs peu clairs, apparence dépassée, mots parfois peu lisibles, beaucoup de données d'un coup et manque d'automatisation (date).

Sur cette interface on note la présence de champs non décrit auparavant, comme par exemple « Descripteur spatial » ou « Type de savoir ». Ces champs n'ont pas été mentionnés car ne sont plus présents dans la base de données étudiée. Ils ont été dépréciés pour des raisons de clarté et d'intuitivité.

La seconde fonctionnalité de ce masque est la **visualisation** et **modification** des données. Cette interface doit être robuste et intègre, afin d'assurer une gestion correcte des données. A

partir du bouton « CONSULTATION D'ENREGISTREMENTS POUR MODIFICATION », le masque déprécié se présentait ainsi :

La

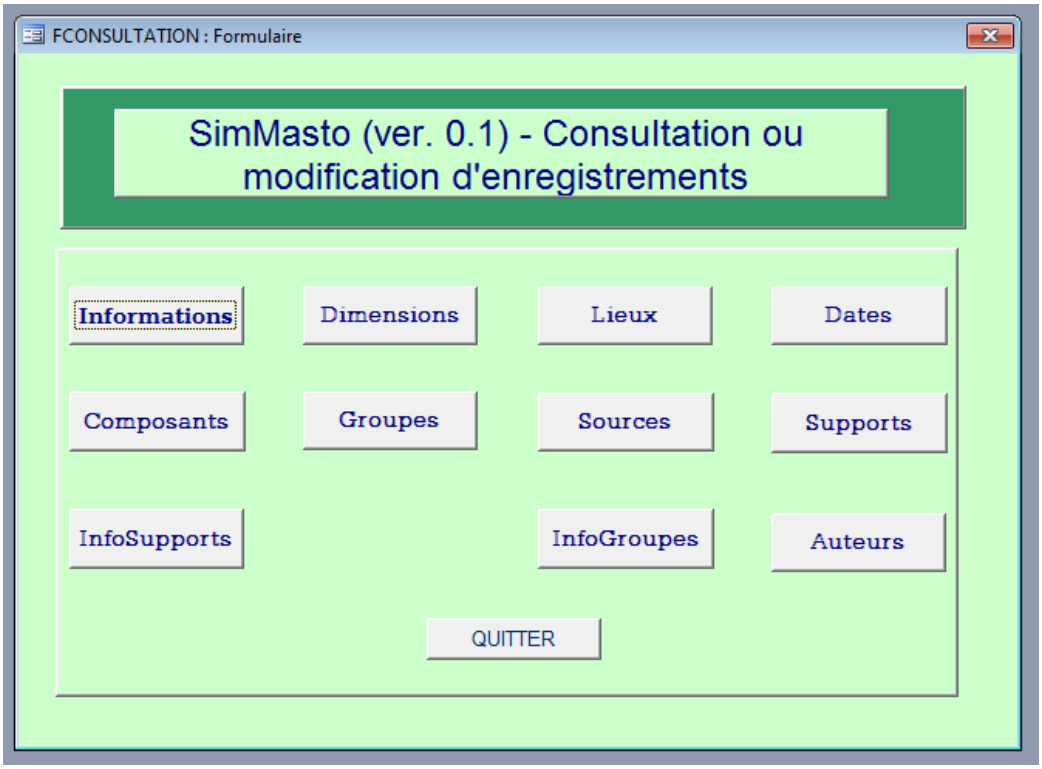
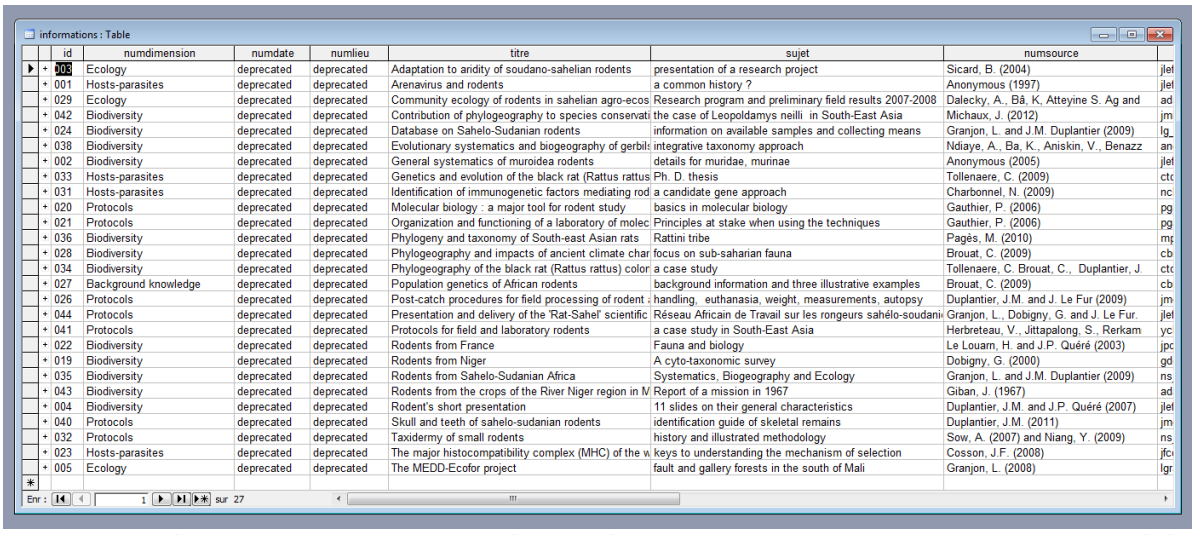


Figure 7 - Ancien masque : menu consultation



id	numdimension	numdate	numlieu	titre	sujet	numsource
003	Ecology	deprecated	deprecated	Adaptation to aridity of soudano-sahelian rodents	presentation of a research project	Sicard, B. (2004)
001	Hosts-parasites	deprecated	deprecated	Arenavirus and rodents	a common history ?	Anonymous (1997)
029	Ecology	deprecated	deprecated	Community ecology of rodents in sahelian agro-ecos	Research program and preliminary field results 2007-2008	Dalecky, A., Bâ, K, Atteyine S. Ag and
042	Biodiversity	deprecated	deprecated	Contribution of phylogeography to species conservati	the case of Leopoldamys neilli in South-East Asia	Michaux, J. (2012)
024	Biodiversity	deprecated	deprecated	Database on Sahelo-Sudanian rodents	information on available samples and collecting means	Granjon, L. and J.M. Duplantier (2009)
038	Biodiversity	deprecated	deprecated	Evolutionary systematics and biogeography of gerbils	integrative taxonomy approach	Ndiaye, A., Ba, K., Aniskin, V., Benazz
002	Biodiversity	deprecated	deprecated	General systematics of muroidea rodents	details for muridae, murinae	Anonymous (2005)
033	Hosts-parasites	deprecated	deprecated	Genetics and evolution of the black rat (Rattus rattus	Ph. D. thesis	Tollenaere, C. (2009)
031	Hosts-parasites	deprecated	deprecated	Identification of immunogenetic factors mediating rod	a candidate gene approach	Charbonnel, N. (2009)
020	Protocols	deprecated	deprecated	Molecular biology : a major tool for rodent study	basics in molecular biology	Gauthier, P. (2006)
021	Protocols	deprecated	deprecated	Organization and functioning of a laboratory of molec	Principles at stake when using the techniques	Gauthier, P. (2006)
036	Biodiversity	deprecated	deprecated	Phylogeny and taxonomy of South-east Asian rats	Rattini tribe	Pagès, M. (2010)
028	Biodiversity	deprecated	deprecated	Phylogeography and impacts of ancient climate char	focus on sub-saharian fauna	Brouat, C. (2009)
034	Biodiversity	deprecated	deprecated	Phylogeography of the black rat (Rattus rattus) color	a case study	Tollenaere, C. Brouat, C., Duplantier, J.
027	Background knowledge	deprecated	deprecated	Population genetics of African rodents	background information and three illustrative examples	Brouat, C. (2009)
026	Protocols	deprecated	deprecated	Post-catch procedures for field processing of rodent	handling, euthanasia, weight, measurements, autopsy	Duplantier, J.M. and J. Le Fur (2009)
044	Protocols	deprecated	deprecated	Presentation and delivery of the 'Rat-Sahel' scientific	Réseau Africain de Travail sur les rongeurs sahélo-soudani	Granjon, L., Dobigny, G. and J. Le Fur.
041	Protocols	deprecated	deprecated	Protocols for field and laboratory rodents	a case study in South-East Asia	Herbreteau, V., Jittapalong, S., Rerkam
022	Biodiversity	deprecated	deprecated	Rodents from France	Fauna and biology	Le Louarn, H. and J.P. Quéré (2003)
019	Biodiversity	deprecated	deprecated	Rodents from Niger	A cyto-taxonomic survey	Dobigny, G. (2000)
035	Biodiversity	deprecated	deprecated	Rodents from Sahelo-Sudanian Africa	Systematics, Biogeography and Ecology	Granjon, L. and J.M. Duplantier (2009)
043	Biodiversity	deprecated	deprecated	Rodents from the crops of the River Niger region in M	Report of a mission in 1967	Giban, J. (1967)
004	Biodiversity	deprecated	deprecated	Rodent's short presentation	11 slides on their general characteristics	Duplantier, J.M. and J.P. Quéré (2007)
040	Protocols	deprecated	deprecated	Skull and teeth of sahelo-sudanian rodents	identification guide of skeletal remains	Duplantier, J.M. (2011)
032	Protocols	deprecated	deprecated	Taxidermy of small rodents	history and illustrated methodology	Sow, A. (2007) and Niang, Y. (2009)
023	Hosts-parasites	deprecated	deprecated	The major histocompatibility complex (MHC) of the w	keys to understanding the mechanism of selection	Cosson, J.F. (2008)
005	Ecology	deprecated	deprecated	The MEDD-Ecofor project	fault and gallery forests in the south of Mali	Granjon, L. (2008)

Figure 8 - Ancien masque : consultation

ment

Points positifs : Visualisation directe des données, modification en temps réel.

Points négatifs : Design dépassé, de nombreux éléments dispensables lors d'une simple visualisation, le manque de sobriété.

L'ancien masque est donc un masque **simple** et **fonctionnel**, permettant une gestion efficace des données. Cependant, en plus de son obsolescence, son implémentation est désormais inadéquate compte tenu des récentes évolutions de la base. Il présente des défauts tels que le manque d'ergonomie dans certaines interfaces, la présence d'éléments dépréciés, le manque de clarté ou encore l'apparence dépassée. D'autre part, il ne peut être déployé sur Internet et dépend d'un logiciel propriétaire et **ne peut donc pas être utilisé librement.**

A l'aide de cette analyse, il est désormais possible de déterminer et définir les **objectifs** et **contraintes** du nouveau masque à développer. En effet, à travers les points négatifs relevés sur l'ancien masque, la vision finale du projet devient claire et permet de formuler les éléments du **cahier des charges**.

1.3. Vision du commanditaire

Depuis la présentation même du sujet, la vision globale du projet contient les éléments essentiels suivants, demandés par le client :

- **La vitesse de saisie** : l'application doit offrir une interface ergonomique qui permet une insertion d'informations rapide,
- **La robustesse** : la saisie et la consultation des données doit respecter l'intégrité du système et du schéma de données.

Ces éléments sont les lignes directrices de ce projet. Il est donc important de toujours s'y référer afin d'implémenter une application correspondant aux besoins. A partir de ces éléments fondamentaux et des critères définis grâce à l'analyse de l'ancien masque, il est possible de mettre au point la vision globale du projet :

Le nouveau masque du Centre d'Informations (CI) doit être une application ergonomique, robuste et design. Destinée à tout utilisateur, cette application doit être simple, claire, intuitive et sobre. Permettant à la fois insertion, modification, et suppression des données, le nouveau masque doit respecter l'intégrité de celles-ci et assurer une saisie sans perte ni mauvaise structuration.

1.4. Besoin non-fonctionnels

1.4.1. Spécifications techniques

Le développement du nouveau masque doit être en accord avec les précédents choix faits pour le **Centre d'Informations**. Celui-ci, implémenté à l'aide de Java, se présente sous la forme d'une application web « **Apache Tomcat** ». Ainsi, le choix du commanditaire pour cette nouvelle application est un logiciel appelé « **WaveMaker** ».

WaveMaker est un logiciel de développement d'application web orienté vers la gestion de bases de données. Il permet à l'aide d'une interface simple et complète de combiner de

nombreuses notions de programmation telles que **Java**, **HQL**, **JQuery** et d'autres composants d'applications.

Le choix de WaveMaker pour le développement du nouveau masque s'explique par les raisons suivantes :

- Spécialement conçu pour la gestion de base de données, les fonctionnalités de WaveMaker faciliteront le développement de l'application,
- WaveMaker crée des applications identiques au Centre d'Information, c'est-à-dire des applications « **Apache Tomcat** »,
- Très modulable et extensible, WaveMaker offre la possibilité de développer de nombreuses fonctionnalités de tout type,
- Des personnes compétentes sur le centre peuvent faciliter le développement.

1.4.2. Ergonomie

L'ergonomie est un élément important dans la conception du nouveau masque. En effet la mise en place d'une interface ergonomique permettra une saisie rapide et pratique.

Afin d'analyser les besoins mis en jeu pour une telle interface, il est nécessaire d'entreprendre une démarche de **recherche** et d'approfondissement des notions liées à l'ergonomie, afin d'en maitriser les **principes**.

Les recherches qui ont été effectuées permettent de définir deux composantes indispensables dans l'implémentation d'une interface ergonomique :

- L'ergonomie générale de l'application,
- Le design.

Ces deux éléments ont alors été définis par des critères essentiels dans la mise en place d'une application ergonomique :

ERGONOMIE GÉNÉRALE
▪ Simplicité, ▪ Sobriété, ▪ Choix et liberté, ▪ Repères, ▪ Indications, ▪ Catégorisation, ▪ Navigations cognitive, ▪ Continuité, ▪ FeedBack, ▪ Simplification générale

DESIGN
▪ Contraste, ▪ Visibilité, ▪ Légèreté,

Une interface ergonomique est donc une interface **dynamique**, **compréhensible**, et en **communication** avec l'utilisateur (rapide, facile et efficace). Elle doit lui offrir une expérience guidée, à l'aide d'indications et de retours, mais doit lui laisser la liberté de sélection et modification. Il est essentiel que cette interface soit **sobre** et structurée à l'aide d'éléments **constants** et logiques afin d'offrir des repères de navigation.

Enfin, à travers des jeux de couleurs, thèmes, illustrations et images, une interface ergonomique doit être percevable de manière **rapide** et **intuitive**.

Afin d'en tirer le meilleur modèle de développement possible, ces critères ont été rassemblés et triés selon leur **valeur** dans le projet :

Critères	Valeurs
Sobriété	★★★
Simplicité	★★★
Choix et liberté	★★★
Repères	★★★
Indications claires	★★★
Catégorisation	★★★
Images	★★☆
Navigation cognitive	★★☆
Préférences	★☆☆
Continuité	★☆☆
FeedBack	★☆☆
Simplification	★☆☆
Couleurs	★☆☆

On remarque à l'aide de ce tri les critères principaux à respecter. Ceux-ci seront prioritaires et indispensables à la réalisation de l'objectif ergonomique.

1.5. Besoins fonctionnels

Une fois la vision globale du projet établie et les recherches préliminaires effectuées, les fonctions du cahier des charges peuvent alors être définies. Celles-ci ont été développées au fur et à mesure durant le développement, à travers un système itératif (défini dans le rapport d'activité).

Chacune des fonctionnalités (8 au total) a été catégorisée par un ordre de priorité, selon leur valeur dans le projet.

CRÉATION D'UN DESCRIPTEUR SIMPLE

Description : Si le descripteur simple recherché n'existe pas, l'utilisateur remplit le formulaire de définition du nouvel élément qui sera inséré dans la base de données.
Contrainte(s) :

- Respect des champs obligatoires,
- Auto-complétion des champs de type « liste »,
- Vérification d'unicité.

Priorité : 

CRÉATION D'UN DESCRIPTEUR COMPOSITE

Description : Si le descripteur composite recherché n'existe pas, l'utilisateur remplit le formulaire de définition du nouvel élément qui sera inséré dans la base de données.
Contrainte(s) :

- Respect des champs obligatoires,
- Auto-complétion des champs de type « liste »,
- Vérification d'unicité,
- Vérification de l'existence des fichiers joints,
- Vérification des clés étrangères.

Priorité :

CRÉATION D'UNE INFORMATION

Description : L'utilisateur remplit un formulaire définissant la nouvelle information qui sera insérée dans la base de données.
Contrainte(s) :

- Génération automatique d'identifiant,
- Respect des champs obligatoires,
- Auto-complétion des champs de type « liste »,
- Vérification d'unicité,
- Vérification de l'existence des fichiers joints,
- Vérification des clés étrangères.

Priorité : ■■■■■

CRÉATION D'UNE RELATION COMPOSITE

Description : L'utilisateur sélectionne les entités à lier à l'information concernée. Cette fonctionnalité ne s'attribue qu'aux descripteurs composites.
Contrainte(s) :

- Auto-complétion des champs de type « liste »,
- Vérification d'unicité.

Priorité : ■■■■□

CHANGEMENT BASE DE DONNÉES

Description : L'utilisateur se connecte à la base de données qu'il souhaite pour modifier et gérer les entités.
Contrainte(s) :

- Vérification existence base de données,
- Vérification Utilisateur/Mot de passe,
- Erreur précises.

Priorité : ■■■■□

MODIFICATION / SUPPRESSION

Description : L'utilisateur modifie ou supprime n'importe quelle entité sélectionnée.
Contrainte(s) :

- Vérification liens : suppression seulement si entité orpheline.

Priorité : ■■■□□

CONTRÔLE D'INTÉGRITÉ

2. RAPPORT TECHNIQUE

Dans ce rapport technique seront abordés la **conception**, le **résultat** et les **perspectives de développement**. Malgré les nombreux éléments techniques du projet, une synthèse générale des points important sera réalisée afin d'en faciliter la compréhension et la visualisation du résultat.

2.1. Conception

2.1.1. Diagrammes

La conception des fonctionnalités de l'application a permis de définir une première approche de la relation voulue entre l'utilisateur et le système. En effet, dans le but d'offrir une interface ergonomique et intuitive, il est important de comprendre quelles seront les utilités principales de l'utilisateur et comment seront-elles **réalisées**.

Ainsi, le premier diagramme réalisé afin de concevoir les interactions de l'utilisateur est un diagramme de **cas d'utilisation**, décrivant les principales fonctionnalités recherchées et utilisées par l'utilisateur :

Défini à l'aide du commanditaire, on note donc un tronc principal orienté sur la **création** et la

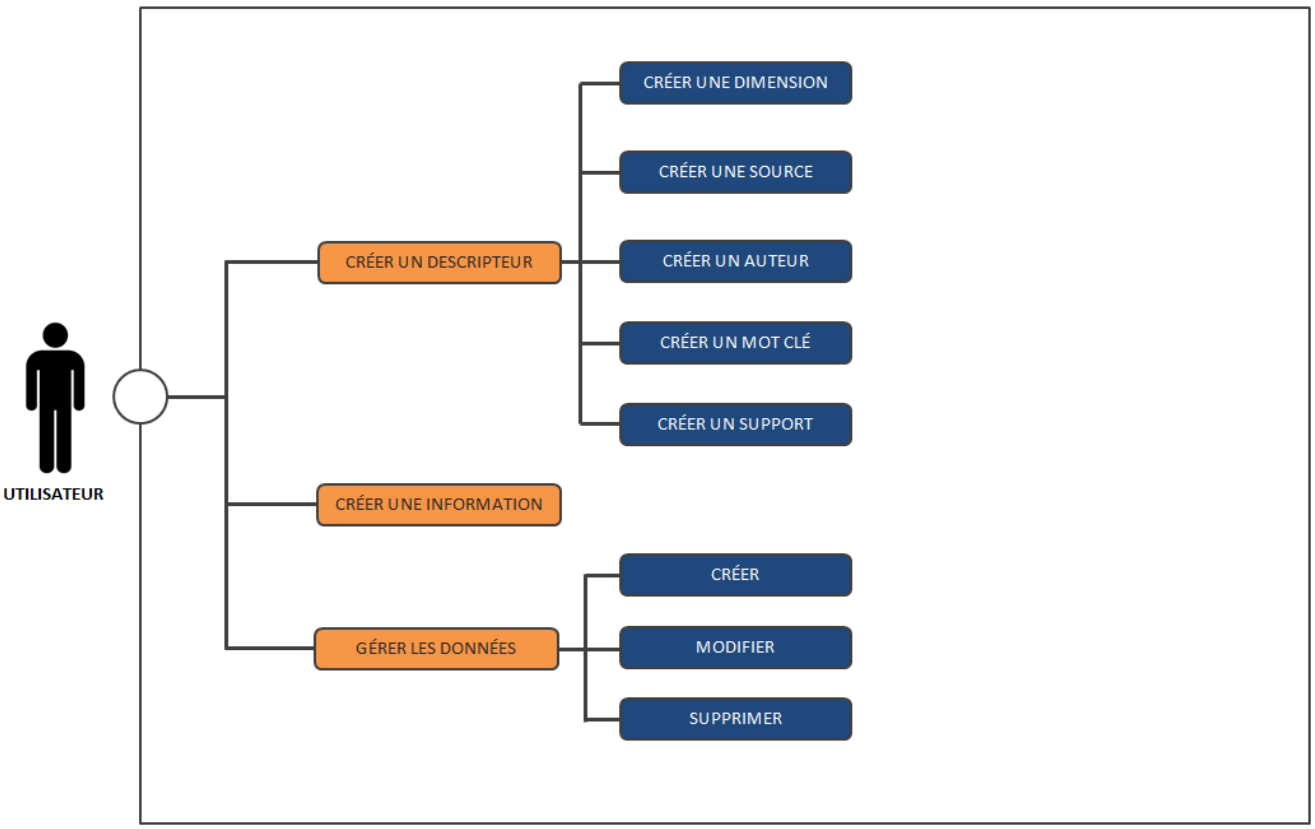


Figure 9 - Diagramme : Cas d'utilisation

gestion de données. Ces fonctionnalités sont donc les premières à réaliser et permettront d'implémenter un module important de l'application.

Afin de concevoir avec précision le déroulement de ces fonctionnalités, il est important de visualiser les étapes d'interactions entre l'utilisateur et le système. Ainsi, il sera possible de développer une application répondant à ces critères. Pour cela, la conception d'un diagramme de séquence sur la fonctionnalité « **Créer un descripteur** » permet une visualisation claire :

Il est donc possible de définir un système de saisie basé sur :

- Une vérification sur le **format** (contenu géré par le masque),

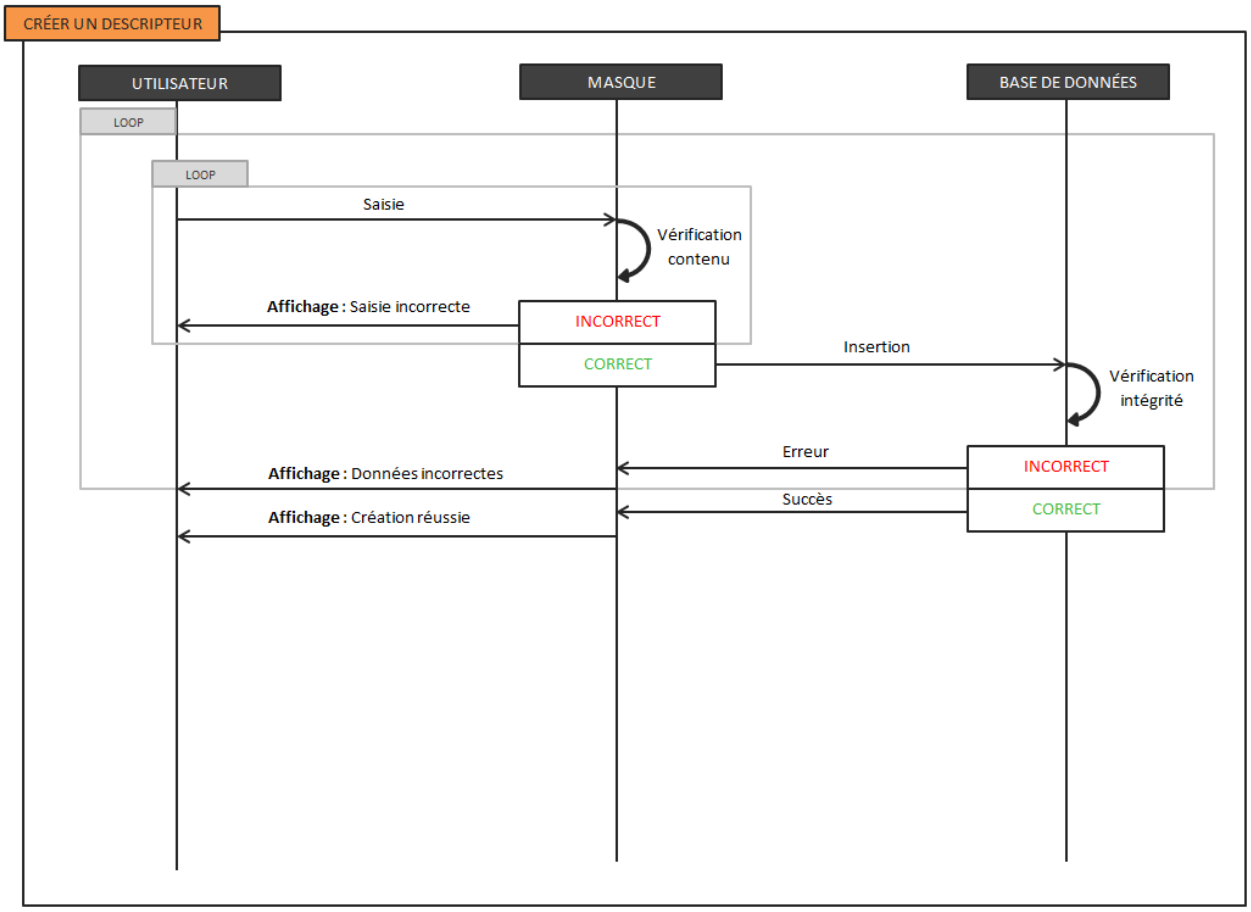


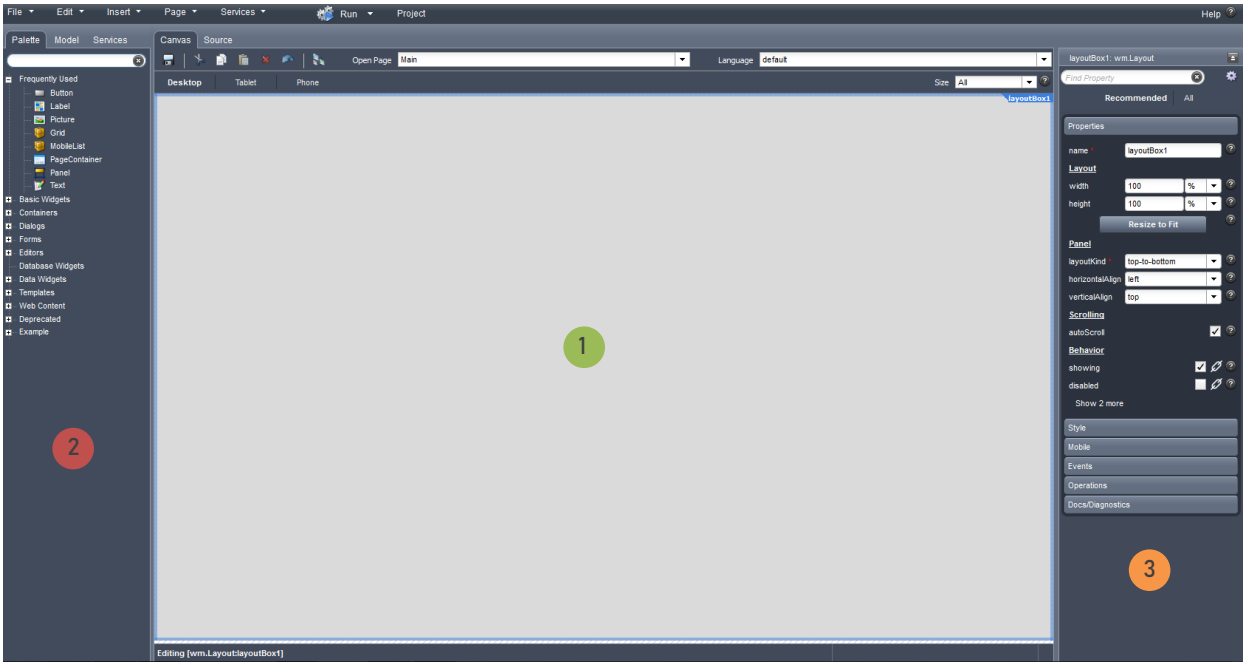
Figure 10 - Diagramme : Séquence

- Une vérification sur l'**intégrité** des données (unicité).

Ici on s'aperçoit du rôle intermédiaire du masque entre l'utilisateur et la base de données. Son rôle apparait donc crucial pour l'ensemble de l'application.

2.1.2. WaveMaker

Le développement de l'application étant effectué à l'aide du logiciel **WaveMaker**, la première étape était donc son installation et sa prise en main. Celui se présente sous la forme suivante :



WaveMaker est un logiciel basé sur un fonctionnement de Widgets. Il permet, sans faire appel à des notions de programmation, de manipuler des objets avec facilité en les faisant communiquer entre eux. Orienté vers l'application web, il est alors optimal pour mettre en forme une interface simple et intuitive.

Le fonctionnement de WaveMaker se sépare en quatre parties :

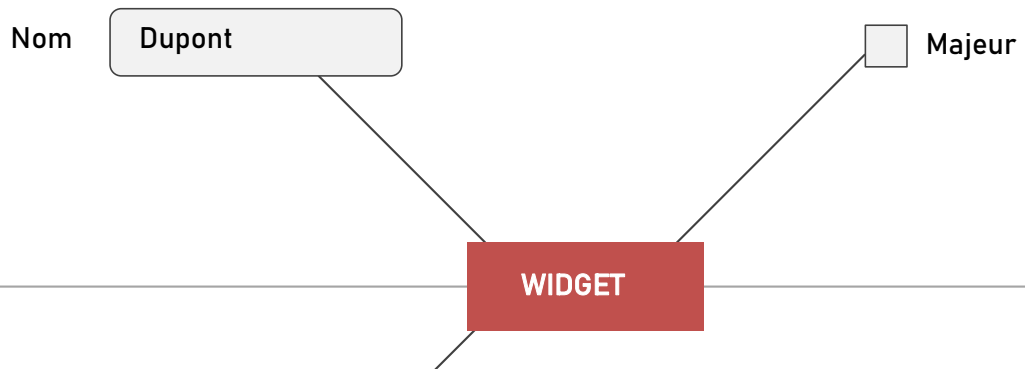
1. **Panneau central** ou « **Canvas** », il permet la visualisation de l'application,
2. Panneau des **widgets**, il recense l'ensemble des widgets par catégorie,
3. **Propriétés**, cet onglet permet de paramétrer le widget sélectionné,
4. **Barre des menus**, regroupe l'ensemble des fonctionnalités du projet.

Contrairement à un développement basique, cette interface est très intuitive et permet une construction rapide d'application. En effet, il suffit par exemple de faire glisser un widget sur le panneau central pour l'instancier et le paramétrer.

Cependant, **pourquoi WaveMaker et que sont les widgets ?**

2.1.2.1. Widgets

Un widget définit un composant d'interface graphique avec lequel un utilisateur peut interagir. Ils peuvent être une zone de texte, une liste déroulante, un bouton, une case à cocher et bien d'autres éléments :



Ce fonctionnement de widget n'existe pas seulement avec WaveMaker mais sur bien d'autres logiciels d'aide au développement. Alors **pourquoi WaveMaker** ?

WaveMaker possède parmi tous ces widgets, un groupe entier de widgets dédiés à la gestion de **base de données** : ce sont des outils très pratiques, qui permettent en quelques clics de mettre en place une application web de gestion de base données très complète. A l'aide de ces widgets, il est possible de :

- Afficher,
- Insérer,
- Modifier,
- Supprimer.

Le commanditaire a donc choisi ce logiciel pour ses applications dans les bases de données. En effet, en plus des widgets très **puissants** il est possible, pour des fonctionnalités plus particulières et poussées, d'intégrer des requêtes de données personnalisées. Par exemple, il est possible de récupérer le compte d'un attribut précis d'une table et de l'intégrer dans l'application.

Afin d'utiliser ces widgets, il est nécessaire d'importer une base de données dans le projet WaveMaker :

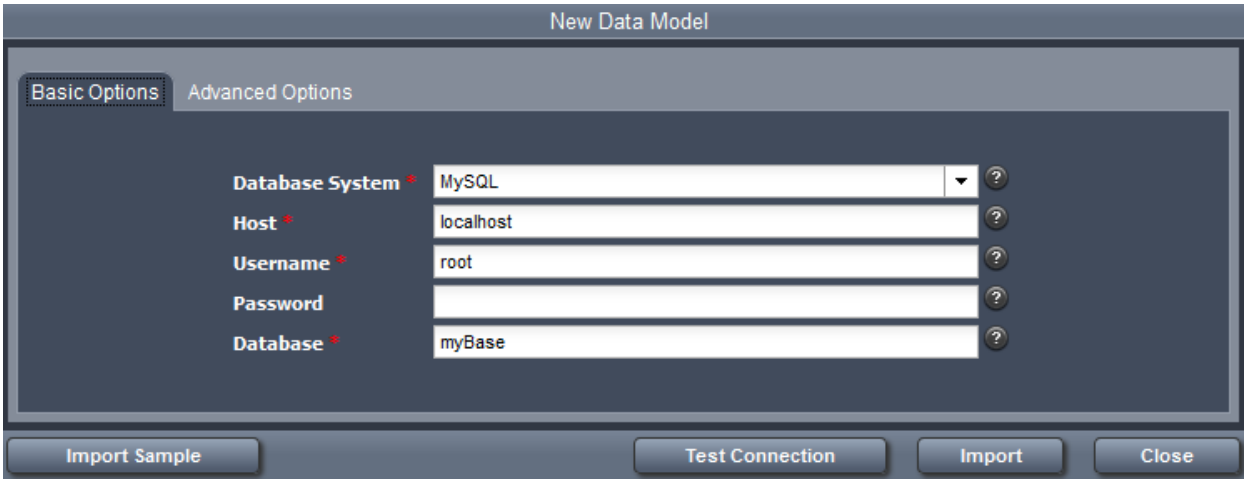


Figure 11 - WaveMaker : connexion

A partir de cette action, WaveMaker génère une structure qui lui permet d'interagir avec la base données. Ainsi, il propose désormais un widget pour chaque table de la base, de la façon suivante :

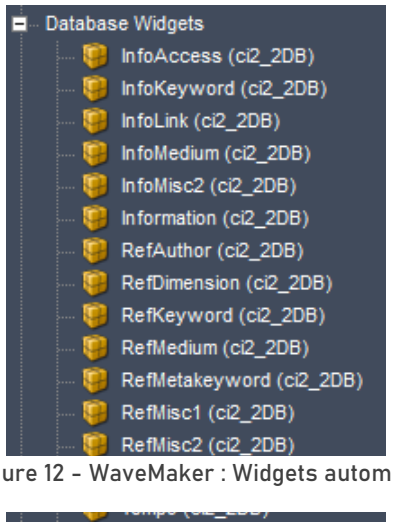


Figure 12 - WaveMaker : Widgets automatiques

Chacun de ces widgets est automatiquement et offrent les possibilités suivantes :

- Une visualisation immédiate du contenu de la table,
- Une description pour chaque ligne sélectionnée,
- Une possibilité d'insertion, modification et suppression.

Il suffit alors à l'aide d'un seul clic de faire glisser le widget, et celui-ci génère alors l'interface suivante :

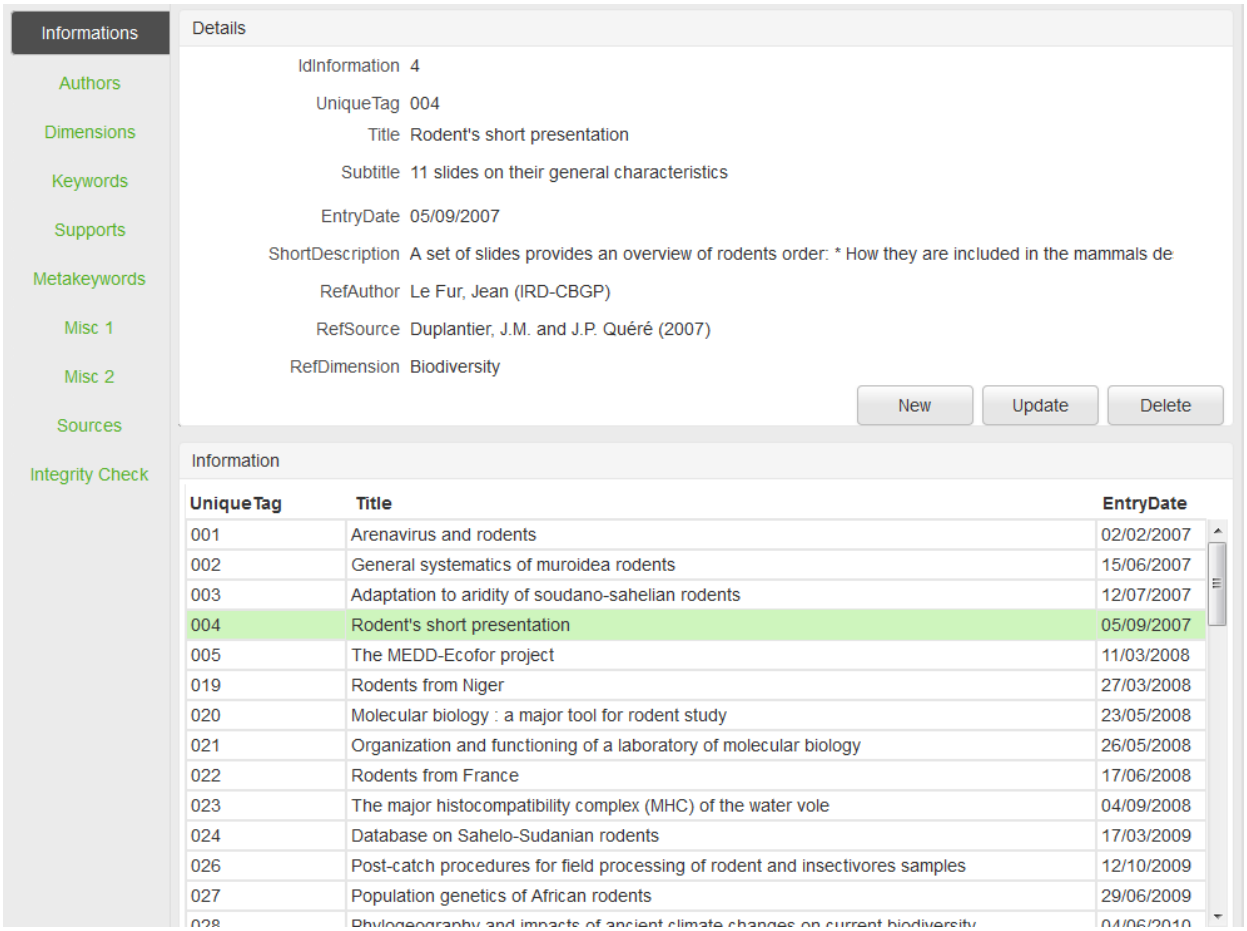


Figure 13 - WaveMaker : Interface automatique

Cette fonctionnalité est la plus **basique** du logiciel, et donc très générique. Elle est ici utilisée seulement pour l'interface de **consultation des données**, car cela offre une navigation simple, claire et intuitive.

Il est donc important de remarquer que l'interface d'insertion proposée dans ces widgets automatiques ne **correspond pas** à celle recherchée : le masque doit être doté de formulaires et interfaces de saisie intelligents, ergonomique, guidant l'utilisateur à travers l'insertion. Il faut donc utiliser autre chose pour les fonctionnalités suivantes :

- Insertion de descripteurs,
- Insertion d'informations,
- Ajout de relations,
- Contrôle d'intégrité,
- Paramètrage.

Pour cela, le masque a été construit à l'aide des nombreux widgets de saisie ainsi qu'une fonctionnalité très importante dans WaveMaker : le **Script**.

2.1.2.2. Script, HQL et Java

WaveMaker est un logiciel très modulable. Pour cela, il offre la possibilité d'intégrer du code **Java**, **HQL**, **CSS** ainsi que du jQuery. Cette intégration de script est très importante et permet de réaliser de nombreuses fonctionnalités :

- Les interactions **évènementielles** : apparitions, changements, déplacements,
- La gestion des **variables** : sélection de données, insertion depuis formulaires, suppressions.

Par exemple, sur la fonctionnalité « **Créer un auteur** », le jQuery agit sur plusieurs éléments :

1 Vérification du contenu des champs : ici, le champ « fullName » n'a pas été rempli, l'application affiche donc un avertissement,

2 Insertion : une fois les champs rempli, le script récupère le contenu des champs, traite les données si besoin, puis les insère dans la base de données en manipulant des objets de WaveMaker.

Il existe de nombreux types de variables WaveMaker, chacun possédant plusieurs modes de fonctionnement. De ce fait, WaveMaker offre un grand nombre de possibilités pour insérer, modifier et supprimer les données. Ici, des objets appelés « **Live Form** » sont utilisés afin d'insérer le nouvel Auteur.

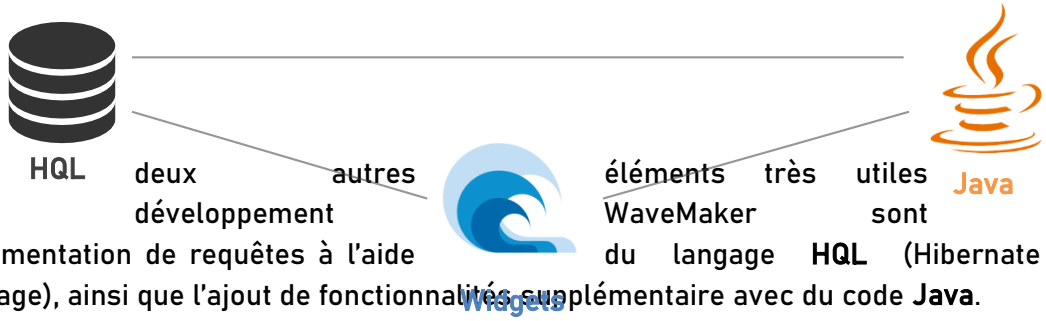
jQuery

```
//Create a new Author
CreateAuthorButtonClick: function(inSender) {
    //Check for empty field(s)
    if((this.AuthorInitialsEditor.getDisplayValue()) !== ""
        && (this.AuthorFullNameEditor.getDisplayValue()) !== "")
    {
        //RefAuhtorVar stores new data
        this.RefAuthorVar.setValue("initials", this.AuthorInitialsEditor.getDataValue());
        this.RefAuthorVar.setValue("fullName", this.AuthorFullNameEditor.getDataValue());
        //RefAuthorLiveForm inserts from previous data
        this.RefAuthorLiveForm.setDataSet(this.RefAuthorVar);
        this.RefAuthorLiveForm.insertData();
        this.AuthorInitialsEditor.clear();
        this.AuthorFullNameEditor.clear();
    } else
    {
        //Display missing informations
        if((this.AuthorInitialsEditor.getDisplayValue()) === "")
        {
            this.MissingAuthorInitialsField.setValue("showing", true);
            this.AuthorInitialsEditor.setValue("placeHolder", "authorInitials");
        }
        if((this.AuthorFullNameEditor.getDisplayValue()) === "")
        {
            this.MissingAuthorFullNameField.setValue("showing", true);
            this.AuthorFullNameEditor.setValue("placeHolder", "authorFullName");
```

La gestion d'un nouvel auteur est ici gérée en deux scénarios possibles :

- **Tous les champs** sont remplis : le données rentrées par l'utilisateur sont récupérées et transmises à un **Live Form**, qui insère par la suite les données dans la table. Enfin, les champs sont réinitialisés (vidés).
- La saisie n'est **pas complète** : les avertissements sont affichés et l'insertion n'est pas lancée.

Ainsi, de nombreuses fonctionnalités sont gérées à l'aide du script qui permet de communiquer avec toutes les entités du système, et de les lier entre eux :



Les deux autres éléments très utiles dans le développement WaveMaker sont l'implémentation de requêtes à l'aide du langage **HQL** (Hibernate Query Language), ainsi que l'ajout de fonctionnalités supplémentaire avec du code **Java**.

Les requêtes **HQL** permettent d'effectuer des sélections particulières dans la base. Par exemple, dans la fonctionnalité « **Créer une information** », il est nécessaire d'implémenter une génération automatique d'identifiant. Pour cela, il a été décidé que celle-ci serait composée d'un préfixe (« **I** » comme *Information*), suivi d'une incrémentation de l'id maximum. Par exemple, si l'id maximum est le numéro 25, alors la proposition générée sera celle-ci :

Identifiant | I26

Il est alors nécessaire de récupérer l'id maximum de la table :

HQL

```
SELECT MAX(idInformation)
FROM Information
```

Également, pour contrôler si les données saisies par l'utilisateur n'existent pas déjà dans la base, une requête **HQL** est utilisée :

HQL

```
SELECT idInformation
FROM Information
WHERE uniqueTag = :uT
```

Ici, la variable uniqueTag est comparée à un paramètre, le contenu du champ. Une fois cette requête effectuée, le script récupère le résultat et l'analyse :

jQuery

```
//CheckUniqueTagUnicity
uniqueTagExistenceServiceVarSuccess: function(inSender, inDeprecated) {
    if(this.uniqueTagExistenceServiceVar.getCount() > 0
        && (this.uniqueTagEditor.getDisplayValue()) !== "")
    {
        this.IdAlreadyExistingLabel.setValue("showing", true);
        this.MissingIdInformationField.setValue("showing", true);
    }
},
```

Dans cette fonction, le script compte le nombre d'éléments trouvés avec l'uniqueTag donné en paramètre (celui inscrit dans le champ). S'il existe au moins **une information** avec cet uniqueTag, et le champ **n'est pas vide**, alors un message d'avertissement est donné. L'insertion n'est pas acceptée tant que l'utilisateur ne propose pas une **nouvelle** valeur.

Enfin, l'intégration de programme **Java** est utilisée pour les fonctionnalités systèmes particulières. Permettant d'effectuer des algorithmes puissants, le Java permet dans le masque d'effectuer un **listage récursif** de fichiers. Par la suite, la liste de ces fichiers est proposée à l'utilisateur pour une sélection plus aisée.

```java
public void list(String path, ArrayList<String> aList)
{
    file = new File(path);
    files = file.listFiles();

    if(files == null)
    {
        return;
    }

    for(File f : files)
    {
        if(f.isDirectory())
        {
            list(f.getAbsolutePath(), aList);
        } else {
            String relativePath = f.getAbsolutePath();
            aList.add(filterPath(relativePath, beginingTree));
        }
    }
}
```

permettant de récupérer le chemin de chaque fichier à partir du dossier. Par exemple, un élément de la liste peut être « **dossier/sous-dossier/fichier1.txt** » si le dossier donné en paramètre est « **dossier** ».

2.1.3. Déploiement : problème technique

Une fois la première version du masque établie, il est important de s'intéresser au **déploiement** de l'application afin de s'assurer que celui-ci est possible et conforme.

Le Centre d'Informations étant une application destinée au **plus grand nombre** d'utilisateurs, celle-ci doit offrir la possibilité de se déployer sur toute plateforme, et être capable de s'adapter à toute base de données (de type Centre d'Informations).

Pour cela, le masque doit être capable de modifier la base de données sur laquelle il agit :

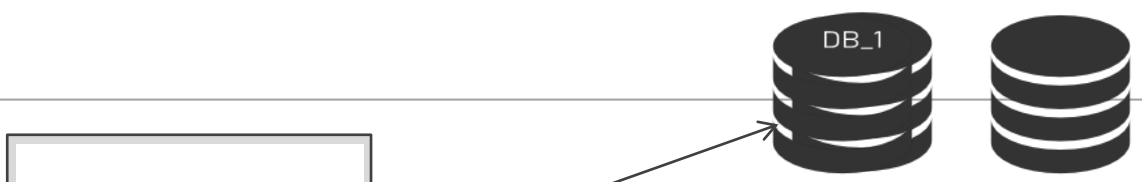


Figure 14 - Déploiement masque

Après de nombreuses recherches, la conclusion sur cette fonctionnalité est la suivante :

> WaveMaker **ne propose pas** la possibilité d'implémenter une application capable de changer de base de données à volonté.

En effet, lorsqu'une base de données est importée dans l'application, celle-ci est insérée dans le code du projet de manière non générique, de ce fait il n'est pas possible de la modifier. Cela implique que pour chaque base de données sur laquelle un utilisateur désire utiliser le masque, il serait nécessaire d'effectuer un nouveau masque.

Cette contrainte, bien que imprévue, **ne peut exister**. Il est donc nécessaire de modifier l'outil de développement utilisé pour l'implémentation du masque, malgré les nombreux avantages de WaveMaker.

2.1.4. De WaveMaker au PHP

Le développement du masque sous WaveMaker est donc compromis. Il nécessaire d'avancer vers une nouvelle méthode plus souple, qui permettra l'adaptation du masque à n'importe quelle base de données.

Cependant, le temps passé au développement sous WaveMaker **n'est pas en vain**. Bien au contraire, l'implémentation de la première version du masque à l'aide de cette interface simple et intuitive a permis de mettre en place les contraintes et les besoins mis en jeu. En effet, la **structure principale** de l'application étant établie, il est possible d'utiliser ce travail comme **maquette** et **modèle** de développement, et donc de procéder à une nouvelle méthode plus rapidement.

Le choix du nouveau mode de développement doit donc répondre aux critères suivants :

- **Modulable** : permettre l'implémentation des nombreuses fonctionnalités requises,
- **Puissant** : gérer un nombre important de données rapidement,
- **Adaptable** : permettre le déploiement de l'application sur toutes les plateformes et liée à toute base de données sélectionnée.

Afin de répondre à ces contraintes le choix est donc le langage de programmation web PHP :

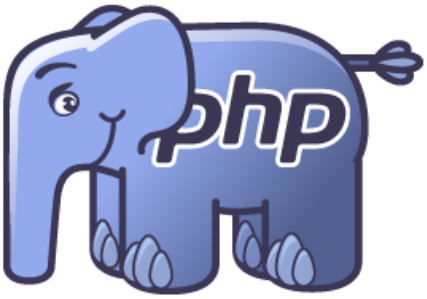


Figure 15 - Logo PHP

Le **PHP** (*Hypertext Preprocessor*) est un langage utilisé pour implémenter des applications web dynamiques. Orienté objet, il est très **modulable** et peut être utilisé pour effectuer de nombreuse fonctionnalités. Il est très utilisé et possède un **rapport puissant** avec les bases de données.

2.1.5. PHP

Le masque final est donc réalisé en **PHP**. Cette partie décrit l'architecture du projet, son fonctionnement, son arborescence ainsi que les outils utilisés dans l'ensemble du programme.

2.1.5.1. Architecture

Afin de mettre en place une application sans redondance et structurée, l'architecture choisie pour l'application est une méthode appelée « **MVC** ». Ce modèle de développement est souvent utilisé et permet de séparer et ordonner les différents composants d'une application web de la façon suivante :

- M « **Modèle** » : ensemble des fonctionnalités et de la structuration des données, c'est le cœur des méthodes et des manipulations principales de l'application.
- V « **Vue** » : simple affichage des données renvoyées par le modèle, les vues ne font que présenter et n'effectue aucune action. Elles peuvent de plus servir de réception pour les actions de l'utilisateur (formulaires, évènements).
- C « **Contrôleur** » : gestion et coordination de l'ensemble du projet, le contrôleur synchronise le modèle et la vue et enclenche les actions à effectuer.

Il est possible de schématiser le fonctionnement d'une application **MVC** :

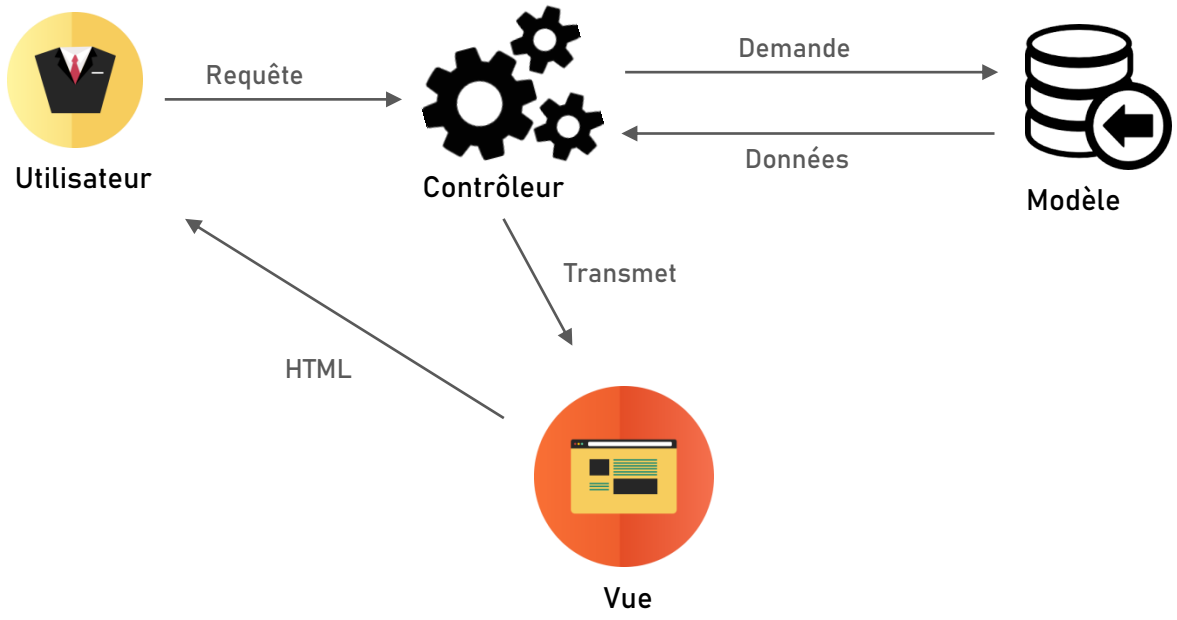


Figure 16 - Fonctionnement MVC

2.1.5.2. Arborescence

Basé sur la structure **MVC**, l'arborescence du projet présente donc ses éléments, en plus de certaines autre catégories secondaires au fonctionnement de l'application : les fichiers de configurations, la documentation, les images et enfin les fichiers de style.

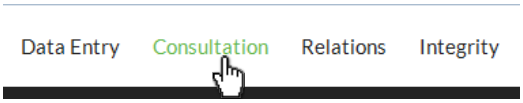
Voici la description de cette **arborescence** :

2.1.5.3. Fonctionnement

Afin de bien le comprendre le fonctionnement du projet et le rôle de chaque élément, il est nécessaire de retracer les interactions à partir du commencement d'une fonctionnalité : le **clic de l'utilisateur**.

Fonctionnalité : Consultation des données

1



L'utilisateur clique sur l'onglet « **Consultation** ». C'est un lien qui mène vers l'adresse **index.php?action=consultation**

2 dispatcher.php

```php
if (!is_null(myGet('action')))
    $action = myGet('action');
else
    $action = "dataEntry";

require ROOT . DS . 'controller' . DS . 'actions.php';
```

Le dispatcher récupère l'action dans la **barre URL**. La valeur est alors mise dans une variable, puis le contrôleur est appelé.

3 actions.php

```php
DataBaseModel::connect();

switch($action)
{
```

Le contrôleur établit la connexion à la base de données puis segmente ses actions en fonction de la variable récupérée par le dispatcher. Il va alors chercher « **consultation** »

4 actions.php

```php
case 'consultation':

    require ROOT . DS . 'controller' . DS . 'dataEntryRequirements.php';
    require ROOT . DS . 'controller' . DS . 'relationEditionRequirements.php';

    //All entities are selected and counted
    $informations = Select::descriptor('information', 'uniqueTag');
    $informationsCount = Select::dataCount('information');

    $authors = Select::descriptor('ref_author', 'fullName');
    $authorsCount = Select::dataCount('ref_author');

    $dimensions = Select::descriptor('ref_dimension', 'dimensionName');
    $dimensionsCount = Select::dataCount('ref_dimension');

    $keywords = Select::keywords(myGet('keywordsOrder'));
    $keywordsCount = Select::dataCount('ref_keyword');


    $mediums = Select::descriptor('ref_medium', 'mediumName');
    $mediumsCount = Select::dataCount('ref_medium');

    $metakeywords = Select::descriptor('ref_metakeyword', 'metaKeywordName');
    $metakeywordsCount = Select::dataCount('ref_metakeyword');

    $misc1s = Select::descriptor('ref_misc1', 'misc1Name');
    $misc1sCount = Select::dataCount('ref_misc1');

    $misc2s = Select::descriptor('ref_misc2', 'misc2Name');
    $misc2sCount = Select::dataCount('ref_misc2');

    $sources = Select::descriptor('ref_source', 'header');
    $sourcesCount = Select::dataCount('ref_source');

    $source = 'insert_consultation';

    $pageTitle = 'Consultation';
    $view = 'consultation';

    break;
```

Le contrôleur, dans le cas de la consultation, récupère toutes les entités de la base, puis indique la **vue à charger**. Ici, la vue consultation.php utilise donc les données récupérées.

2.1.5.4. Configurations

Dans l'optique de conserver les paramètres de l'application entre chaque instance, le masque utilise un **fichier XML**. Celui est situé dans le dossier « **config** » et stocke les configurations de la base de données et des options personnalisables de l'application.

Son contenu est le suivant :

XML

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Settings>
    <DatabaseConnexion>
        <hostname>localhost</hostname>
        <database>stagedut</database>
        <username>root</username>
        <password></password>
    </DatabaseConnexion>
    <General>
        <proposition>
            <prefix>INF</prefix>
            <mode>User-increment</mode>
        </proposition>
        <misc>
            <misc1/>
            <misc2/>
        </misc>
        <ci/>
        <theme>Dark</theme>
    </General>
</Settings>
```

1 « **config.php** » récupère les valeurs de connexion.

2 L'élément « **proposition** » définit les paramètres de génération automatique d'identifiant. Il existe deux modes « Auto-increment » indiquant d'automatiser le numéro d'identifiant, et « User-increment » indiquant l'inverse.

3 Les nœuds « **misc1** » et « **misc2** » permettent de définir leur état dans l'application « activé » (*hidden*) ou « désactivé » (*vide*).

4 Paramètre supplémentaires, « **ci** » permet de spécifier un Centre d'Informations lié au masque, « **theme** » permet de définir le thème, Dark ou Light.

2.1.5.5. Outils

Afin de faciliter certains éléments de l'application, des outils de développement annexes sont utilisés. Il s'agit principalement d'outils aidant à la création de composants graphiques et à la dynamisation de l'application.

Bootstrap est un **Framework** permettant d'implémenter de manière simple et structurée des éléments graphiques dans une application. Il associe HTML, CSS et Javascript afin de créer des outils puissants et ergonomiques.

Javascript est un langage de programmation de scripts qui permet de dynamiser une page web à l'aide d'interactions simples sur les composants. Animations, déplacements, apparitions et bien d'autres possibilités graphiques sont possibles à l'aide de Javascript.

jQuery est une bibliothèque Javascript créée pour faciliter l'écriture de scripts dans une application. Cette librairie offre la possibilité d'effectuer des fonctionnalités complexes en Javascript en une seule instruction.

Ces outils sont très pratiques et permettent de mettre en place une **application structurée** et respectant les **normes** du web. Ici ils sont utilisés dans les fonctionnalités suivantes :

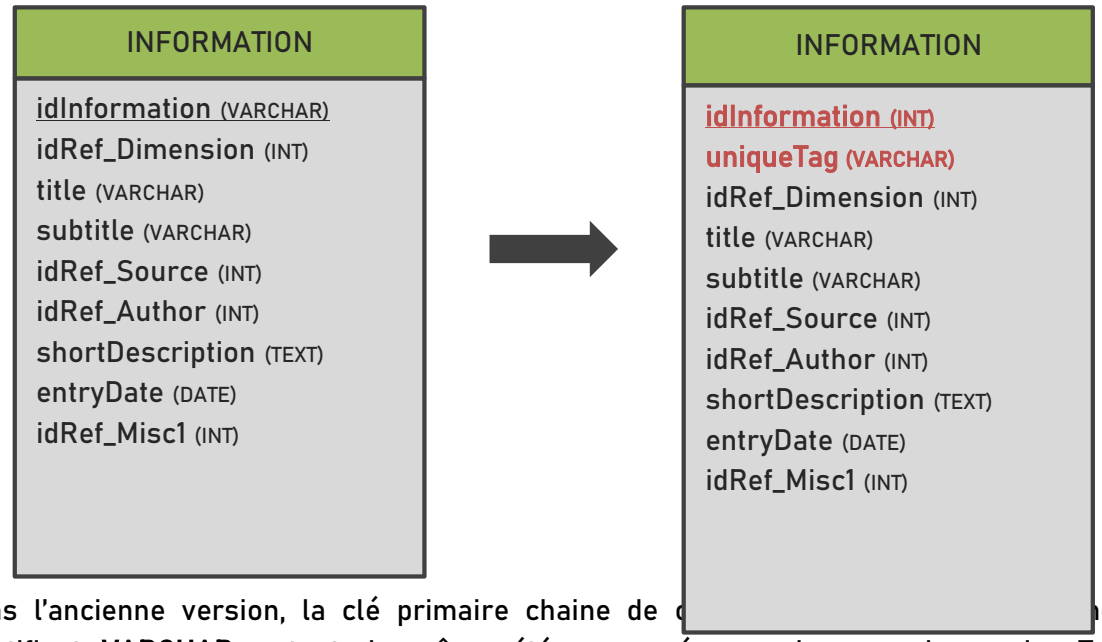
- Apparence et design,
- Fenêtres adaptives (responsive),
- Sélection de date,
- Contrôle des formulaires,
- Fenêtres pop-up,
- Messages bulle,
- Multi-onglets.

2.1.6. Centre d'Informations

Le masque et le Centre d'Informations sont très connectés. En effet, le masque gère des données que le Centre d'Informations affiche. Ainsi, toute correction ou changement effectué durant le développement du masque doivent alors être corrigées dans le Centre d'Informations. De plus, de nouvelles fonctionnalités ont été implémentées.

2.1.6.1. Base de données

Afin d'améliorer les performances et l'intégrité de la base, certains éléments de sa structure ont été modifiés. Cependant, le changement principal est le suivant :



Dans l'ancienne version, la clé primaire chaine de c nes. Un identifiant **VARCHAR** a tout de même été conservé sous le nom de « uniqueTag » et l'idInformation converti en **INTEGER**. Une telle modification demande pour les bases existantes une réinsertion des données. Pour cela il faut utiliser les procédures SQL stockées développées. Il existe une procédure pour chacune des tables suivantes :

- info_keyword,
- info_medium,
- info_access,
- info_link,
- tempo

2.1.6.2. Nouvelles fonctionnalités

La modification de la base de données ayant impliqué des modifications du Centre d'Informations, certaines fonctionnalités ont de plus été ajoutées.

Le Centre d'Informations est une application web développée en **Java**. Pour gérer l'affichage de ses pages, elle associe les langages **HTML** et **XML** à travers un fichier de présentation **XSL**. Les fonctionnalités ajoutées étant des corrections d'affichage, les modifications du code se trouvent donc dans ces trois éléments. Les nouvelles fonctionnalités ajoutées sont :

DIFFÉRENCIATION DES MOTS-CLÉS
Description : Lors de l'affichage des mots clés d'une information, l'application différencie ceux qui n'appartiennent qu'à cette information, et ceux qui sont reliés à d'autres informations, qui sont alors affichés sous forme de liens.

DIFFÉRENCIATION DES FORMATS DE SUPPORTS

Ces modifications sont présentes dans les fichiers Java suivants :

- **C_XMLDocumentFactory** : générateur du XML,
- **C_DescriptorSimple** : définition d'un descripteur simple,
- **C_DataFactory** : gestion des données,
- Les fichiers **XSL**.

2.2. Résultat

Dans cette partie, le résultat du projet sera comparé aux objectifs initiaux et au cahier des charges. La version finale étant celle développée en PHP, la première version du masque réalisée sous WaveMaker ne sera affichée qu'en annexe.

Tout d'abord, voici l'apparence générale du nouveau masque de saisie ainsi que la fonctionnalité principale :

Insérer une information

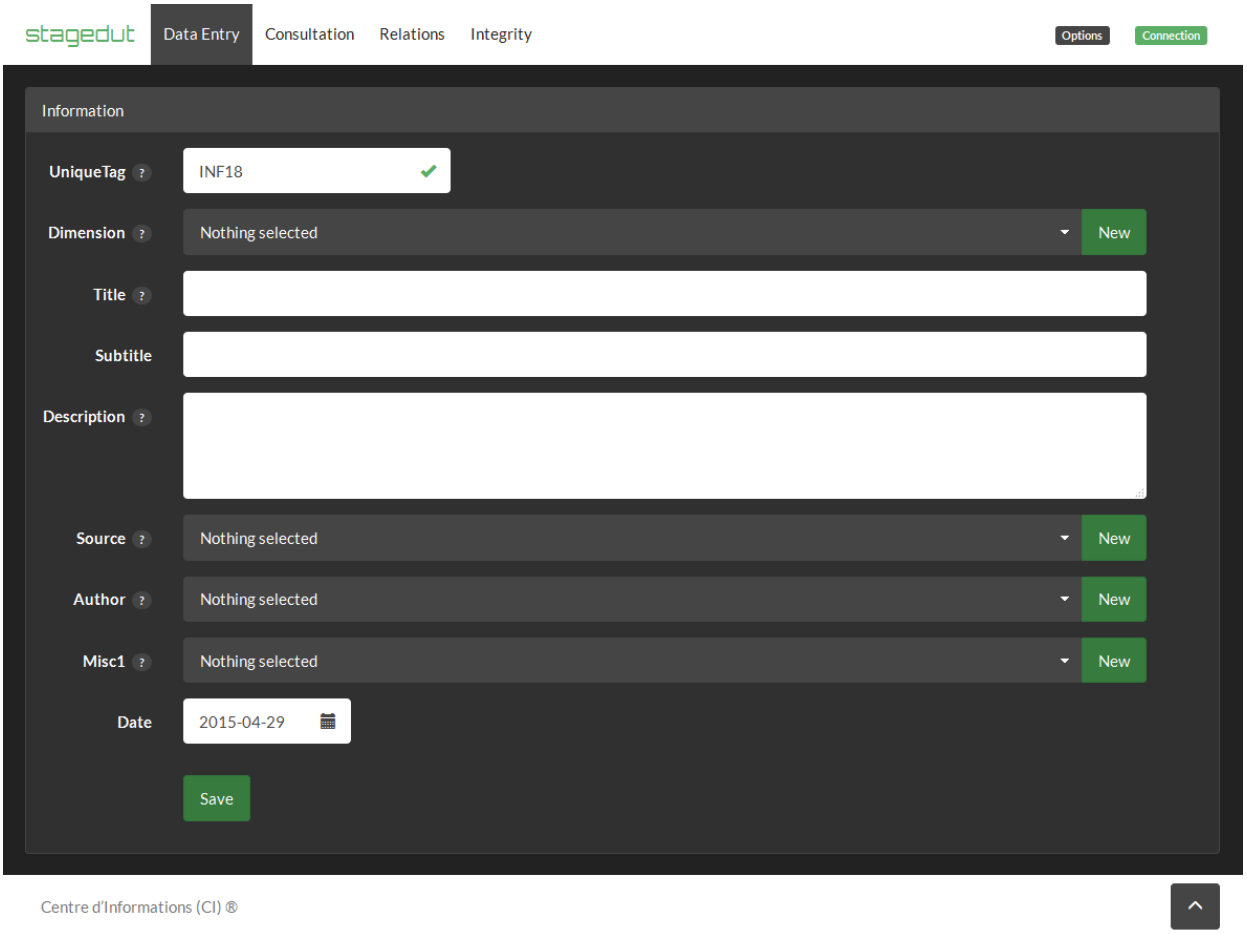


Figure 17 - Nouveau masque : insertion d'informations

Sur cette page d'accueil, la fonctionnalité présentée par défaut est l'insertion d'une information, soit « Data Entry ». Élément très important du cahier des charges, cette interface respecte les contraintes fixées :

- Un identifiant est généré automatiquement (ici « **INF18** »),
- La vérification d'unicité est effectuée (✔),
- Les champs étrangers sont listés en auto-complétion,
- Les champs obligatoires sont respectés.

De plus, les contraintes ergonomiques fixées pour l'ensemble de l'application ont été appliquées :

- Sobriété,
- Simplicité,
- FeedBack (vérification unicité, ✖ ou ✔),
- Textes d'aides (?),
- Couleurs : thème clair et intuitif.

Insérer un descripteur

L'insertion d'une information étant fonctionnelle, étudions la création des **descripteurs**, simples et composites :

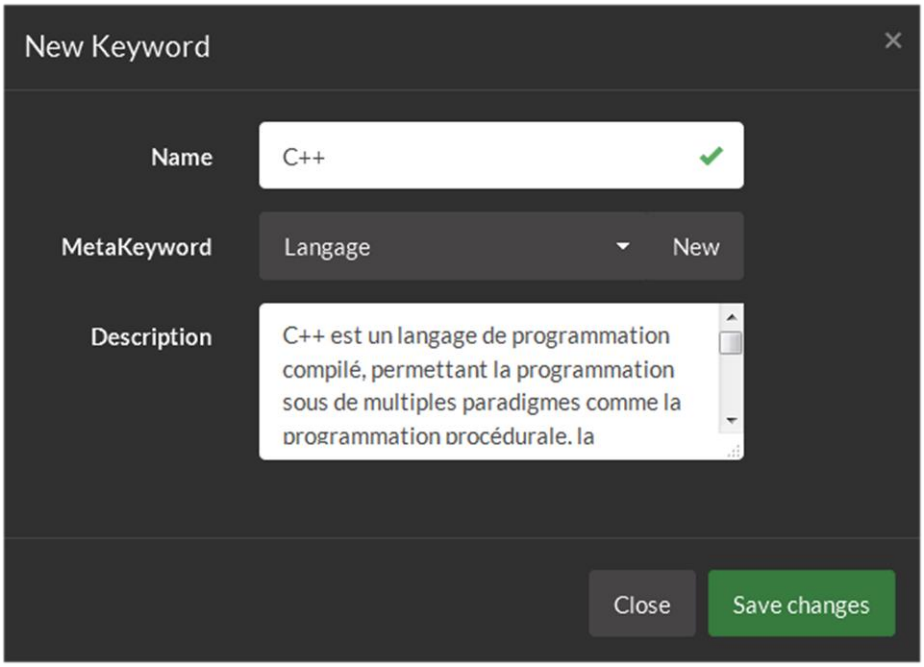


Figure 18 - Nouveau masque : Nouveau descripteur

La création d'un descripteur est semblable à celle d'une information :

- Contrôle d'unicité,
- Champs étrangers listés en auto-complétion,
- Champs obligatoires respectés.

Le système utilisé pour le « **Keyword** » est identique pour chaque descripteur. Ils apparaissent sous la forme d'une fenêtre pop-up modale. Ainsi, il est possible de créer un descripteur durant l'insertion d'une information ainsi que pendant une consultation.

Enfin, le dernier élément de création du masque est l'ajout de relations entre les **descripteurs composites** et les **informations**.

Insérer une relation

Les relations sont gérées à l'aide de tables d'associations. Les vérifications à faire sont donc importantes quant aux unicités et existences des clés étrangères :

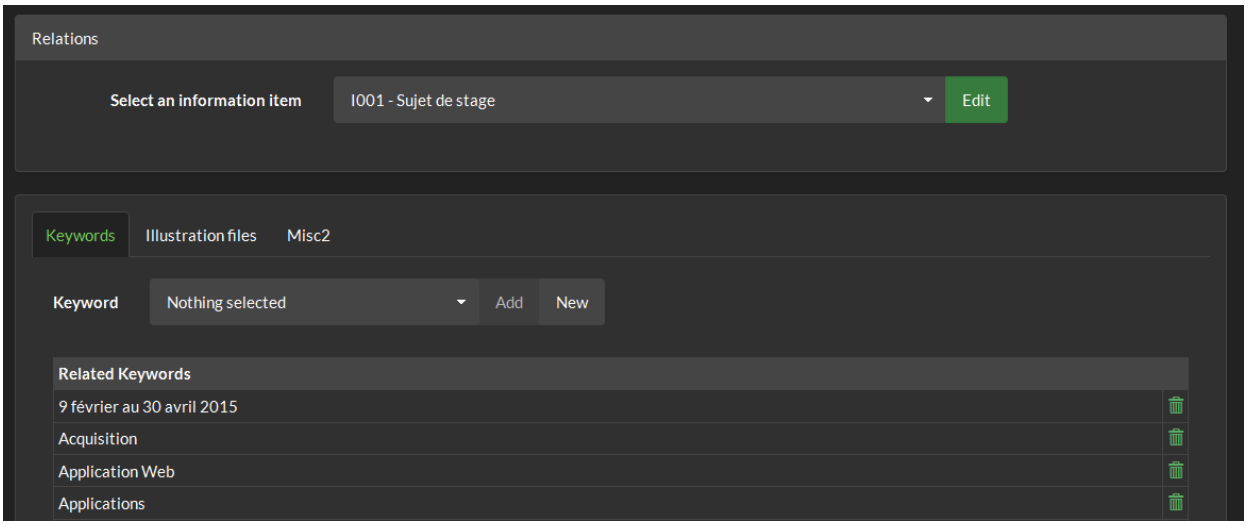


Figure 19 - Nouveau masque : Édition des relations

Les contraintes de cette fonctionnalité ont été respectées :

- Lorsqu'un descripteur composite est sélectionné et ajouté, il disparait de la liste de propositions. Ainsi, il ne peut y avoir de doublons.
- Lorsqu'aucun descripteur n'est sélectionné, il est impossible de faire une insertion,
- Les champs listes sont en auto-complétion.

Grâce à ses contraintes, l'utilisateur ne peut insérer de doublons et de données vides.

Les fonctionnalités d'insertion ont donc été réalisées selon le cahier des charges.

Changer de base de données

Il faut maintenant s'intéresser à l'une des fonctionnalités les plus importantes de ce masque, celle qui a déclenché le basculement de **WaveMaker** vers **PHP** : le changement de base de données en temps réel :

Figure 20 - Nouveau masque : Changement de base de données

Ici, la fonctionnalité est simple et ne requiert peu de contraintes :

- La base de données doit exister,
- Les identifiants doivent être corrects.

Ainsi à tout moment de l'utilisation, il est possible de modifier la base cible.

Contrôle d'intégrité

Le contrôle d'intégrité doit être clair et permettre à l'utilisateur de percevoir rapidement les éléments non-connectés et lui offrir la possibilité de suppression.

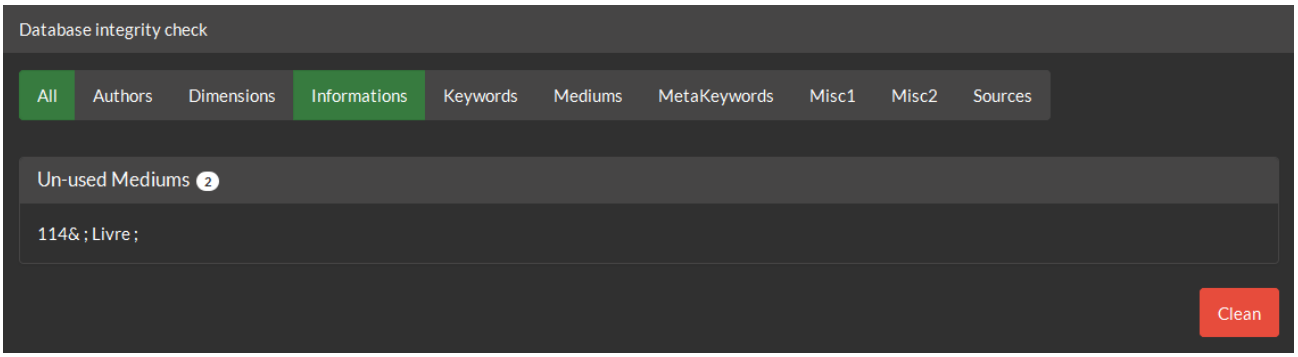


Figure 21 - Nouveau masque : Contrôle d'intégrité

Ici il est donc possible de faire le contrôle pour chaque entité ou bien pour l'ensemble de la base. **Le contrôle est donc conforme aux demandes**.

Consultation

La fonctionnalité de consultation doit permettre une rapide visualisation des données, les modifications des éléments et enfin leur suppression.

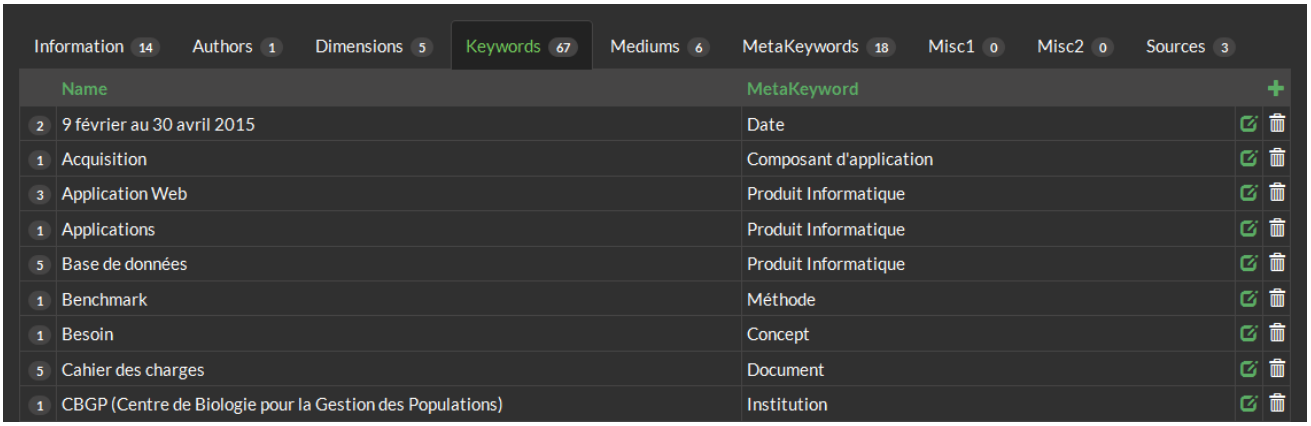


Figure 22 - Nouveau masque : Consultation

La seule contrainte est la suivante : une entité ne peut être supprimée que si elle ne possède aucun lien vers d'autres entités. Ici, lorsque c'est le cas, l'icône « poubelle » est grisée.

Paramètres

Grâce aux paramètres, l'utilisateur peut modifier l'ensemble des configurations de l'application. Il s'agit donc d'une interface simple et intuitive :

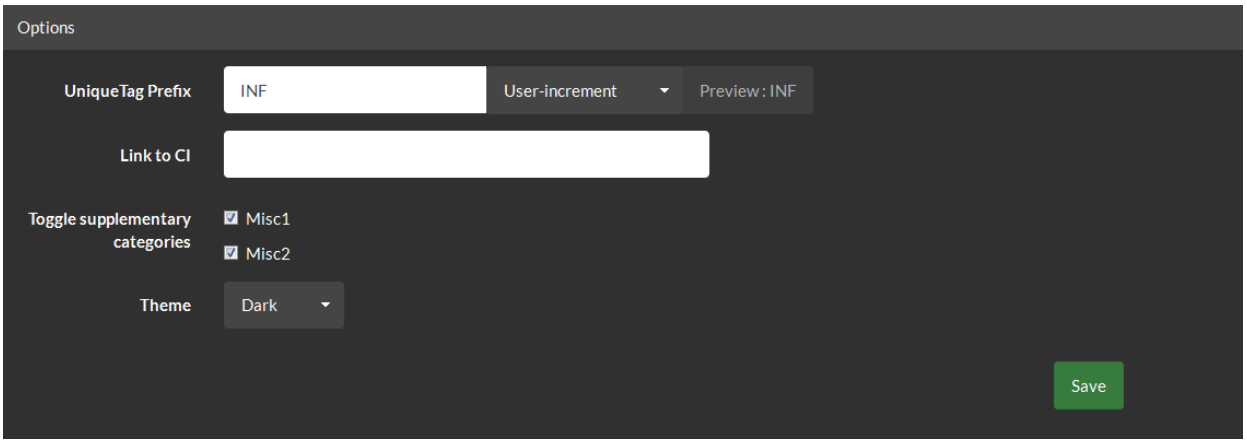


Figure 23 - Nouveau masque : Paramétrage

Parmi les contraintes mises en place dans le cahier des charges, la possibilité de modifier le thème a été rajoutée.

Conclusion

Le résultat de l'application est donc adapté aux demandes du commanditaire. C'est un masque simple et intuitif, capable d'effectuer l'ensemble des fonctionnalités requises. Les contraintes ont été respectées afin de répondre aux besoins mis en jeux dans le cahier des charges.

De plus, certaines fonctionnalités ont été rajoutées durant le développement afin d'enrichir les possibilités du masque. Enfin, il permet un déploiement totalement libre et sans limite, ce qui était l'objectif recherché lors du changement vers le PHP.

2.3. Perspectives

Le masque de saisie est complet. Cependant, il existe des fonctionnalités, découvertes et mises en place durant le développement, qui n'ont pas été implémentées. Ces fonctionnalités représentent les futures possibilités de développement.

EXPORTATION DE LA BASE

Description : Afin d'effectuer des sauvegardes, l'utilisateur peut télécharger au format SQL le contenu de la base sur laquelle est connectée le masque.
Contrainte(s) :

- Fichier .sql,
- Téléchargement automatique,
- Backup de la base de données.

SÉCURISATION

Description : Lorsqu'un même masque est utilisé par plusieurs utilisateurs sur un réseau, l'utilisateur utilise la fonctionnalité de sécurisation pour basculer la base vers un mode protégé sans craintes pour les données.
Contrainte(s) :

- Création d'une base spéciale.

LISTAGE DES FICHIERS

Description : Lorsque l'utilisateur sélectionne un support ou une icône, il peut lister les fichiers présents dans les répertoires correspondants.
Contrainte(s) :

- Répertoires fixes, non modifiables.

L'application est donc encore modifiable et ajustable à tout besoin. Sa structure permet aux développeurs d'insérer ultérieurement des fonctionnalités supplémentaires. Les trois fonctionnalités listées sont des fortes éventualités et permettraient de compléter la vision globale du masque.

3. MANUEL D'UTILISATION

3.1. Installation

3.1.1. Local

1 Afin d'installer le masque sur un ordinateur en local, il faut tout d'abord télécharger et installer le logiciel **WampServer** :

WampServer est une plateforme permettant de faire fonctionner localement des scripts **PHP**.

Téléchargement

Une fois **WampServer** téléchargé, installez-le en vous laissant guider. Vous pouvez choisir un répertoire d'installation personnel ou bien garder « **C:\wamp** ».

2 Si cela n'est pas déjà fait, récupérez l'archive du masque. Décompressez-la afin d'obtenir le dossier source de l'application.

3 Copiez ce dossier et placez-vous dans le répertoire d'installation choisi précédemment (« **C:\wamp** » de base). Rendez-vous alors dans le dossier « www » et copiez le dossier du masque.

4 Pour la première utilisation du masque, il est nécessaire de spécifier une base de données. Par la suite, il suffira d'utiliser l'onglet « Connection » pour changer de base. Pour cela, rendez-vous dans le dossier « **config** », et ouvrez le fichier « **settings.xml** ». Vous trouverez alors les éléments suivants :

```
<DatabaseConnexion>
        <hostname>localhost</hostname>
        <database>ci</database>
        <username>root</username>
        <password></password>
</DatabaseConnexion>
```

Remplissez alors ces balises avec les identifiants de votre base de données.

⚠ **Attention** : Il est important d'effectuer ces modifications pour la première utilisation sans quoi l'application ne pourra fonctionner.

5 Le masque est maintenant fonctionnel. Pour l'utiliser, il faut suffit de vous rendre à l'adresse suivante : « **http://localhost/CI_Mask /** » où **CI_Mask** peut être différent si vous avez modifié le dossier source du masque.

⚠ **Pour chaque utilisation,** il est nécessaire que WampServer soit allumé.

3.1.2. Serveur

L'installation du masque sur un serveur est une procédure plus simple car elle ne requiert normalement aucune installation. Cependant, selon les serveurs cette installation peut être **différente**.

1 Récupérez l'archive du masque. Décompressez-la afin d'obtenir le dossier source de l'application.

2 Copiez ce dossier dans votre serveur. Évidemment, ce serveur doit **supporter le PHP**, et le répertoire contenant le masque est propre à votre serveur.

3 De manière identique au déploiement local, rendez-vous dans le dossier « **config** » du masque, et ouvrez le fichier « **settings.xml** ». Vous trouverez alors les données suivantes à modifier (seulement pour la première utilisation) :

```
<DatabaseConnexion>
        <hostname>localhost</hostname>
        <database>ci</database>
        <username>root</username>
        <password></password>
</DatabaseConnexion>
```

⚠ Si lors de l'utilisation du masque, les fonctionnalités de **paramétrage** et de **connexion** ne fonctionnent pas, cela signifie que le serveur n'a pas les **droits** de modifier l'application. Il faut donc attribuer au dossier du masque les **droits nécessaires** pour ces modifications.

4 l'adresse de votre serveur vers le dossier du masque.

3.2. Utilisation

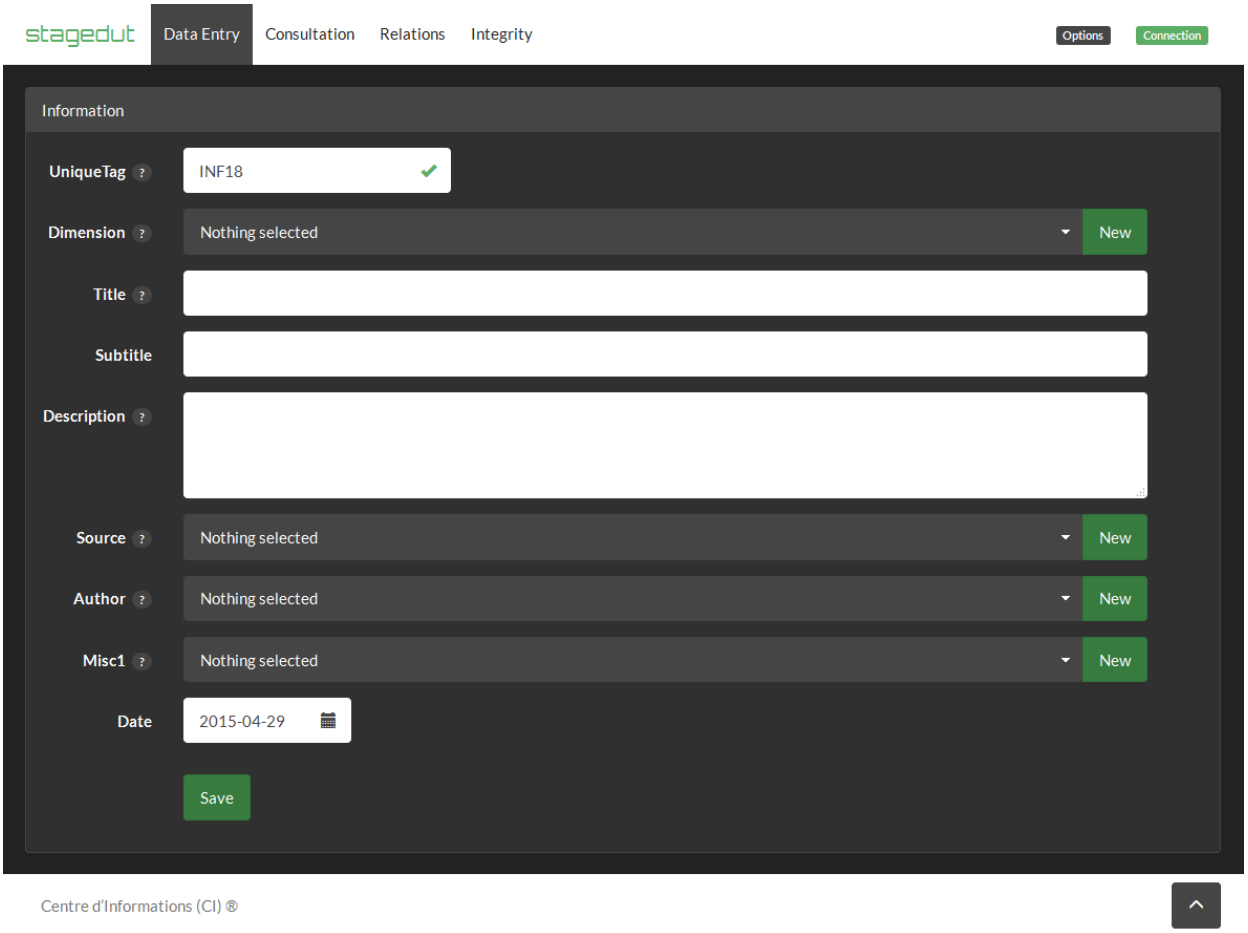
Le masque est organisé en 6 interfaces :

- L'insertion d'**informations** (accueil),
- L'insertion de **relations**,
- La **consultation**,
- Le contrôle d'**intégrité**,

- Les **options**,
- La **connexion**.

3.2.1. Data Entry

L'onglet « Data Entry » est l'onglet par défaut de l'application. Lors du lancement, son interface apparait :



Cette interface est un formulaire, et permet d'insérer une information. Ici chaque champ correspond aux attributs d'une information. Afin d'effectuer une **insertion**, il suffit de remplir au moins les champs « **UniqueTag** » et « **Title** », puis cliquer sur

Save

Afin d'insérer de nouveaux descripteurs durant l'insertion (Auteur, Source, etc..), il suffit de cliquer sur les boutons correspondants :

New

3.2.2. Relations

Une fois le bouton

Save

cliqué, l'application se redirige vers la page « Relations ». Cette page a pour objectif, pour une information sélectionnée, d'**ajouter** ou **supprimer** des relations. Après une insertion depuis « **Data Entry** », l'information est donc automatiquement sélectionnée :

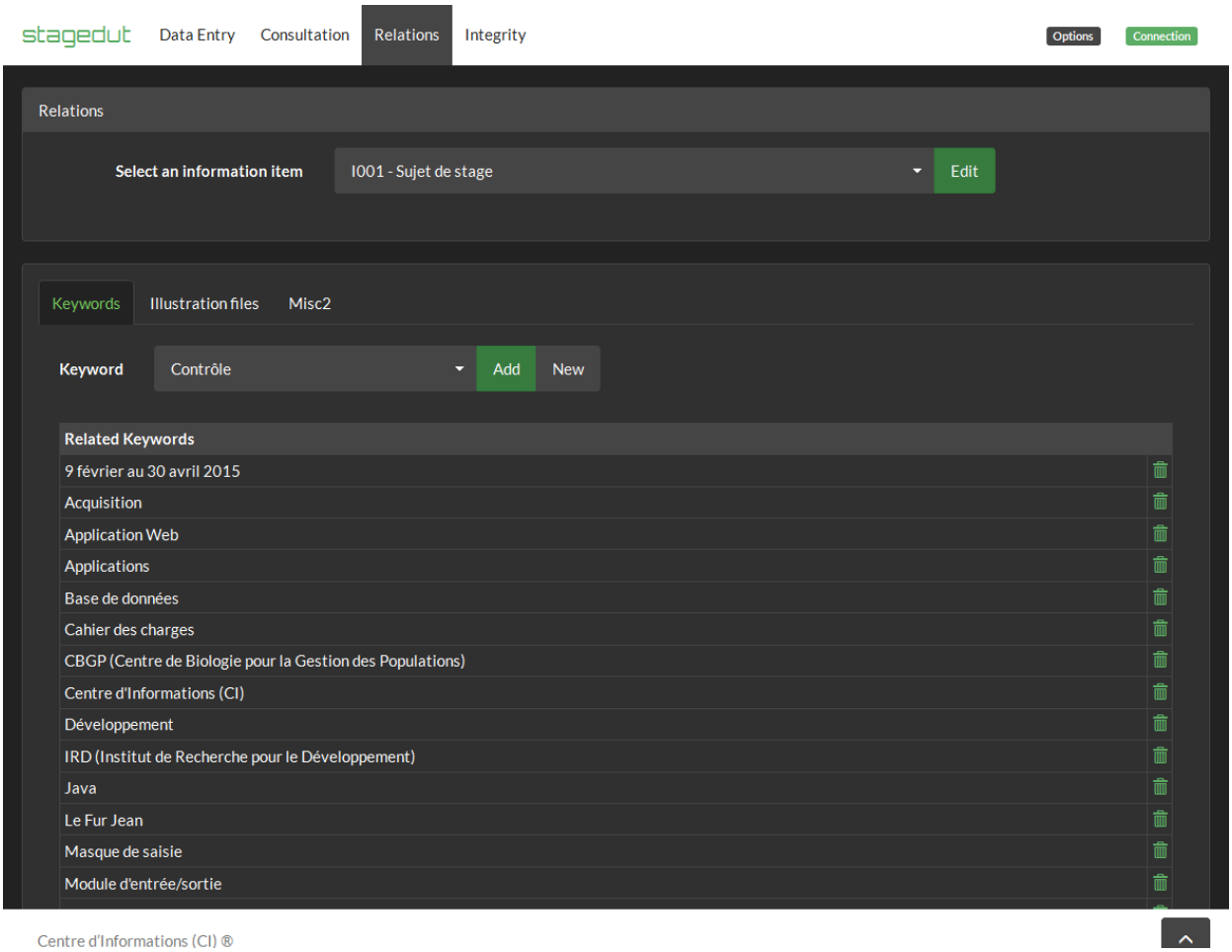


Figure 25 - Relations

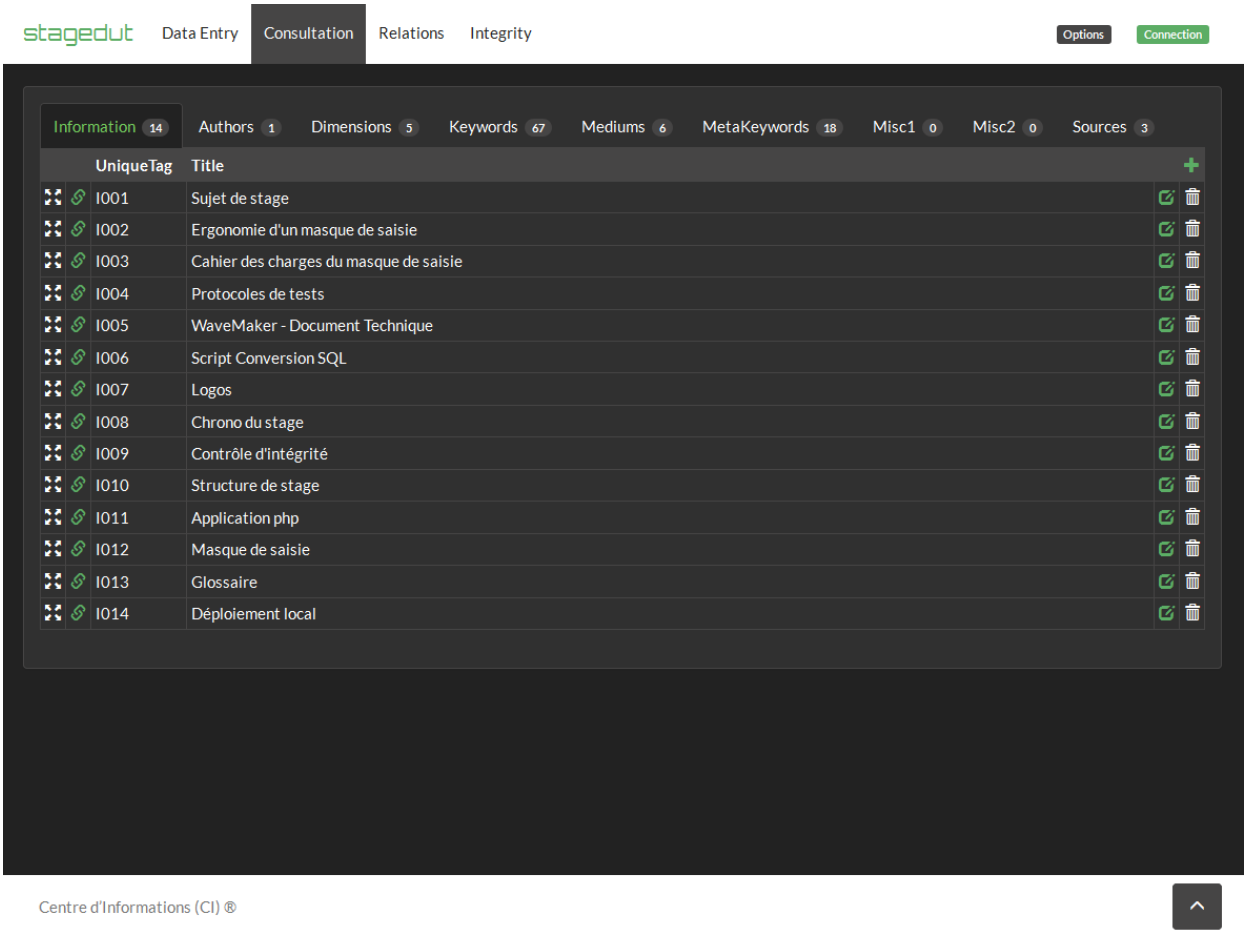
Sur cette interface, il est possible d'ajouter des relations de mots-clés, de supports et de Misc2 (attribut optionnel). Afin d'**ajouter une relation**, il suffit de sélectionner un descripteur (ici «Contrôle »), puis de cliquer sur le bouton **Add**

A l'inverse, pour supprimer une relation, cliquez sur

⚠ **Attention** : L'action de suppression supprime **la relation**, et en aucun cas le mot clé sélectionné.

3.2.3. Consultation

L'interface de consultation permet de visualiser, modifier et supprimer des données. Elle se présente donc comme un affichage basique des données avec des possibilités de modification :



Pour chaque entité, il y a deux actions :

- Modification,
- Suppression.

⚠ Lorsque la suppression d'un élément est impossible car il est connecté avec d'autres entités (contraintes), la poubelle de suppression est grisée.

De plus, pour les « Informations », il est possible d'effectuer les fonctionnalités suivantes :

- Voir l'information dans le Centre d'Informations correspondant (si renseigné),
- Editer les relations de l'information.

3.2.4. Intégrité

L'onglet « **Integrity** » regroupe les fonctionnalités concernant le maintien et l'intégrité de la base. Vu dans les perspectives, la possibilité de télécharger une instance de la base sous forme de **fichier SQL**, si réalisée, apparaitra dans cet onglet.

On y trouve pour l'instant les fonctionnalités suivantes :

- Sécurisation de la base : version préliminaire et non définitive,
- Contrôle d'intégrité.

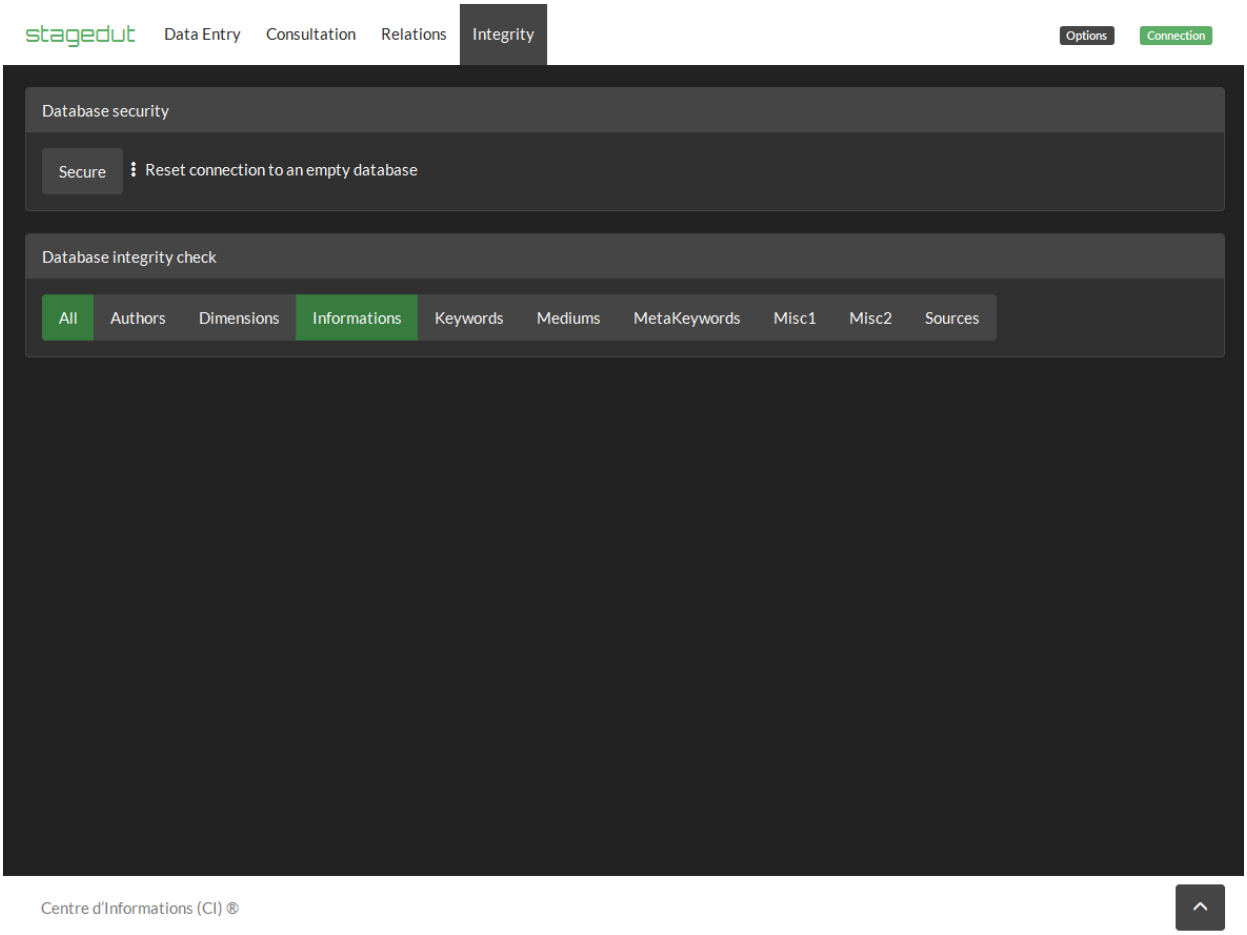


Figure 27 - Integrity

L'utilisation de ces fonctionnalités est simple, il suffit par exemple de cliquer sur **All** afin d'effectuer un contrôle d'intégrité sur l'ensemble de l'application.

Une fois le mode de contrôle choisi, il suffit de cliquer sur **Clean** pour supprimer les entités non intègres.

3.2.5. Paramétrage

Grâce à l'interface « Options », il est possible de configurer plusieurs éléments de l'application, comme la génération d'identifiant, l'affichage d'éléments ou encore le thème.

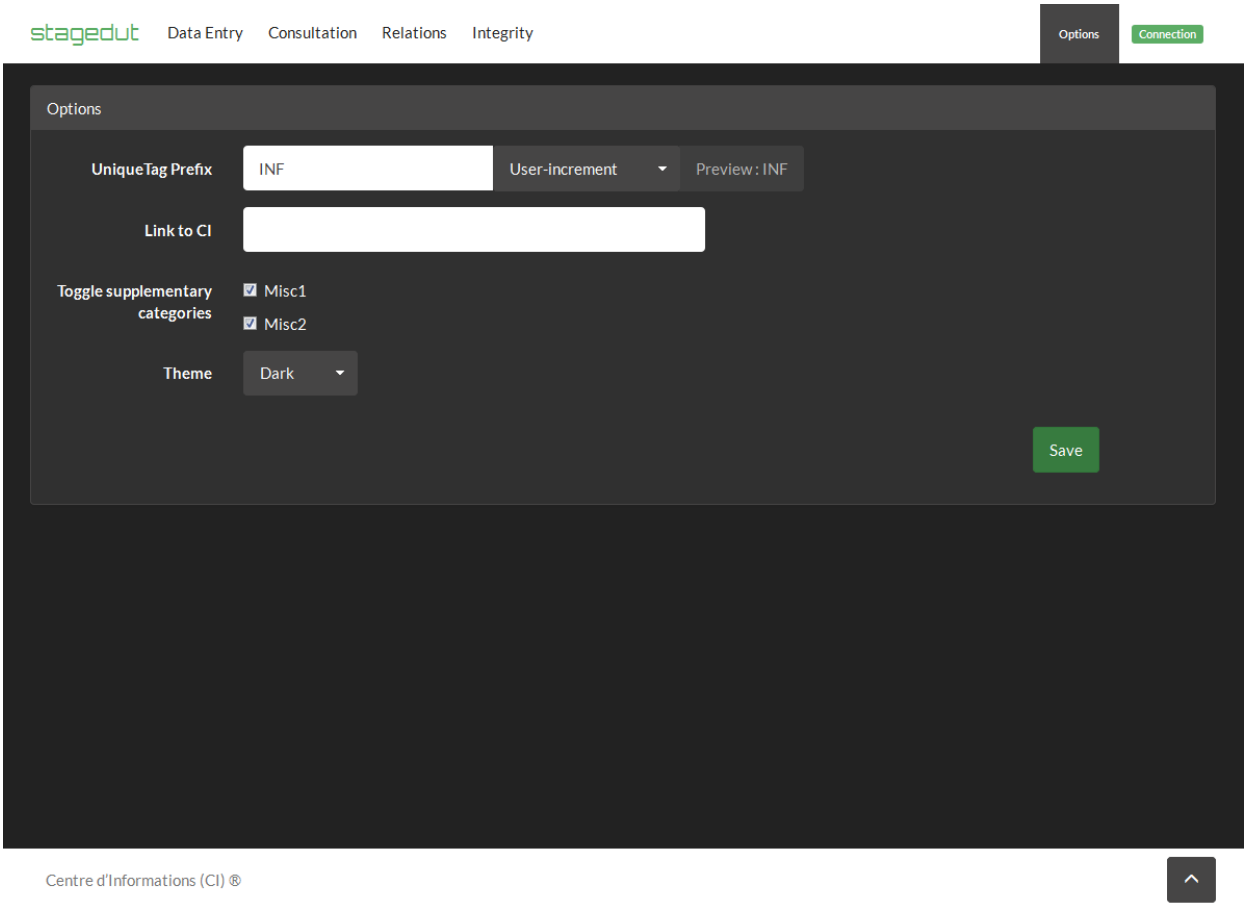


Figure 28 - Options

Ici il est possible de modifier les éléments suivants :

- **UniteTag Prefix** : définition de la génération d'identifiant ;
- **Link to CI** : établir le lien vers un Centre d'Informations,
- **Toggle supplementary categories** : Activer/Désactiver les Misc1 et Misc2,
- **Theme** : sélection du thème.

Afin d'appliquer les modifications, il faut cliquer sur **Save**

⚠ Lorsqu'un lien est établi vers un Centre d'Informations, assurez-vous d'indiquer la racine de l'adresse.

3.2.6. Connexion

La dernière fonctionnalité du masque, et une des plus importantes, est la modification en temps réel de la base de données cible. Pour cela, un formulaire pour les identifiants est nécessaire et se présente ainsi :

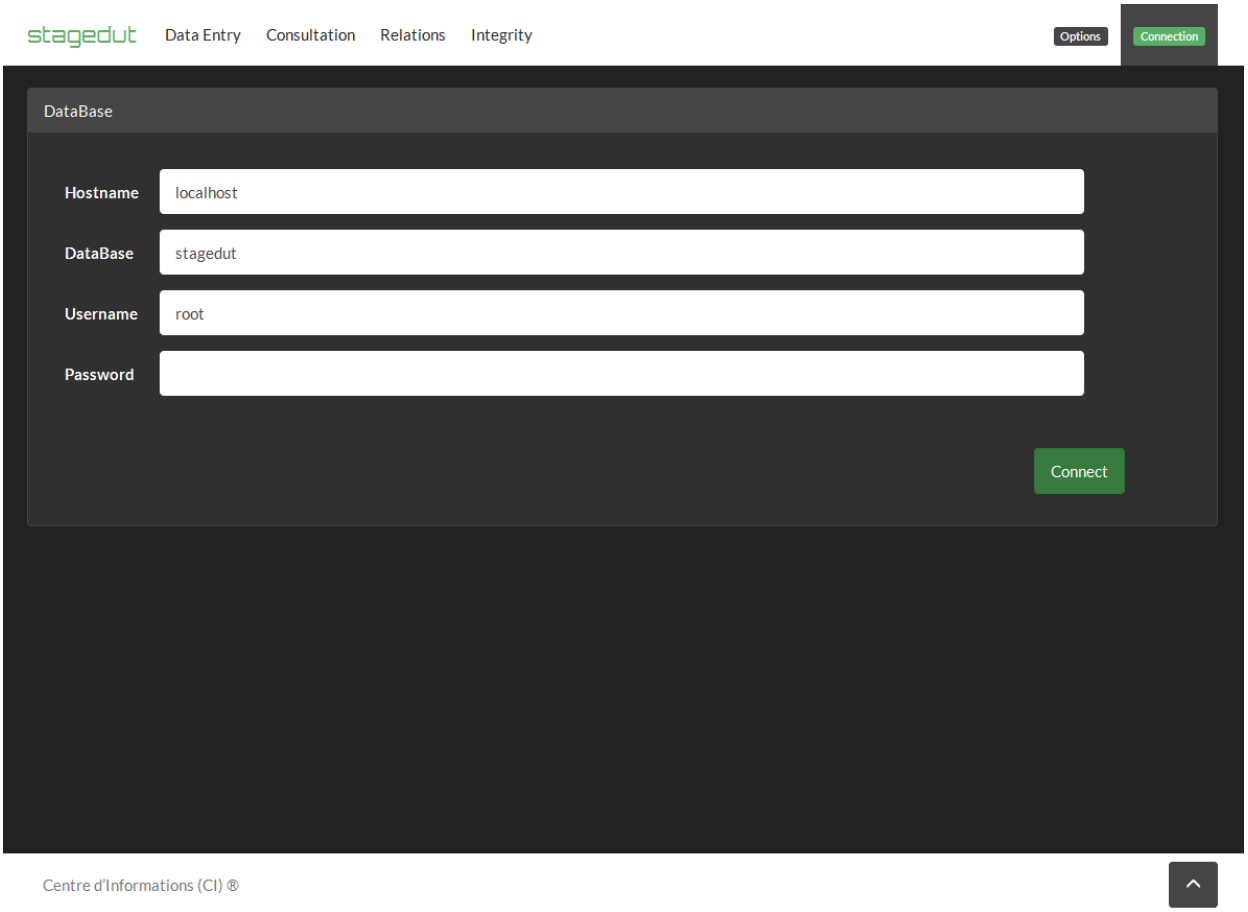


Figure 29 - Connection

Chaque champ représente un identifiant de la base de données, identique à toute connexion à une base de données. Afin d'appliquer la connexion, il suffit de cliquer sur le bouton **Connect**

Si les identifiants sont erronés, un message d'erreur vous indiquera la source du problème : **hostname** invalide, **base de données** invalide ou encore identifiants d'utilisateur erronés.

Une fois la connexion établie, l'application est redirigée vers la **page d'accueil**.

4. RAPPORT D'ACTIVITÉ

Dans cette partie, les méthodes utilisées pour l'**organisation** du travail et du **développement** seront expliquées et détaillées. Durant ce stage, plusieurs méthodes ont été utilisées afin de planifier et synchroniser le travail avec le commanditaire : l'application de planification **Trello**, un fichier sur le déroulement du projet mis à jour **quotidiennement**, une carte mentale sur les **enjeux** et les **thématiques** du stage et enfin des séries de **tests** et de mise en **application** des produits informatiques.

4.1. Carte mentale

Depuis le premier jour du stage, une carte mentale sur les différents éléments du sujet et du projet est tenue à jour. Cette carte permet de visualiser l'ensemble de projet en détail et analyser ce qui a été réalisé, et ce qui reste à réaliser.

Petit à petit, la carte mentale s'est remplie, et à chaque fonctionnalité et avancement du projet, les éléments ont été définis comme « réalisé ». En voici un extrait :

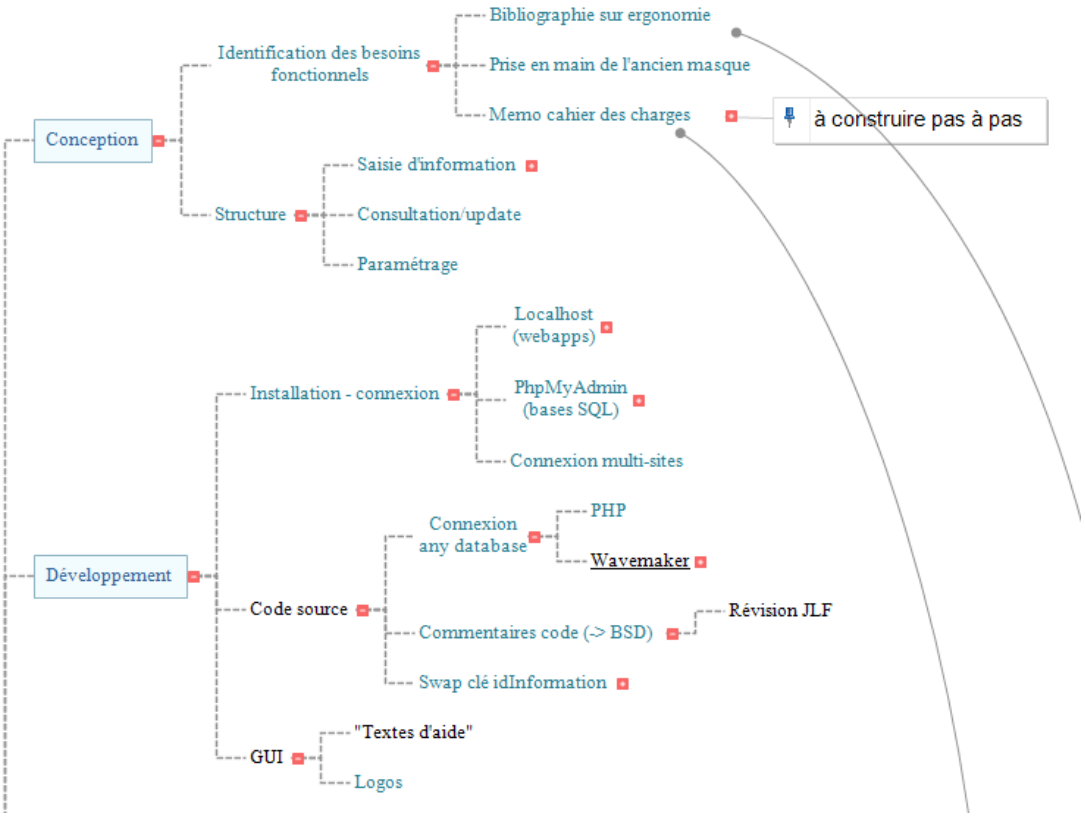


Figure 30 - Carte mentale

4.2. Fichier « chrono »

Afin de garder une trace sur les évolutions du développement et montrer en temps réel les avancées du projet au commanditaire, j'ai tenu tout au long du stage un fichier que l'on appelle « **chrono** ».

Ce fichier consiste à mettre au point ce qui a été fait dans la journée, les nouvelles fonctionnalités, les problèmes rencontrés, les résultats, etc…

Par exemple, voici un extrait du « **chrono** » pour la journée du **16 février 2015** :

16/02/2015

- Version 1a du Mémo 4 – Protocoles de test :
  - Définition des caractéristiques d'information
  - Source 1 : Diagramme cas d'utilisation
  - Source 2 : Diagramme de séquence
- Conception du masque :
  - Ajout de l'onglet Settings
  - Possibilité de modifier le chemin d'accès pour le listage de fichiers
  - Création d'un MetaKeyword avec vérification si champs vides
  - Correction du bouton Reset
  - Correction des onglets
  - Correction élément date
  - Correction sélection Source
  - Création d'une information :
    - Insertion dans la table Information
    - Insertion dans la table InfoKeyword : Gros problème de boucle : WaveMaker est incapable d'enchainer les insertions
      Résolution ? : Blocage dans boucle/Appel sur évènements ou implémentation Java (codée en dur, peu possible pour le déploiement)

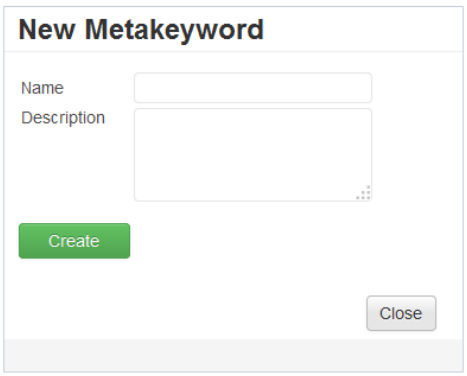


Figure 12 - Création d'un Metakeyword

Figure 31 - Fichier "chrono"

Ce document a probablement été le plus important de notre stage technique, de support pour les recherches, de prise de notes…etc.
Il est important de noter que tous les fichiers du stage ont été stockés sur un **répertoire partagé**, permettant ainsi une excellente communication et partage des données.

4.3. Trello

La carte mentale et le fichier « chrono » sont des outils très pratiques et nous ont permis de mettre en place une organisation partagée. Cependant, en tant que développeur de l'application, j'avais besoin d'un outil pour organiser de manière claire les fonctionnalités à faire, en cours de développement ou encore terminée.

Pour cela, j'ai choisi d'utiliser l'outil **Trello** :

Trello est un outil de gestion de projet en ligne, basé sur une organisation des projets en plusieurs tâches ou « **stories** ». Il est possible d'assigner un « **label** » à la tâche afin de la catégoriser selon un type (Difficulté, Priorité, Valeur..) ainsi que des tests qui permettent de définir si la tâche est réalisée ou non. Voici un exemple de mon projet **Trello** durant le stage :

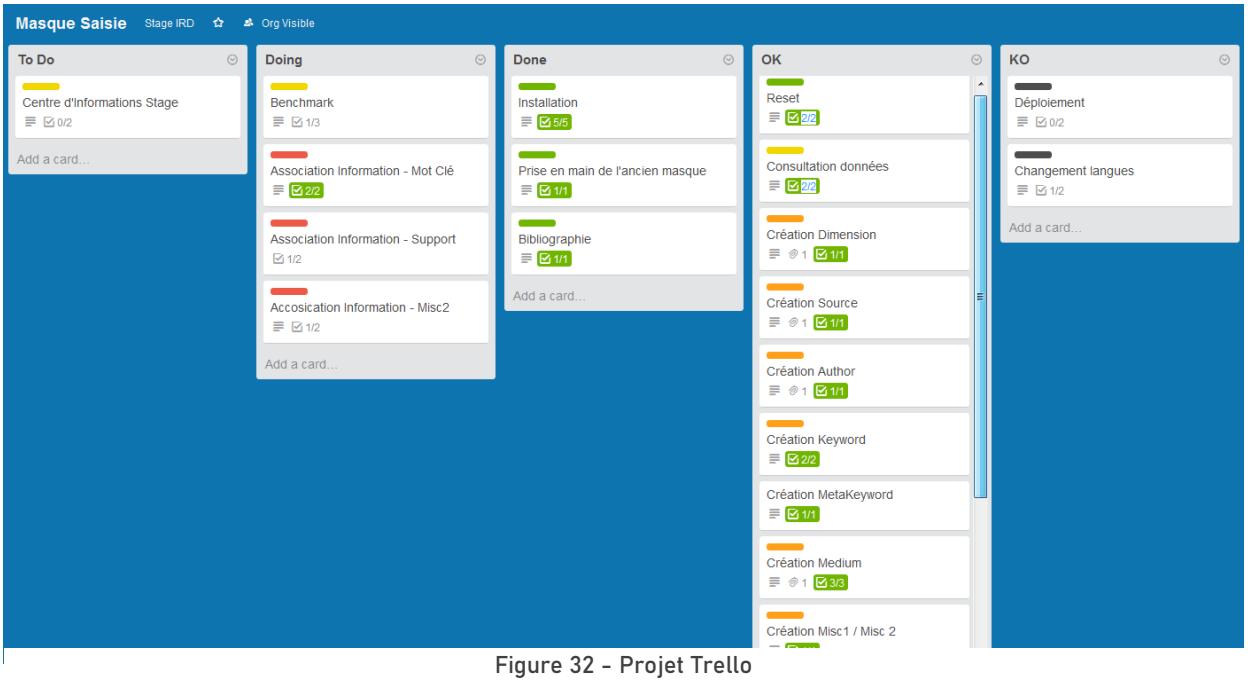


Figure 32 - Projet Trello

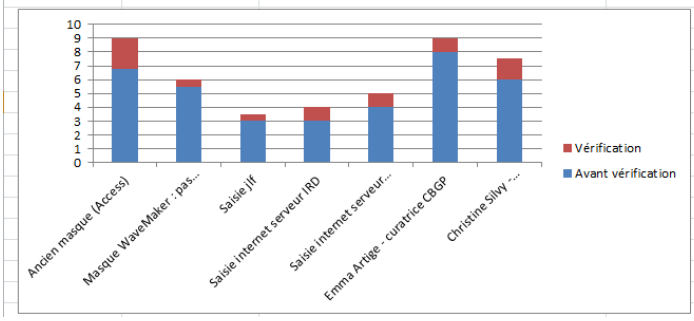
A l'aide de cette interface, j'ai pu organiser de manière claire mes étapes de développements et planifier de façon optimale.

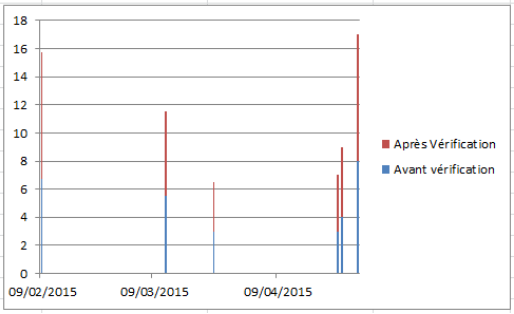
4.4. Phases de tests

Durant le stage, afin de **valider** certaines étapes de développement et porter un regard **critique** sur l'application, des tests ont été effectués. Ces tests ont été très importants pour l'avancée du stage. En effet, en mettant le masque en application, il a été possible d'identifier les points positifs et négatifs, afin de les corriger.

Ces tests ont été effectués à travers des **benchmark** : le but était d'insérer une information dans la base à l'aide du masque, le tout **chronométré** et **vérifié**. Les tests ont été effectués par plusieurs personnes du CBGP, afin d'évaluer la capacité ergonomique et le temps moyen d'insertion d'information. Ces données ont été saisies dans un fichier Microsoft Excel afin de les analyser :

Date	Information	Description	Avant vérification	Vérification	Après Vérification	Préparation
09/02/2015	bandia-018	Ancien masque (Access)	6,75	2,25	9	
12/03/2015	bandia-018	Masque WaveMaker : pas de New dans les mediums	5,5	0,5	6	
24/03/2015	bandia-018	Saisie jlf	3	0,5	3,5	
24/04/2015	bandia-018	Saisie internet serveur IRD	3	1	4	
25/04/2015	bandia-018	Saisie internet serveur IRD,mask 3.2, JLF	4	1	5	
29/04/2015	bandia-018	Emma Artige - curatrice CBGP	8	1	9	
30/04/2015	bandia-018	Christine Silvy - Documentaliste CBGP	6	1,5	7,5	5
19/03/2015	StageDUT-I001	Masque PHP: stoppée avant vérification	10			
25/03/2015	StageDUT-I002	+ correction bug / source	10	6	16	
24/03/2015	StageDUT-I003		6	1	7	
13/04/2015	StageDUT-I012		13	1	14	8
13/04/2015	StageDUT-I013		2			





A l'aide de ces données, il nous a été possible de déterminer une durée moyenne de saisie : **environ 10 à 15 minutes** avec la préparation.

Figure 33 - Analyse de tests

5. CONCLUSION

Le projet « **Opérationnalisation d'un centre d'information scientifique** » avait pour objectif d'offrir au système « Centre d'Informations » une interface de saisie qui agit comme point d'entrée de l'application. **Sobre**, **ergonomique** et **intuitif**, le masque de saisie doit permettre une insertion **rapide** et **fonctionnelle** des informations dans la base de données, un maintien des données ainsi qu'une possibilité de déploiement souple et adaptable à de nombreux profils d'utilisation.

A l'aide du commanditaire et des premières versions obsolètes du masque, il a été possible de déterminer les **contraintes** et les **enjeux** d'une telle implémentation. Malgré un problème technique survenu durant le stage nous obligeant à modifier le mode de développement, les objectifs du **cahier des charges** ont été remplis et réalisé en temps voulu. Nous avons alors implémenté une application capable d'insérer de manière **intuitive** des informations, ainsi qu'un maintien **robuste** et **efficace** des données. Cependant, dans l'optique d'améliorer l'expérience de l'utilisateur et la sécurité de l'application, il est possible d'imaginer une future amélioration du système.

Ce projet m'a permis de mettre en application mes connaissances théoriques acquises à l'IUT ainsi que dans mes développements personnels. Sous la tutelle de **M. Jean Le Fur**, j'ai pu découvrir pour la première fois l'expérience de la vie en entreprise ainsi que son fonctionnement. En effet, travailler en **collaboration** sur un projet destiné à être utilisé par de nombreux utilisateurs représente pour moi un point fort dans cette expérience.

Enfin, l'utilisation des outils de méthode de travail vus à l'IUT ainsi que celles utilisées durant ce stage n'ont fait que **renforcer l'importance** d'une telle organisation de projet. Ces méthodes m'ont permis de travailler de manière **synchronisée** avec le commanditaire offrant ainsi une meilleure **communication**.

Ce projet m'a donc apporté une expérience **indispensable** et **significative** dans mon parcours professionnel, et m'a offert une vision enrichissante du monde professionnel.

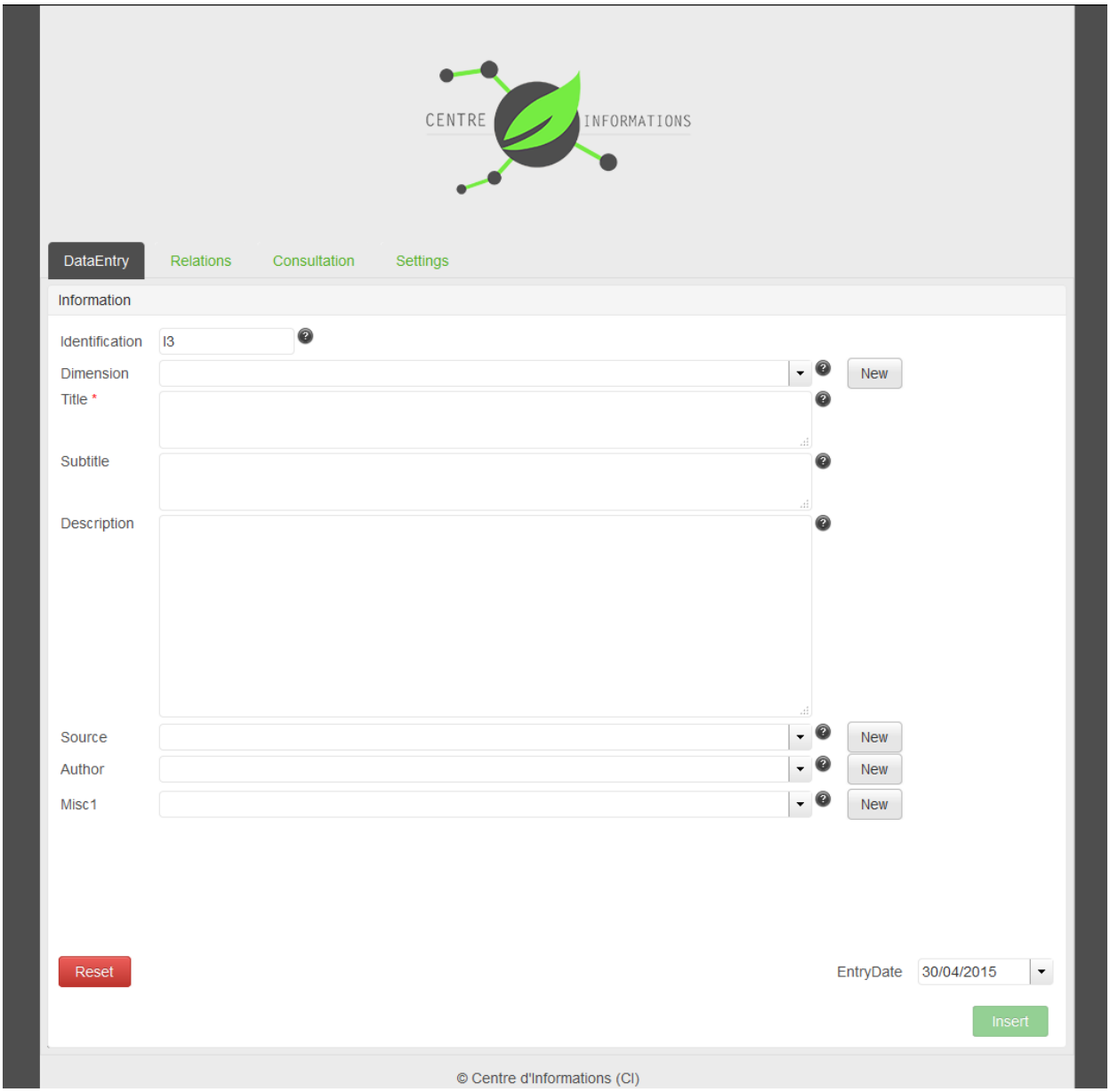
SITOGRAPHIE

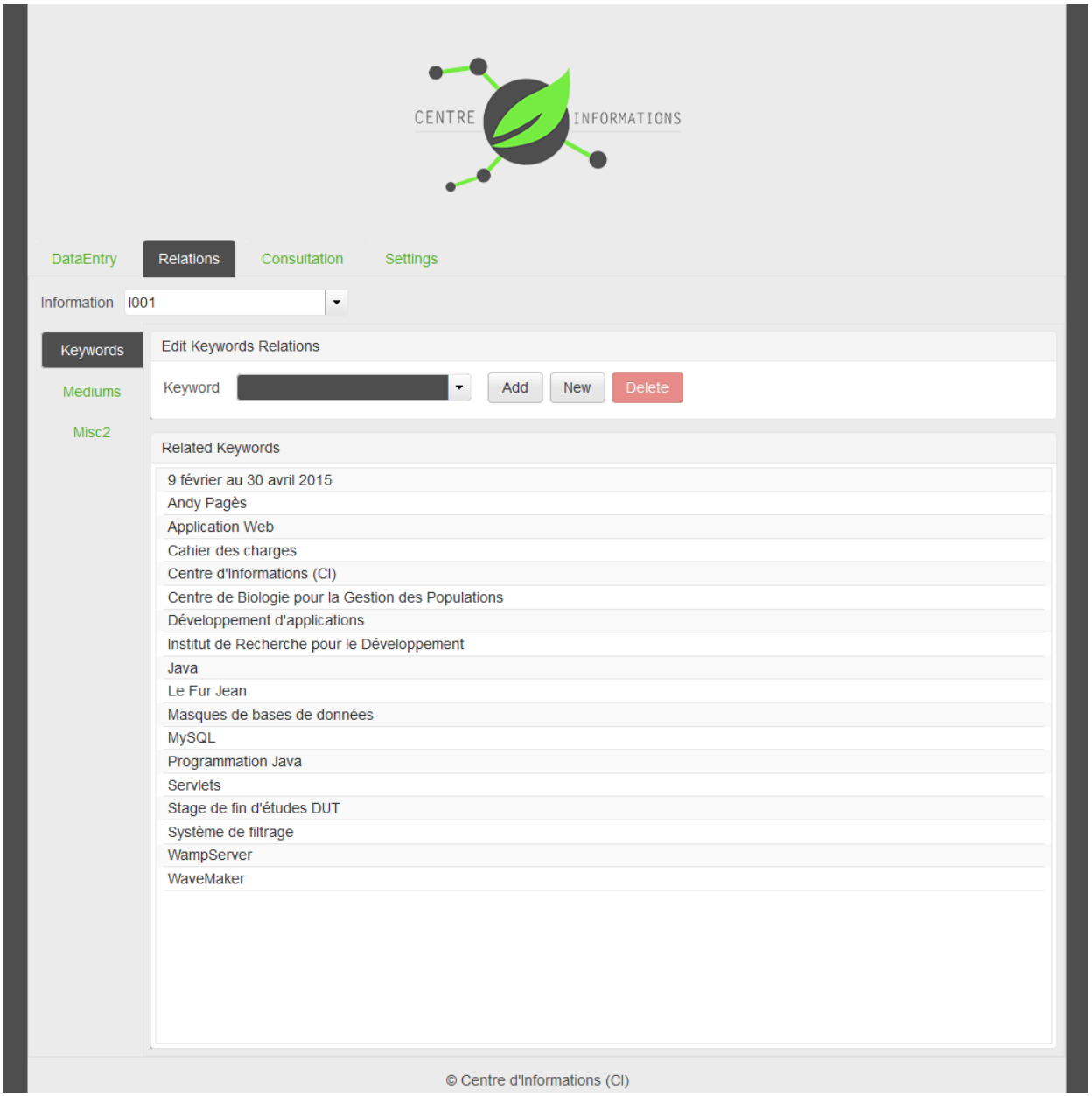
Adresse	Type de site	Sujet
http://getbootstrap.com/	Site officiel	Framework CSS
http://stackoverflow.com/	Site officiel	Dépannage programmation

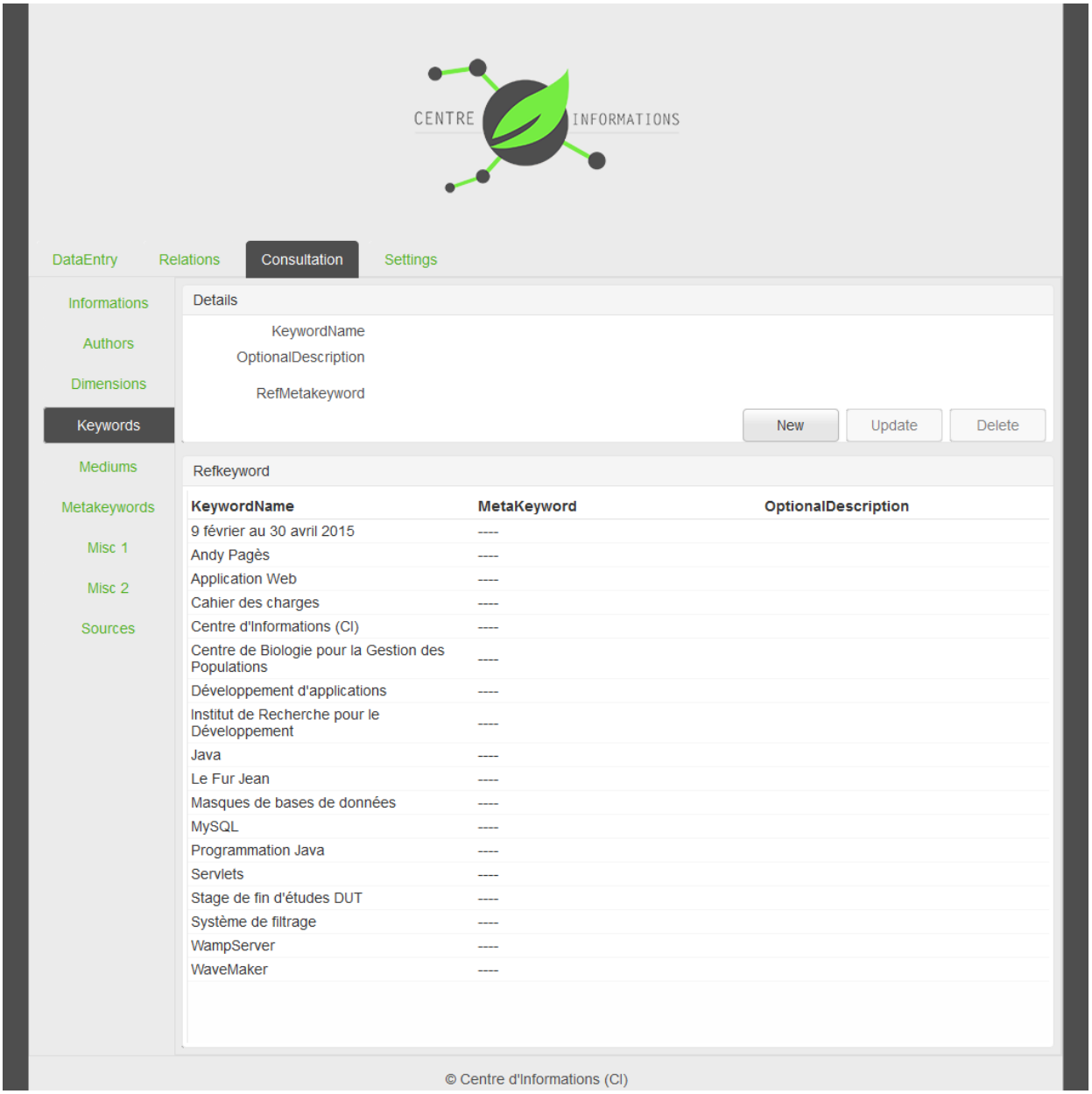
http://dev.wavemaker.com	Site officiel	Forum WaveMaker
https://developer.android.com/design/index.html	Site Officiel	Recherche ergonomique

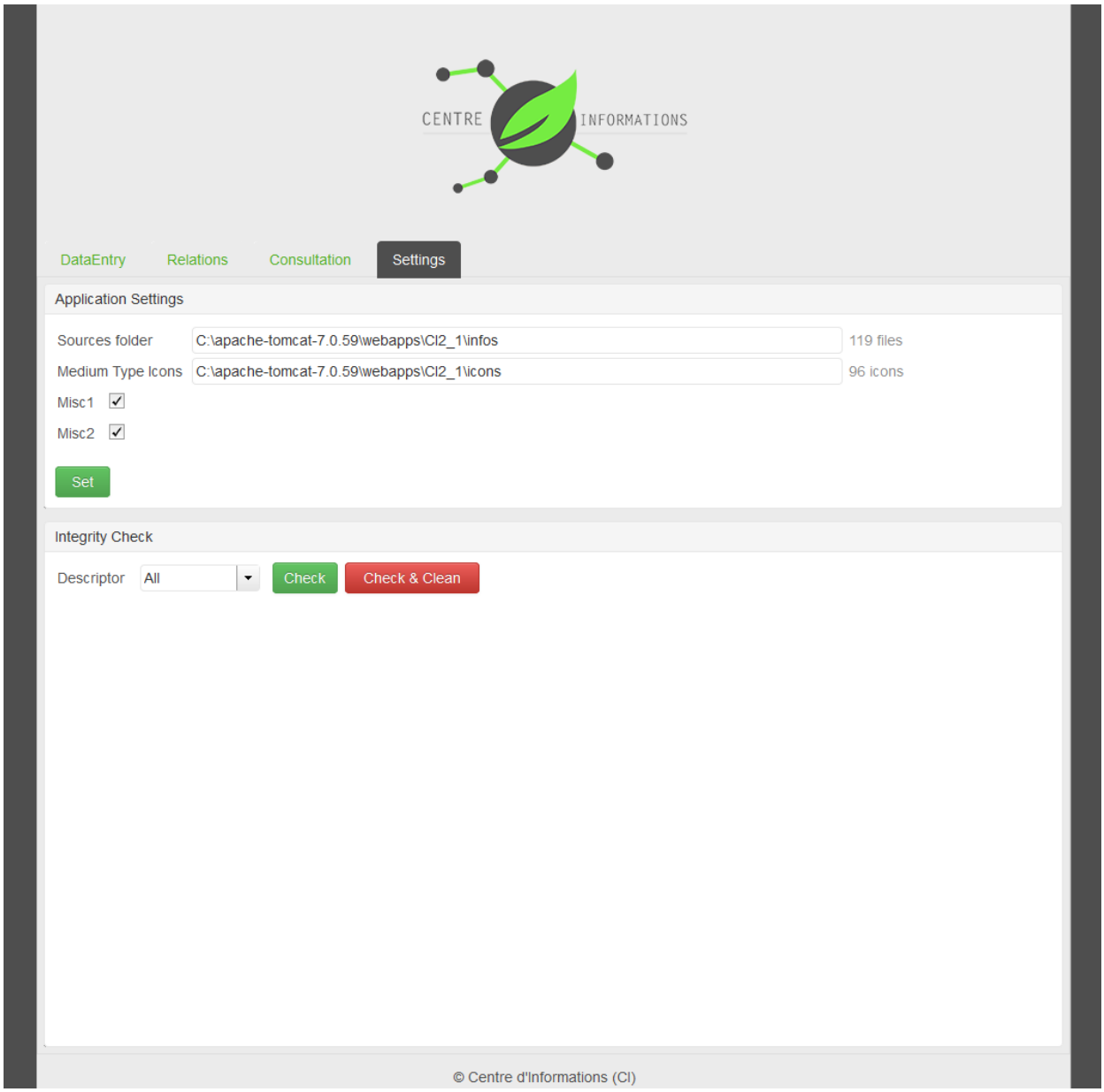
ANNEXES

Annexe 1 : Masque WaveMaker




CENTRE INFORMATIONS
DataEntry
Relations
Consultation
Settings
Information I001
Keywords
Mediums
Misc2
Edit Keywords Relations
Keyword
Add
New
Delete
Related Keywords
9 février au 30 avril 2015
Andy Pagès
Application Web
Cahier des charges
Centre d'Informations (CI)
Centre de Biologie pour la Gestion des Populations
Développement d'applications
Institut de Recherche pour le Développement
Java
Le Fur Jean
Masques de bases de données
MySQL
Programmation Java
Servlets
Stage de fin d'études DUT
Système de filtrage
WampServer
WaveMaker
© Centre d'Informations (CI)


CENTRE INFORMATIONS
DataEntry
Relations
Consultation
Settings
Informations
Authors
Dimensions
Keywords
Mediums
Metakeywords
Misc 1
Misc 2
Sources
Details
KeywordName
OptionalDescription
RefMetakeyword
New
Update
Delete
Refkeyword
KeywordName
MetaKeyword
OptionalDescription
9 février au 30 avril 2015 ----
Andy Pagès ----
Application Web ----
Cahier des charges ----
Centre d'Informations (CI) ----
Centre de Biologie pour la Gestion des Populations ----
Développement d'applications ----
Institut de Recherche pour le Développement ----
Java ----
Le Fur Jean ----
Masques de bases de données ----
MySQL ----
Programmation Java ----
Servlets ----
Stage de fin d'études DUT ----
Système de filtrage ----
WampServer ----
WaveMaker ----
© Centre d'Informations (CI)


CENTRE INFORMATIONS
DataEntry
Relations
Consultation
Settings
Application Settings
Sources folder
C:\apache-tomcat-7.0.59\webapps\CI2_1\infos
119 files
Medium Type Icons
C:\apache-tomcat-7.0.59\webapps\CI2_1\icons
96 icons
Misc1
Misc2
Set
Integrity Check
Descriptor
All
Check
Check & Clean
© Centre d'Informations (CI)

Annexe 2 : Détails ergonomiques

★★★ Simplicité : Phrases courtes et brèves, afin de ne pas surcharger l'attention de l'utilisateur,

★★★ Sobriété : Pas trop d'éléments à la fois, seulement ceux essentiels et nécessaires afin de ne pas embrouiller l'utilisateur,

★★★ Choix et liberté : Choix prédéfinis, mais possibilité de modification, afin de ne pas limiter et frustrer l'utilisateur,

★★★ Repères : Offrir à l'utilisateur des repères de navigation afin qu'il sache toujours dans quelle partie de l'application il se situe,

★★★ Indications claires : Donner des indications claires et essentielles à l'utilisateur, sans entrer dans les détails techniques et inutiles,

★★★ Catégorisation : Évaluer l'importance des différentes tâches et les distinguer par leurs accès, plus ou moins rapide, plus ou moins visible,

★★☆ Images : Les images sont plus rapides que les mots,

★★☆ Navigation cognitive : Utilisation d'illustrations et de représentations pour la navigation, afin de faciliter les efforts de compréhension,

★☆☆ Préférences d'utilisateurs : Sauvegarde des préférences d'utilisation d'utilisateurs, afin d'éviter les répétitions,

★☆☆ Continuité et correspondance : Garder un schéma d'interface identique pour chaque élément afin d'accélérer les repères,

★☆☆ FeedBack : Offrir à l'utilisateur des indications fréquentes (mais légères) à l'aide de confirmations, retours et indices graphiques,

★☆☆ Simplification générale : Automatiser et simplifier les tâches principales et complexes afin d'éviter à l'utilisateur des manipulations et des prises de décision inutiles,

★☆☆ Couleurs : Utiliser une palette de couleur qui offre du contraste et une visibilité agréable.

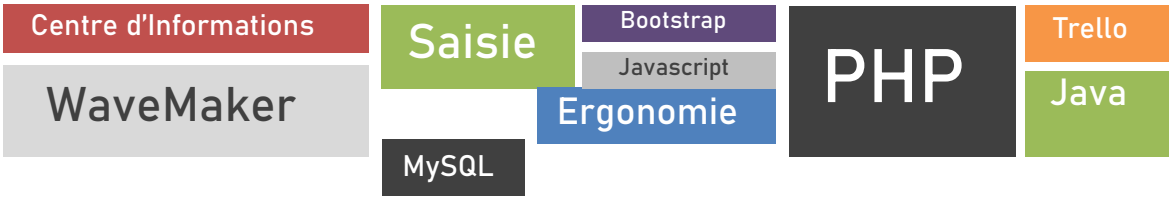
Annexe 3 : Procédures SQL

```
BEGIN

    DECLARE v_c1_idInfo VARCHAR(4);
    DECLARE v_c2_idKey INT;
    DECLARE v_idRecup INTEGER;
    DECLARE done BOOLEAN DEFAULT 0;

    DECLARE c1 CURSOR
    FOR
    SELECT idInformation FROM info_keyword_old ORDER BY idInformation, idRef_Keyword;

    DECLARE c2 CURSOR
    FOR
    SELECT idRef_Keyword FROM `info_keyword_old` ORDER BY idInformation, idRef_Keyword;

    DECLARE CONTINUE HANDLER FOR SQLSTATE '02000' SET done=1;

    OPEN c1;
    OPEN c2;

    FETCH c1 INTO v_c1_idInfo;
    FETCH c2 INTO v_c2_idKey;

    REPEAT
        SELECT idInformation INTO v_idRecup
        FROM information
        WHERE uniqueTag = v_c1_idInfo;

        INSERT INTO `info_keyword` (`idInformation`, `idRef_Keyword`) VALUES (v_idRecup, v_c2_idKey);

        FETCH c1 INTO v_c1_idInfo;
        FETCH c2 INTO v_c2_idKey;
    UNTIL done END REPEAT;

    CLOSE c1;
    CLOSE c2;

END
```

RÉSUMÉ

Le '**Centre d'Informations (CI)**' est une application qui vise à mettre en relation des informations unitaires sur divers domaines de recherche scientifique. L'application élabore à partir d'informations calibrées un ensemble de mots-clés sur lequel est construite une ontologie du domaine. Les trois ensembles **informations - mots-clés - typologies** conduisent à un réseau sémantique qui peut être utilisé pour naviguer dans le domaine de connaissance. Le projet « **Opérationnalisation d'un centre d'information scientifique** » a pour objectif d'offrir aux utilisateurs une application permettant d'insérer facilement des informations dans la base de données. Cette application doit être **ergonomique**, **sobre** et **intuitive**, et doit permettre un maintien robuste des données.

Le système a été réalisé à l'aide des langages **PHP**, **SQL** et **JavaScript**. Il utilise la bibliothèque **jQuery**, le système de gestion de bases de données **MySQL**, et le Framework **Bootstrap**.



SUMMARY

The '**Centre d'Informations (CI)**' is an application that aims to link information items on various areas of scientific research. From calibrated informations, the application develops a set of keywords on which is built a domain ontology. **Informations, keywords, typologies** : all lead to a semantic network that can be used to navigate within the field of knowledge. The project « **Operationalizing a scientific information center** » aims to provide users an application with which they can easily insert informations into the database. This application must be **ergonomic**, **simple** and **intuitive**, and must allow a robust data hold.

The system was realized using **PHP**, **SQL** and **JavaScript**. It also uses the **jQuery** library, the **MySQL** database management system, and **Bootstrap** Framework.