



INSTITUT SUPÉRIEUR D'INFORMATIQUE
DE MODÉLISATION ET DE LEURS APPLICATIONS

Campus des Cézeaux
24 Av. des Landais
BP 10125
63173 AUBIERE CEDEX

RAPPORT D'INTÉGRATION D'APPLICATION
FILIERE GÉNIE LOGICIEL ET SYSTÈMES INFORMATIQUES

VALORISATION D'UNE BASE DE DONNÉES AVEC
GOOGLE APP ENGINE ET CLIENT WEB

Étudiants : Antoine COLMARD et Nicolas PRUGNE

Antoine COLMARD et Nicolas PRUGNE : *Rapport d'intégration d'application*, Valorisation
d'une base de données avec Google App Engine et client web, Décembre 2014

TABLE DES FIGURES

FIGURE 1	Classe Author	5
FIGURE 2	Classe Book	6
FIGURE 3	Vue principale	6
FIGURE 4	Bouton d'ajout	7
FIGURE 5	Ajout d'un auteur	7
FIGURE 6	Auteur modifié	7
FIGURE 7	Ajout d'un livre	8
FIGURE 8	Recherche d'un auteur pour un livre	8
FIGURE 9	Livre modifié	9
FIGURE 10	Recherche sans paramètre	9
FIGURE 11	Recherche avec paramètre	9

TABLE DES MATIÈRES

1	INTRODUCTION	1
2	STRUCTURE DE LA BASE	2
3	SERVEUR ET WEB SERVICES	3
4	CLIENT DE L'APPLICATION	4
4.1	Choix du client	4
4.2	Structure du client	5
4.2.1	Design	5
4.2.2	Modèle	5
4.2.3	Fonctionnement	6
4.3	Communication avec le serveur	9
5	RÉSULTATS ET TESTS	11
6	CONCLUSION	12

INTRODUCTION

À l'ère du *Big Data*, la majorité des applications informatiques recueillent des quantités importantes de données au sein de leurs bases. La problématique de la valorisation de ces données entre alors en compte. Cette valorisation passe dans un premier en la création d'une application serveur pouvant communiquer avec la base, puis en l'exposition des méthodes de cette application pour que des applications client puissent interagir avec la base de données. L'utilisation de *web service* permet de rendre l'application disponible n'importe où et permet une pluralité des applications client pouvant communiquer avec le serveur.

Le travail demandé ici est de mettre en valeur une base de données par la création de web services et d'une application client pouvant communiquer avec ceux-ci.

La création de *web services* à partir de rien peut être une tâche complexe. Pour faciliter ce travail, l'utilisation d'API est conseillée. Ici, nous nous baserons sur les API du *Google App Engine* pour la création des web services. Enfin, nous pourrons déployer notre application dans le *Cloud* de Google.

Dans ce rapport, nous verrons dans un premier temps la structure de la base à valoriser. Dans un deuxième temps, nous verrons l'analyse et la réalisation de l'application serveur et les *web services* exposés. Ensuite, nous expliquerons le choix effectué pour l'application client et expliquerons la structure et le fonctionnement de celle-ci. Enfin, nous verrons les résultats et les tests effectués.

STRUCTURE DE LA BASE

SERVEUR ET WEB SERVICES

CLIENT DE L'APPLICATION

Une fois l'application serveur développée, il nous a fallu choisir le type d'application client à développer. Dans cette partie, nous expliciterons le choix effectué puis nous détaillerons la structure du client et la façon dont celle-ci communique avec le serveur.

4.1 CHOIX DU CLIENT

Pour l'application client, nous désirions que l'application soit légère, portable et compatible avec un maximum de plateformes. Le choix du client web s'est alors montré comme une évidence. En effet, celui-ci ne nécessite aucune installation pour le client, est compatible avec les ordinateurs classiques comme avec les mobiles. Le client web a l'avantage de pouvoir être maintenu plus facilement puisque ses fonctionnalités sont mises à jour pour tous les clients dès qu'il est mis à jour. Le déploiement est également facilité puisqu'il suffira de le déployer sur le cloud Google en même temps que l'application serveur.

Dans le cadre de ce projet, nous avons préféré utiliser un client riche basé sur du HTML5 et du JavaScript. Les interactions avec le serveur sont donc effectuées via des requêtes AJAX. Nous avons préféré cette technique car elle permet de décentraliser la logique de la page ce qui permet d'alléger la charge du serveur et peut donner un plus grand dynamisme à la page. De plus, l'utilisation de cette technique est favorisée par le développement des normes du web et l'abandon de certains navigateurs obsolètes auxquels on préfère désormais des navigateurs avec des mises à jours automatisées. Le développement de ses applications peut donc de plus en plus se baser sur des API JavaScript innovantes.

4.2 STRUCTURE DU CLIENT

Maintenant que le choix du client a été explicité, nous allons analyser plus précisément sa structure. Dans un premier temps, nous aborderons son design. Ensuite, nous verrons le modèle développé en JavaScript pour le stockage et la gestion des entités. Enfin, nous verrons le fonctionnement du client.

4.2.1 DESIGN

Le design et l'ergonomie occupe une grande part de tout projet. Pour celui-ci, nous nous sommes basés sur un le Framework CSS *Metro UI CSS*¹. Celui offre un *flat & responsive design* et de nombreux composants avec une syntaxe relativement légère. Il inclut également un script JavaScript permettant l'animation de ses composants se basant sur jQuery et son plugin jQueryUI.

4.2.2 MODÈLE

Le client est capable de communiquer avec le web service et donc d'effectuer toutes les opérations du CRUD pour des entités de types Auteur et Livre. Nous avons donc choisi de créer deux classes Author (figure 1) et Book (figure 2) permettant d'effectuer ces opérations. De plus, ces classes sont capables de donner une visualisation des informations des entités dans l'interface. Les classes créées sont dépendantes de jQuery.

Author
<div><div>-id: Number</div><div>-name: String</div><div>-firstName: String</div><div>-address: String</div></div>
<div><div>+create(): Author</div><div>+read(query: String): List<Author></div><div>+read(id: Number): Author</div><div>+update(name: String, firstName: String, address: String)</div><div>+delete()</div></div>

FIGURE 1: Classe Author

En effet, leur création utilise les fonctionnalités de celui-ci pour pouvoir obtenir une syntaxe claire.

¹ metroui.org.ua

Book
-id: Number -title: String -description: String -price: Number -authorId: Number
+create(): Book +read(query: String): List<Book> +read(id: Number): Book +update(title: String, description: String, price: Number, authorId: Number) +delete()

FIGURE 2: Classe Book

4.2.3 FONCTIONNEMENT

Dans cette section, nous verrons les fonctionnalités des différentes vues de l'application. La vue principale contient la barre de navigation (figure 3). Celle-ci contient un élément avec le nom de l'application qui permet de revenir sur la page d'accueil, une barre de recherche qui permet de rechercher une chaîne de caractère parmi les auteurs et les livres. Enfin, un bouton d'ajout permet d'ajouter un nouveau livre ou un nouvel auteur (figure 4).



FIGURE 3: Vue principale

Lorsque l'on clique sur le bouton d'ajout d'auteur, un nouvel auteur est créé (figure 5). Les propriétés de l'auteur peuvent être éditées en cliquant sur les éléments. Ceux-ci sont ensuite transformés en input. La perte de focus de l'élément remet l'élément à son état d'origine (h1, h2 ou h3) et appelle la méthode mise à jour auprès du web service. La figure 6 montre une vue avec un auteur modifié. L'auteur peut également être supprimé en cliquant sur l'icône de corbeille en haut à droite.

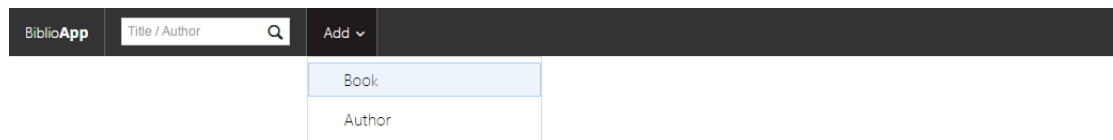


FIGURE 4: Bouton d'ajout



FIGURE 5: Ajout d'un auteur

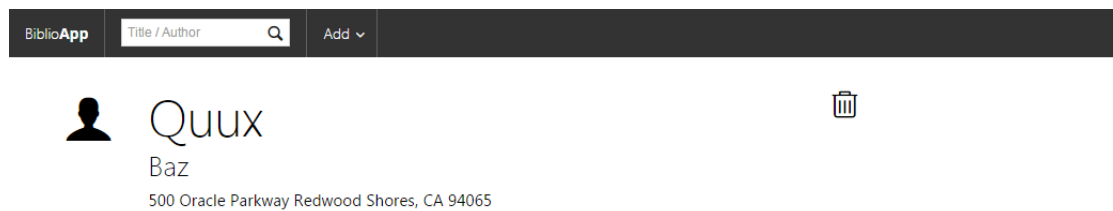
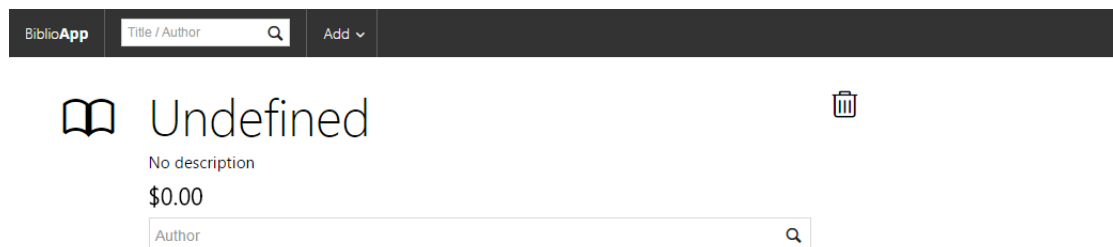


FIGURE 6: Auteur modifié

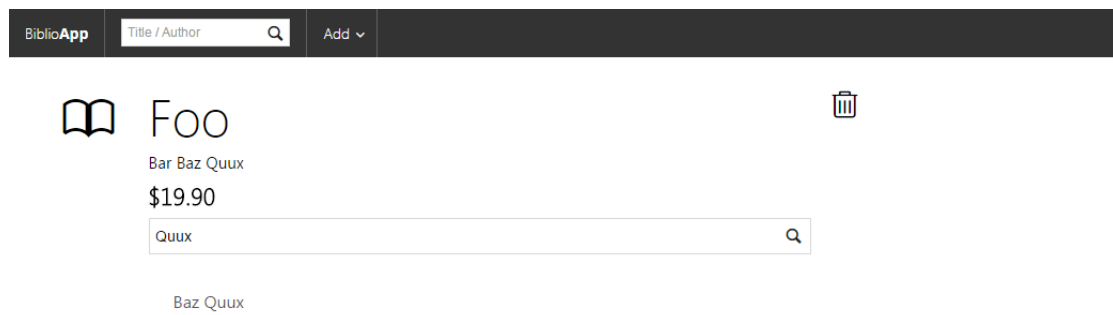
L'ajout d'un livre est également possible en cliquant sur le bouton d'ajout de livre (figure 7). Les propriétés du livre peuvent également être modifiées en cliquant sur

les éléments. Ceux-ci sont transformés en input ou textarea. Comme pour l’auteur, les éléments reviennent à leur état d’origine (h1, p et h3) avec la perte de focus et sont ensuite mis à jour auprès du serveur. Pour changer l’auteur du livre, on peut effectuer une recherche dans la barre de recherche du formulaire. La liste des auteurs correspondant sera ensuite affichée (figure 8). Un clic sur l’un d’entre eux change l’auteur du livre. Pour changer à nouveau l’auteur, il faut cliquer sur l’icône de crayon pour réafficher la barre de recherche. La figure 9 montre une vue avec un livre modifié. On peut également



The screenshot shows the top navigation bar of the BiblioApp with a search input labeled 'Title / Author' and an 'Add' button. Below the bar, a book form is displayed. It features a book icon, the title 'Undefined', a trash icon, and the text 'No description'. The price is set to '\$0.00'. There is an 'Author' input field with a search icon. At the bottom, there is a horizontal line.

FIGURE 7: Ajout d’un livre



The screenshot shows the same BiblioApp interface as Figure 7, but with the title 'Foo' and the author 'Bar Baz Quux'. The price is now '\$19.90'. The 'Author' input field contains 'Quux' and has a search icon. Below the input field, the text 'Baz Quux' is displayed.

FIGURE 8: Recherche d’un auteur pour un livre

Enfin, la barre de recherche de la barre de navigation permet d’effectuer la recherche d’auteurs et de livres existants. Si aucun paramètre n’est passé, la recherche renvoie tous les auteurs et livres (figure 10). Sinon, le paramètre passé est recherché parmi les auteurs et les livres et les éléments correspondant sont affichés (figure 11).

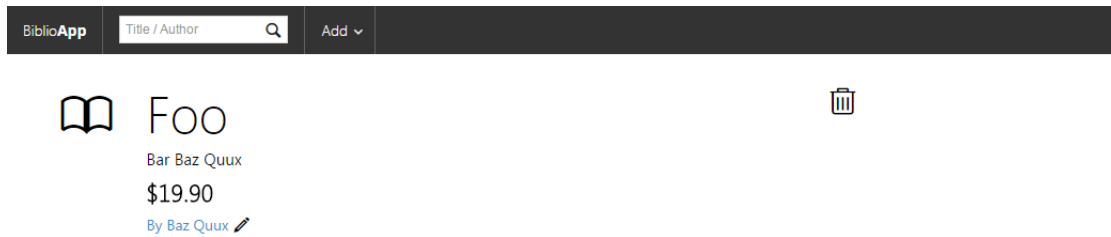


FIGURE 9: Livre modifié

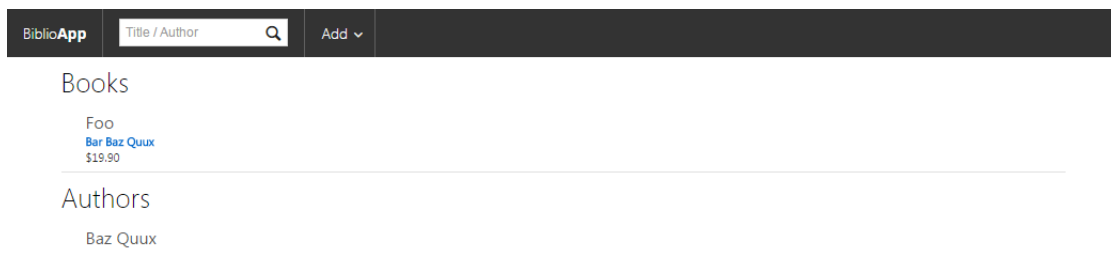


FIGURE 10: Recherche sans paramètre



FIGURE 11: Recherche avec paramètre

4.3 COMMUNICATION AVEC LE SERVEUR

Dans cette section, nous verrons plus en détails la façon dont le client commu-
nique avec le serveur. Nous effectuons des XHR (XML Http Requests) auprès du web

services. Les classes `Book` et `Author` possèdent des méthodes permettant d'effectuer ces requêtes. Les web services utilisant un format soap, nous avons utilisé le plugin `jQuery soap` afin de construire les requêtes à partir d'un JSON. Les requêtes reçues sont ensuite retransformées en JSON à l'aide de la bibliothèque `xml2json`.

Les objets JavaScript sont ensuite construits grâce aux JSON renvoyés par les web services répondant aux requêtes `read()` et `create()`. Les requêtes `delete()` et `update()` n'utilisent pas la réponse à la requête.

RÉSULTATS ET TESTS

CONCLUSION
