



TECH'4'TEAM

L'INGÉNIERIE AU SERVICE DU SPECTACLE

UNIVERSITÉ DE TECHNOLOGIE DE BELFORT-MONTBÉLIARD

Développement logiciel dans le domaine de la Business Intelligence afin d'optimiser la tarification des billets des stades et salles de spectacle

Rapport de stage ST50 – P2015

LARDIES Ludovic

Département Informatique
Filière Ingénieur des logiciels et de la connaissance

Tech'4'Team

152 Bd MacDonald
75019 Paris
<http://tech4team.com>

Tuteur en entreprise
VITOZ Kevin

Suiveur UTBM
LAMSOURIS Sid

TABLE DES MATIÈRES

1 Présentation de l'entreprise	5
1.1 Inventor-e	5
1.2 ArenaMetrix	6
1.2.1 ArenaPublic	6
1.2.2 ArenaProspection	6
1.2.3 ArenaPricing	7
1.2.4 Objectifs	7
1.3 Fonctionnement de l'entreprise et répartition des tâches	7
1.3.1 Organigramme de l'entreprise à la fin de mon stage	8
2 Mise en situation et activités effectuées	10
2.1 Importation manuel de données billetteries des clients dans ArenaPricing	10
2.1.1 Le besoin	10
2.1.2 Première réalisation du front d'importation	12
2.1.3 Deuxième itération pour le logiciel d'import des fichiers	16
2.1.4 Les API des logiciels billetteries	18
2.1.5 La base de données en elle même	19
2.2 ArenaPublic	20
2.2.1 Première approche et problème technique	20
2.2.2 Implémentation technique	21
2.2.3 Remplissage de la base de données	22
2.2.4 Enrichissement des données	24
2.2.5 Implémentation de bout en bout	26
2.3 Accueil et encadrement de stagiaires	27
3 Résultats et conclusion	28
3.1 Résultats	28
3.2 Conclusion	28
A Annexes	30
A.1 Base de données ArenaPricing	31
A.2 Base de données ArenaMetrix	32

Introduction

Dans le cadre de ma formation d'ingénieur à l'Université de Technologie de Belfort Montbéliard j'ai dû effectuer un stage de vingt-six semaines à l'issue de la dernière année de mon cycle d'ingénieur. Ce stage m'a permis de me familiariser d'avantage avec le monde de l'entreprise et de mettre en pratique les connaissances acquises tout le long de mon cursus scolaire.

J'ai effectué mon stage au sein de la startup Tech4Team située à Paris.

Durant ce stage j'ai eu l'occasion de travailler avec des technologies comme Ruby on Rails, Python, PostgreSQL, Riak, différentes API,...

Dans la première partie de ce rapport, je présenterai l'entreprise Tech4Team en faisant un bref aperçu des logiciels développés et de l'organisation de l'équipe. Ensuite j'aborderais les aspects plus technique avec le travail que j'ai eu à effectuer et les problèmes que j'ai rencontrés et eu à traiter.

Je terminerai par une conclusion, les résultats et ferai le bilan de ce stage.

Remerciements

Je tiens à remercier Kevin Vitoz et Ludovic Bordes, qui m'ont fait confiance, m'ont bien accueilli et m'ont fait découvrir leur entreprise.

Je remercie également Marwan Rabbaa pour ses précieux conseils techniques, et son soutien.

Présentation de l'entreprise

Tech4Team est une startup fondée en juillet 2013 par deux entrepreneurs. Elle propose aujourd'hui deux produits majeurs Inventor-e et ArenaMetrix.

1.1 Inventor-e

Inventor-e est un logiciel de gestion de planning, de suivi d'états des lieux et de pilotage de maintenance. Il est conçu pour les infrastructures sportives et culturelles multifonctions.

Le logiciel fonctionne sur le principe de client-serveur. L'utilisateur utilise une tablette Android pour faire un état des lieux ou visualiser son planning. Les photos et autres données sont ensuite synchronisées dans une base de données sur un serveur distant. Il est alors possible pour l'utilisateur de visualiser et de mettre à jour les informations depuis un portail web.



FIGURE 1.1 – Principe de fonctionnement d’Inventor-e

D'une manière plus technique l'application est développée pour Android 4.0 ou ultérieur, la base de données utilise PostgreSQL et l'interface web est développée avec le framework Ruby on Rails.

1.2 ArenaMetrix

ArenaMetrix est une solution logicielle, flexible, clé en main et intégrée de Big Data (analyse et exploitation des données billetteries), disponible en SaaS, pour la connaissance et la fidélisation des clients, la prospection ciblée et la mise en place d'une nouvelle politique de tarification.

ArenaMetrix se décompose en trois briques technologiques indépendantes ou combinées :

- ArenaPublic : Socle e-CRM
- ArenaProspection : Prospection ciblée
- ArenPricing : Tarification dynamique



FIGURE 1.2 – Vision à 360° de ArenaMetrix

1.2.1 ArenaPublic

ArenaPublic est le socle e-CRM d'exploitation des données publics.

Il permet :

- La centralisation des données publiques, B to B et B to C
- Le nettoyage et enrichissement des données (sexe, âge, lieu de résidence, fréquentation du stade, dépenses merchandising...)
- La visualisation graphique de la base de données à 360°
- L'analyse mono variable et bi variable en croisant des données identitaires et de consommation

1.2.2 ArenaProspection

Module d'optimisation du démarchage commercial.

Ce module permet :

1.3. FONCTIONNEMENT DE L'ENTREPRISE ET RÉPARTITION DES TÂCHES

- La sélection d'une liste de contacts
- Le classement des prospects selon leur probabilité de souscription
- La gestion interfacée d'une campagne de téléprospection, de mailing et de SMS auprès des prospects
- L'analyse des retours : réaction au démarchage, taux de conversion et intégration à la priorisation

1.2.3 ArenaPricing

Module de yield management.

Ce module permet :

- La création de typologies de publics par segmentation et profilage
- La prévision de la demande en billets pour un événement
- Recommandations tarifaires en fonction de la date et de la catégorie
- La gestion des flux par le prix, en transparence avec le client final

1.2.4 Objectifs

Tech4Team prévoit plusieurs objectifs pour son produit :

- Démocratisation de l'accès à la culture en ouvrant la voie à des prix différenciés
- Stabilisation voire hausse de la rentabilité des structures culturelles en augmentant le chiffre d'affaires billetterie de 5 à 7%

1.3 Fonctionnement de l'entreprise et répartition des tâches

Durant la majeure partie de mon stage l'équipe se composée de six stagiaires : cinq en développement et un commercial, un développeur expérimenté en CDI et les deux fondateurs.

Sur ArenaMetrix :

- Un stagiaire s'occupait du front-end et de l'affichage des données
- Un stagiaire s'occupait de la partie analyse de données
- Un stagiaire s'occupait du scoring des données
- Le développeur en CDI validait nos choix techniques sur le produit et jouait également le rôle d'administrateur système
- Moi, je m'occupais du back-end et de l'importation des données

Sur Inventor-e :

- Un stagiaire s'occupait de l'application Android
- Le développeur en CDI s'occupait de la base de données et de l'interface web

1.3. FONCTIONNEMENT DE L'ENTREPRISE ET RÉPARTITION DES TÂCHES

Les deux fondateurs s'occupent des relations clients avec le commercial et nous attribuaient des tâches pour améliorer et faire évoluer les produits. Des réunions sont régulièrement organisées pour définir les architectures logiciels, de base de données et les moyens techniques à utiliser.

La gestion de projets se fait avec l'outil open source Redmine. Pour la gestion du code source, nous utilisons Git et la plateforme Gitlab.

1.3.1 Organigramme de l'entreprise à la fin de mon stage

Comme le montre la figure 1.3 page 9, à la fin de mon stage nous étions 13 dans l'entreprise

- Le lead développeur est en CDI depuis 1 an, il s'occupe de toute la partie technique de l'entreprise, que ce soit pour Inventor-e en développant activement l'application web et pour ArenaMetrix mais de manière plus macro.
- Le directeur du développement est en CDI, recruté récemment il s'occupe de faire le lien entre les clients et les développeurs.
- Le responsable développement mobile est un ancien stagiaire passé en CDD, il s'occupe de l'application Android d'Inventor-e.
- Le responsable front-end est un ancien stagiaire et est maintenant en CDD
- Moi qui suis maintenant responsable de la partie back-end, j'ai signé un CDI à la fin de mon stage
- Les deux data scientist, un va signer un CDI à la fin de son stage et l'autre doit faire de l'alternance dans une autre entreprise
- Les autres développeurs et la responsable marketing et communication sont des stagiaires pour 6 mois

1.3. FONCTIONNEMENT DE L'ENTREPRISE ET RÉPARTITION DES TÂCHES

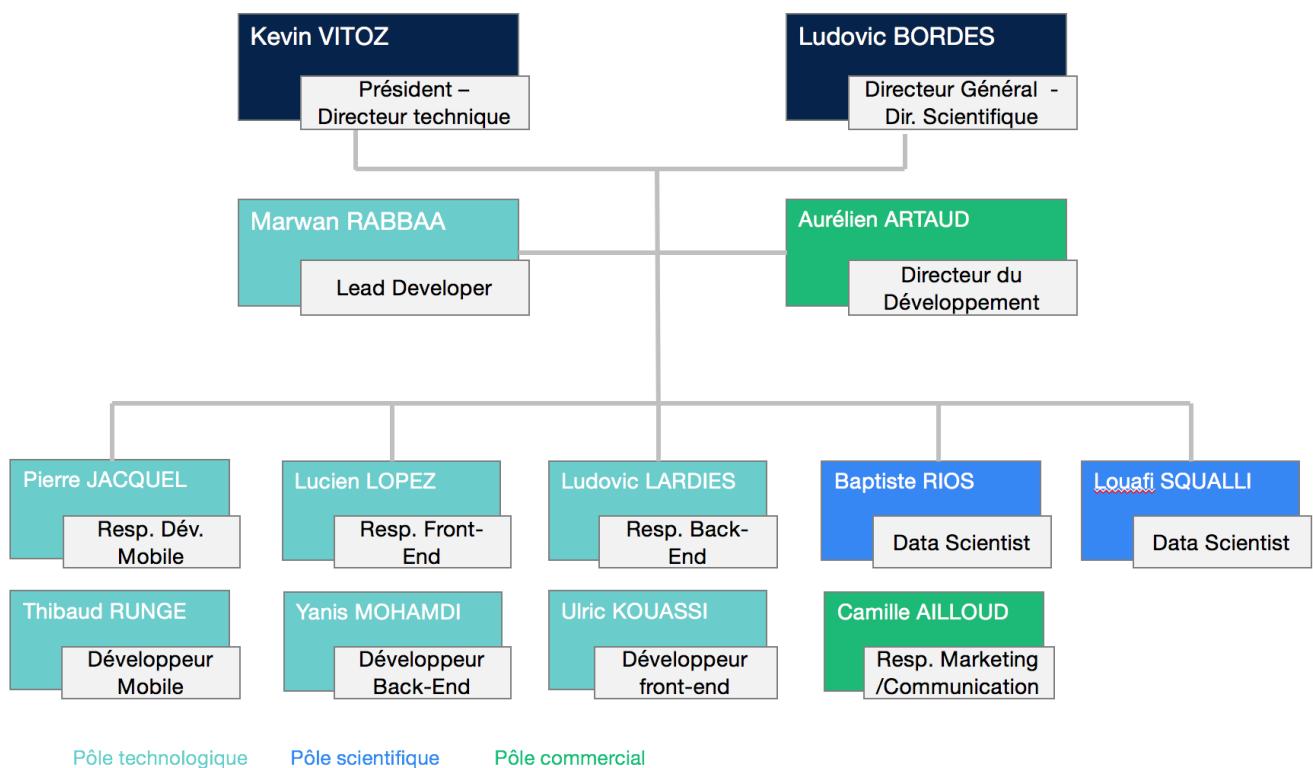


FIGURE 1.3 – Organigramme de l'entreprise

2

Mise en situation et activités effectuées

Lors de mon entretien de stage les fondateurs de l'entreprise m'ont indiqué que j'allais utiliser les technologies Ruby et Python. Le sujet était donc de travailler avec le CTO sur la base de données et de coder le back-end de l'importation de données billetteries.

À mon arrivée l'architecture d'une base de données était déjà réalisée. Mais elle n'était pas utilisée et encore moins remplie. J'ai donc du rapidement l'adapter en fonction des premières données billetteries que j'ai reçue et que je devais insérer. Plusieurs stagiaires avaient géré la base de données auparavant, certaines tables étaient redondantes (parfois en français et en anglais), certaines autres ne contenaient pas de clés étrangères (juste l'id d'une autre table). J'ai donc au fur et à mesure corrigé tout ça.

2.1 Importation manuel de données billetteries des clients dans ArenaPricing

L'une de mes premières tâches en arrivant dans l'entreprise est de créer un logiciel permettant à Tech4Team et à certains clients de facilement insérer dans notre base de données un fichier CSV¹ contenant des données billetteries. À plus long les fichiers CSV ne doivent plus être utilisés, le but final est d'intégrer les API des logiciels de billetterie des clients afin de pouvoir faire du pseudo temps réel.

2.1.1 Le besoin

Le besoin peut donc être visualisé par le schéma 2.1 page 11 ou sur le diagramme de cas d'utilisation 2.2 page 12. Tech4Team ou un client veut insérer de nouveaux tickets dans la base de données ArenaPricing. Il sélectionne alors un fichier CSV à la main

1. Comma-separated values, connu sous le sigle CSV, est un format informatique ouvert représentant des données tabulaires sous forme de valeurs séparées par des virgules.

2.1. IMPORTATION MANUEL DE DONNÉES BILLETTERIES DES CLIENTS DANS ARENAPRICING

depuis l'interface web, le site analyse le fichier, extrait les informations importantes et les insère en base de données. Toute la partie d'analyse et d'insertion doit être invisible du point de vue de l'utilisateur.

Le logiciel d'upload doit également être capable de s'interfacer avec des API.

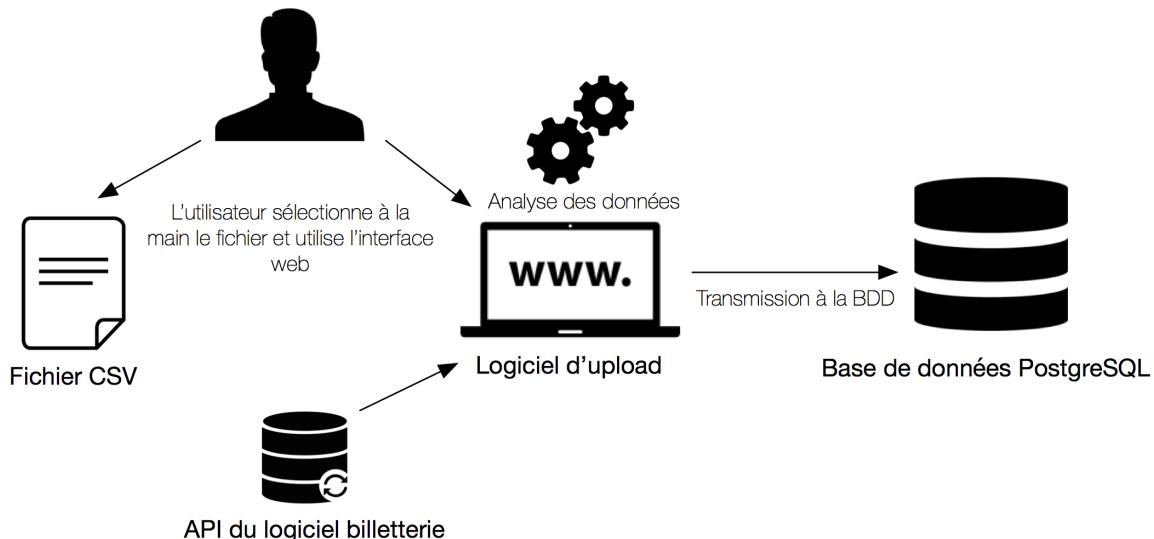


FIGURE 2.1 – Principe de l'interface d'upload

Le diagramme de cas d'utilisation 2.2 page 12 détail les différentes possibilités d'envoie de données pour un utilisateur :

- Via un fichier CSV d'export contenant les données billetteries, le client importe envoie ce fichier sur l'interface d'upload du site.
- Via une API, il peut demander directement le "rapatriement" des données

2.1. IMPORTATION MANUEL DE DONNÉES BILLETTERIES DES CLIENTS DANS ARENAPRICING

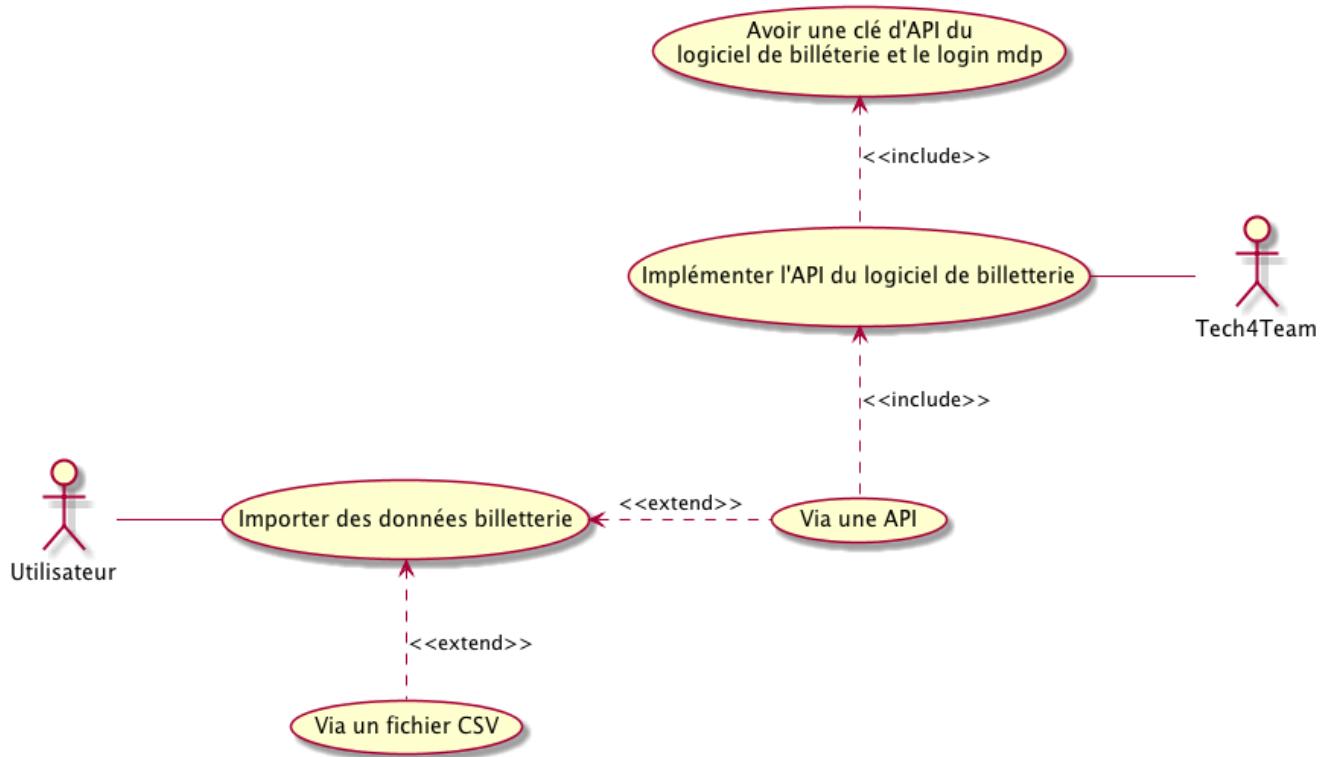


FIGURE 2.2 – Diagramme de cas d'utilisation de l'interface d'upload d'ArenaPricing

2.1.2 Première réalisation du front d'importation

Pour réaliser ce site web d'importation nous avons décidé après discussion avec le lead développeur d'utiliser le langage Python. Ce langage permet de développer rapidement et dispose de nombreux modules s'ajoutant aux fonctions de base. Il est ainsi aisément de parser un fichier CSV. J'ai également utilisé le microframework Flask pour créer les pages web en elle-même et servir de serveur web. Pour le design du site, j'ai utilisé le framework Zurb Foundation afin d'avoir une identité visuelle cohérente avec le reste du site ArenaMetrix.

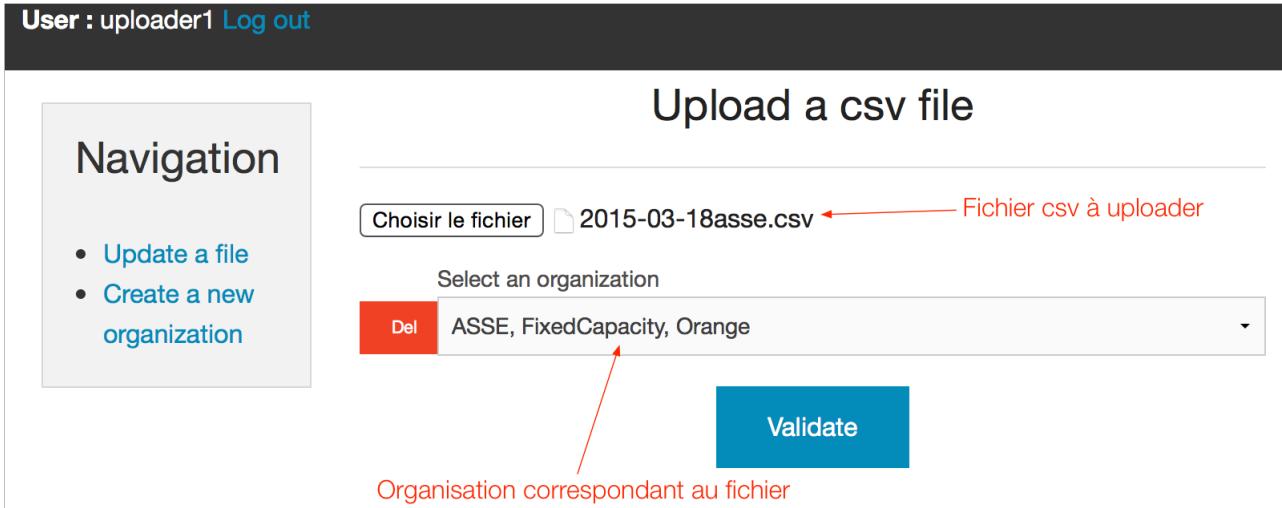
2.1. IMPORTATION MANUEL DE DONNÉES BILLETTERIES DES CLIENTS DANS ARENAPRICING

 python™ + Module d'import de CSV	Analyse et traitement des données
 Flask web development, one drop at a time 	Afficher le site web + Serveur web + Interagir avec l'utilisateur

FIGURE 2.3 – Téchnologies utilisées pour le front

2.1.2.1 L'interface homme-machine obtenue

Comme nous pouvons le voir sur l'impression d'écran 2.4 page 13 l'interface permet à l'utilisateur d'envoyer un fichier CSV préalablement sélectionné. En plus du nom de l'organisation, nous affichons le type d'événement correspondant au fichier envoyé ainsi que le logiciel de billetterie d'où provient l'export.



The screenshot shows a user interface for uploading a CSV file. At the top, it says "User : uploader1 Log out". On the left, there's a "Navigation" sidebar with links to "Update a file" and "Create a new organization". The main area has a title "Upload a csv file". It includes a "Choisir le fichier" button with a file path "2015-03-18asse.csv" and a red arrow pointing to it labeled "Fichier csv à uploader". Below that is a "Select an organization" dropdown menu containing "ASSE, FixedCapacity, Orange" with a red arrow pointing to it labeled "Organisation correspondant au fichier". At the bottom right is a blue "Validate" button.

FIGURE 2.4 – Front d'upload d'un fichier CSV

2.1. IMPORTATION MANUEL DE DONNÉES BILLETTERIES DES CLIENTS DANS ARENAPRICING

2.1.2.2 Parsing des données en Python

Pour parser les fichiers CSV j'utilise le module natif CSV. Le principal problème avec les fichier CSV est l'encodage, parfois c'est du UTF-8, d'autres fois du latin1 ou encore du windows-1252. Le module *chardet* permet de détecter l'encodage d'un fichier.

```
f = open(p_path_file, 'rb')
charset_char = f.read(10000) #read 10000 first characters to detect charset
f.close()
charset = chardet.detect(charset_char)
with open(file_path, 'r', encoding=charset['encoding']) as f:

    #read csv file with csv module
    dict_csv = csv.DictReader(f, delimiter=';', quoting=csv.QUOTE_ALL)

    #use strategy design pattern with Secutix object
    secutix = ParsingStrategyContext(Secutix())

    #parse dictionay and send organization name, event date, export date
    secutix.parse_file(dict_csv, o_name, e_date, export_date)

#ask object to send data to the ruby. e_type is event type
reason = secutix.upload_data(e_type)
return reason
```

Listing 2.1 – Code analysant le fichier CSV

La fonction *open* permet d'ouvrir le fichier *file_path*, ensuite je récupère le dictionnaire associé à au fichier CSV avec *csv.DictReader*. La méthode *parse_file* permet de générer le dictionnaire avec les informations des tickets/achats comme le montre la figure 2.5 page 15, nous obtenons finalement une liste de dictionnaire où chaque entrée de la liste est une ligne du fichier CSV.

2.1. IMPORTATION MANUEL DE DONNÉES BILLETTERIES DES CLIENTS DANS ARENAPRICING

```
► └─ 'billing' (4575243376) = {dict} {'date': '2014-06-02T13:53:00', 'price': {'price': 48.0}}
► └─ 'customer' (4591794864) = {dict} {'address': {}, 'number': '21090195-164648028-02/06/2014 13:53', 'phone': {}, 'identity': {}}
▼ └─ 'event' (4560780064) = {dict} {'exportdate': '2015-05-27T00:00:00', 'name': 'les parapluies de cherbourg', 'place': 'théâtre du châtelet', 'fixed_capacity': True}
    └─ __len__ = {int} 8
    └─ 'date' (4559662136) = {str} '2014-09-11T20:00:00'
    └─ 'exportdate' (4591795120) = {str} '2015-05-27T00:00:00'
    └─ 'fixed_capacity' (4591813744) = {bool} True
    └─ 'name' (4555958456) = {str} 'les parapluies de cherbourg'
    └─ 'organizer' (4591794992) = {str} 'le châtelet'
    └─ 'place' (4571653992) = {str} 'théâtre du châtelet'
    └─ 'season' (4591808784) = {str} '2014-2015'
    └─ 'type' (4555957672) = {str} 'normal'
    └─ 'partner' (4575243488) = {dict} {}
▼ └─ 'purchase' (4591794928) = {dict} {'currency': 'eur', 'chanel': 'migr'}
    └─ __len__ = {int} 2
    └─ 'chanel' (4591809064) = {str} 'migr'
    └─ 'currency' (4558260784) = {str} 'eur'
    └─ 'tariff' (4575243544) = {dict} {}
    └─ 'ticket' (4575243432) = {dict} {'number': '21090195-164648028', 'blockcapacity': 263, 'category': 'cat.3', 'seat': {'number': '10'}, 'rank': 'V', 'block': 'cat.3'}
```

FIGURE 2.5 – Dictionnaire récupéré pour une ligne du fichier CSV

D'un point de vue architecture de l'application j'utilise le design pattern strategy. Ce design pattern est utile lorsqu'un objet peut effectuer plusieurs traitements différents, dépendant d'une variable ou d'un état.

L'implémentation de ce design pattern peut être visualisé sur le diagramme figure 2.6 page 15.

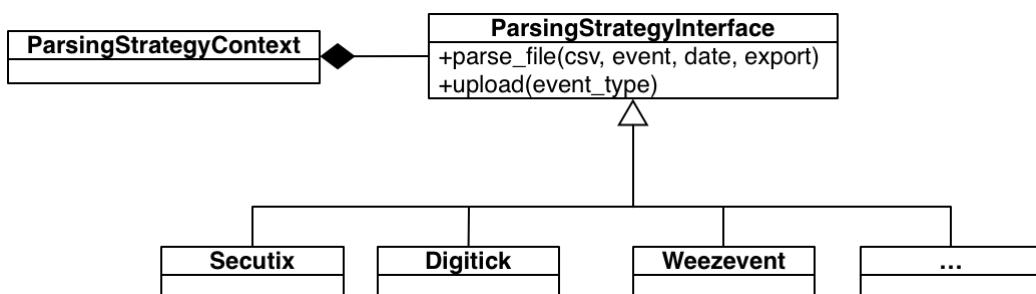


FIGURE 2.6 – Design pattern strategy

2.1.2.3 Problématique posée par ce logiciel

Après avoir testé le logiciel d'importation avec de nombreux clients, il s'avère que les fichiers fournis ne sont pas générique.

Les noms des colonnes ne sont pas les mêmes en fonction des exports alors que le logiciel de billetterie est identique et dans certains cas le changement apparaît pour un même client d'une saison à l'autre.

On se rend donc rapidement compte que nous ne pouvons pas créer un script unique par logiciel de billetterie.

La partie front en Flask n'est actuellement plus utilisée, mais le python qui permet de parser les données et utilisé pour les clients qui n'ont pas d'API et qui uploadent donc des fichiers CSV sur un serveur FTP.

2.1. IMPORTATION MANUEL DE DONNÉES BILLETTERIES DES CLIENTS DANS ARENAPRICING

2.1.3 Deuxième itération pour le logiciel d'import des fichiers

Nous décidons de repenser notre logiciel pour que le client désirant envoyer un fichier CSV puisse spécifier sur l'interface les champs qu'il souhaite importer.

Comme le montre l'image 2.7 page 16 du premier draft que j'ai réalisé, le client doit maintenant faire un travail d'association entre les champs présents dans le fichier CSV leur correspondance réelle.

A Web Page
http://

Nom de l'événement : saisir à la main
Date de l'événement : saisir à la main
Nom de la facility : saisir à la main

Numéro de ticket

Formule de ticket Code postal de l'acheteur

Canal d'achat Adresse de l'acheteur

Date d'achat Ville de l'acheteur

Nom de l'acheteur Code postal de l'acheteur

Prenom de l'acheteur Pays de l'acheteur

Email de l'acheteur

Pour les champs uniques à l'importation et non disponibles dans le CSV on les rentre à la main.

FIGURE 2.7 – Premier draft du logiciel d'importation 2.0

2.1.3.1 Crédit de l'IHM

Pour réaliser l'interface d'upload j'ai utilisé le framework Ruby On Rails afin de pouvoir intégrer facilement et directement la page d'upload au reste du site. Après réflexion, il est plus simple pour nous de maintenir le logiciel principal avec une seule et même technologie : le Ruby. Le Python n'est actuellement utilisé que pour les scripts.

2.1. IMPORTATION MANUEL DE DONNÉES BILLETTERIES DES CLIENTS DANS ARENAPRICING

Liaison des champs disponibles

Données billetteries nécessaires	Données issues du fichier	Données génériques si non disponibles dans le fichier
* Nom de l'événement	Epreuve	
* Date de l'événement	Aucun	08/03/2015
Données billetteries nécessaires		Données issues du fichier
* Numéro ticket	Order ID	
Formule du ticket	Aucun	
* Prix du ticket	Paiement	
Catégorie du ticket	Aucun	
Canal d'achat	Aucun	
* Date d'achat	Date	

A	B	C	D	E	F	G	H	I	J	K	L	M	N
Epreuve	Date	Order ID	Paiement	Adresse ema	Nom	Prénom	Adresse	Code postal	Ville	Pays	Date de naiss:	Téléphone mobile	
Semi-Marath	01/10/2014 00:02	668539	50.00	cmartinsaint	MARTIN	S/Côme	8 rue de l'ind	3000	Moulins	France	26/01/1990	3,367E+10	
Semi-Marath	01/10/2014 00:04	668542	50.00	Mebmercredi	MEBARKI	Mercedes	24 rue des ol	77181	Courtry	France	29/05/1976	3,362E+10	
Semi-Marath	01/10/2014 00:07	668546	50.00	clairehurellet	CLAIRES		20 BIS RUE D	92200	Neuilly sur st France	France	05/05/1980	3,36E+10	
Semi-Marath	01/10/2014 00:14	668549	50.00	Julie.pierre3@PIERRE	Julie		14 rue Cler	75007	Paris	France	14/03/1989	3,363E+10	
Semi-Marath	01/10/2014 00:32	668556	50.00	yoh.marchar	MARCHAN	Yohann	10 allée Ann	75019	Paris	France	30/07/1983	3,37E+10	

Valider

FIGURE 2.8 – Logiciel d'importation des clients

J'obtiens finalement le front-end 2.8 page 17, comme nous pouvons le voir avec le fichier CSV de test les champs sont mappés ensemble. Si la donnée n'est pas présente, une insertion vide se fait en base de données. Les champs marqués avec une étoile sont indispensables. Certains champs si ils sont uniques à l'import, comme le nom de l'événement et la date peuvent être inscrits à la main s'ils ne sont pas fournis dans le fichier CSV.

2.1.3.2 L'importation des fichiers CSV

Afin d'importer les données dans le logiciel de billetterie j'utilise l'ORM ActiveRecord.

```
begin
  country = Pricing::Country.find_by!(name: customer_country_to_ins)
```

2.1. IMPORTATION MANUEL DE DONNÉES BILLETTERIES DES CLIENTS DANS ARENAPRICING

```
rescue ActiveRecord::RecordNotFound
    country = Pricing::Country.new(name: customer_country_to_ins)
end
country.save
begin
    department = Pricing::Department.find_by!(name: customer_zip_to_ins,
        country: country)
rescue ActiveRecord::RecordNotFound
    department = Pricing::Department.new(name: customer_zip_to_ins, country:
        country)
end
department.save
begin
    city = Pricing::City.find_by!(name: customer_city_to_ins, department:
        department)
rescue ActiveRecord::RecordNotFound
    city = Pricing::City.new(name: customer_city_to_ins, department:
        department)
end
city.save
```

Listing 2.2 – Exemple de code insérant une adresse

Avant d'insérer une donnée, je vérifie si elle est présente, si ce n'est pas le cas je rescue l'exception et j'insère la donnée. L'ORM ActiveRecord permet de s'astreindre des contraintes habituelles du SQL, lorsque je fais *country : country* lors d'une intention de département l'ORM comprend qu'il doit lier *country* comme clé étrangère de *dpartement*.

2.1.4 Les API des logiciels billetteries

Différentes approches pour importer les données sont possibles :

- Le client nous donne le login et mot de passe de son logiciel billetterie : nous faisons l'extraction d'un fichier CSV et nous l'importons à la main dans la base de données.
- Le client envoie son fichier CSV par mail : nous l'importons à la main dans la base de données (et lui redemandons le fichier si il manque des champs...).
- Le client dépose son fichier CSV sur un serveur FTP : un script Python est chargé de régulièrement scanner ce serveur FTP et d'importer les données s'il y a un changement.
- La société logiciel billetterie nous donne une clé d'API, ainsi avec le mot de passe et login du client nous pouvons accéder aux données en nous connectant directement à la base de données. C'est l'idéal pour nous, cela nous permet d'automatiser les imports.

2.1. IMPORTATION MANUEL DE DONNÉES BILLETTERIES DES CLIENTS DANS ARENAPRICING

Actuellement nous avons accès aux API des billetteries WeezEvent, Njuko et Digitick. Pour récupérer les données des API j'utilise le module request qui permet de faire des requêtes get.

Ainsi la requête suivante me permet de récupérer les événements d'une organisation :

```
json_events = requests.get("https://api.weezevent.com/events",
                            params={'api_key': API_KEY,
                                     'access_token': token,
                                     'include_closed': True}
                            )
events = json_events.json()['events']
for event in events :
    #Traitement
```

Listing 2.3 – Récupération des événements sur l'API de weezevent

À la fin de mon stage, nous avions accès à trois API de logiciel de billetteries :

- WeezEvent pour les événements :
 - Mondial du tatouage : un salon annuel pour les tatoueurs du monde entier
 - File7 : Une salle de concert en région parisienne.
 - Rhum Fest : Un événement et une exposition sur le thème de rhum
- Njuko pour les événements :
 - Marathon de Paris
- Digitick pour :
 - Le musée Grévin

2.1.5 La base de données en elle même

L'architecture de la base de données ArenaPricing peut être visualisée sur la figure A.1 de l'annexe page 31.

Cette base de données de la première forme normale car tous ses attributs de la base de données ont des valeurs simples (non multiples, non composées).

Elle est également en deuxième forme normale car aucun attribut non clé ne dépend d'une partie de la clé.

Et de troisième forme normale car aucun attribut non clé ne dépend d'un ou plusieurs attributs ne participant pas à la clé.

2.2 ArenaPublic

Le logiciel ArenaPublic a pour but d'être un CRM amélioré, l'utilisateur doit pouvoir au travers de ce logiciel visualiser les données des clients en base de données.

À la différence de ArenaPricing où nous nous concentrons sur les tickets vendus, ici c'est le client qui est au centre de la base de données, à terme cette application doit permettre à l'utilisateur d'afficher des données de scoring².

La base de données ArenaPublic est reliée à la base de données ArenaPricing par l'intermédiaire de la base ArenaMetrix. ArenaMetrix est visible en annexe A.2 page 32, cette base identifie l'utilisateur du site web et gère l'accès aux pages avec la gem TheRole.

2.2.1 Première approche et problème technique

Après avoir défini avec les statisticiens un modèle de base de données relationnelle que nous pensons générique, nous nous rendons compte qu'il nous faut plus de souplesse sur la structure. En effet, les données à scorer et à afficher peuvent varier d'un utilisateur à l'autre et évoluent en fonction de la demande de l'utilisateur et des statisticiens.

2. Le scoring (statistique) est un ensemble de méthodes conduisant à un classement d'individus au sein de groupes préalablement définis ou de segmentation.

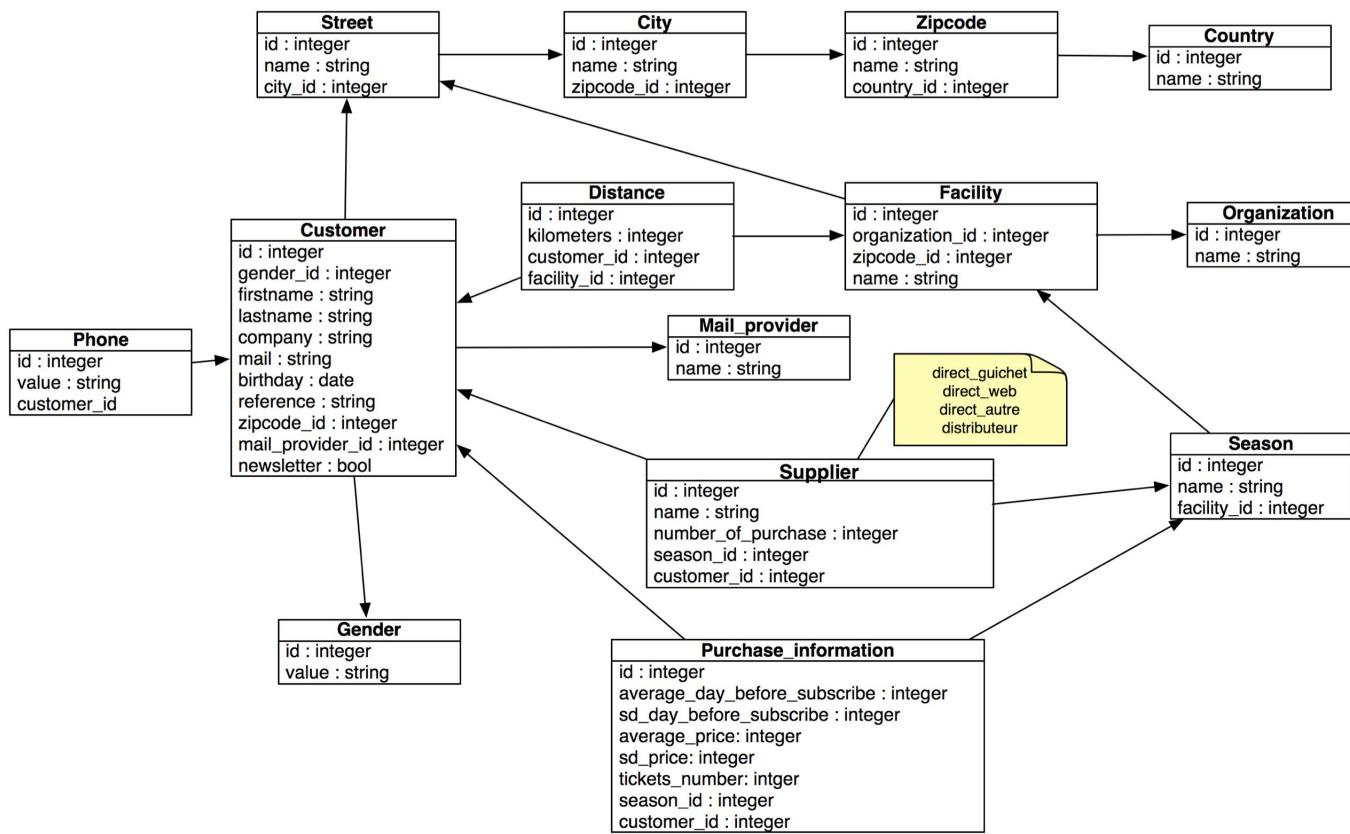


FIGURE 2.9 – Première architecture de la base de données ArenPublic

Comme le montre l'architecture de base de données A.1 page 31 des données comme des écarts types et des moyennes dans les tables "Supplier" et "Purchase information" sont calculées à la volé et utilisés par les statisticiens. Le problème est qu'il nous faut pouvoir ajouter de nouvelles données facilement en fonction de la demande du client ou du besoin des statisticiens.

Nous décidons donc d'utiliser en complément de la base de donnée SQL une base de base de données de type NoSQL³ qui permet de rajouter des informations "à la volée" sans modifier la structure de la base de données.

2.2.2 Implémentation technique

Pour la base de données NoSQL nous voulions une solution simple à mettre d'oeuvre de type clé valeur. Nous utilisons donc le système de base de données décentralisée Riak qui permet de stocker des informations de type clé valeur dans un bucket.

3. Not only SQL

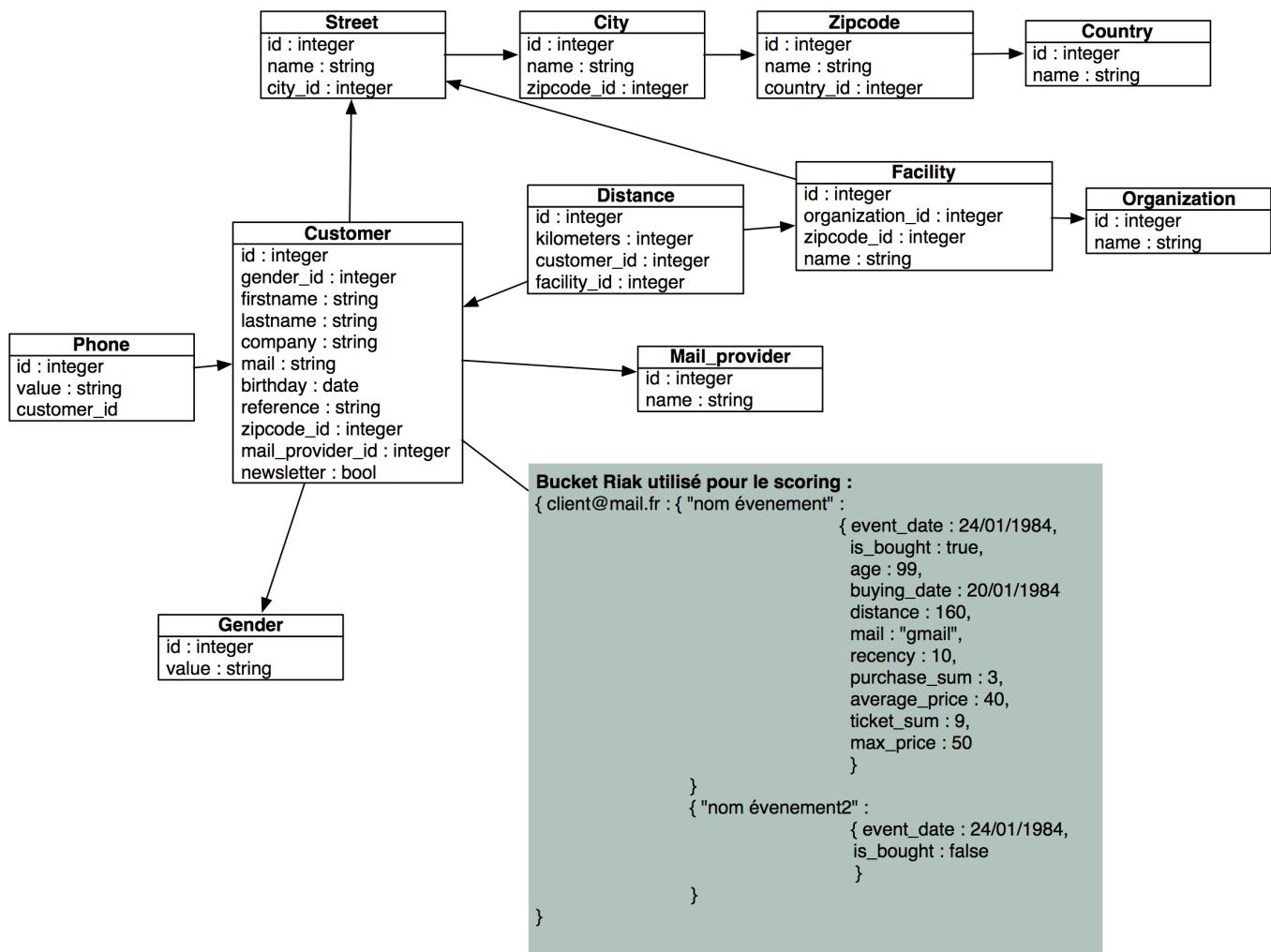


FIGURE 2.10 – Architecture finale de la base de données ArenaPricing

Finalement nous obtenons l'architecture de base de données visible en 2.10 page 22. La clé du Riak est l'adresse mail du client dans la base de données, il s'agit de la seule valeur vraiment indispensable pour identifier les clients de manière unique.

2.2.3 Remplissage de la base de données

Pour remplir la base de données ArenaPublic nous pouvons utiliser deux méthodes :

- Un front d'upload qui permet d'envoyer un fichier CSV avec les clients abonnés pour une saison spécifique.
- En allant piocher directement dans la base de données ArenaPricing.

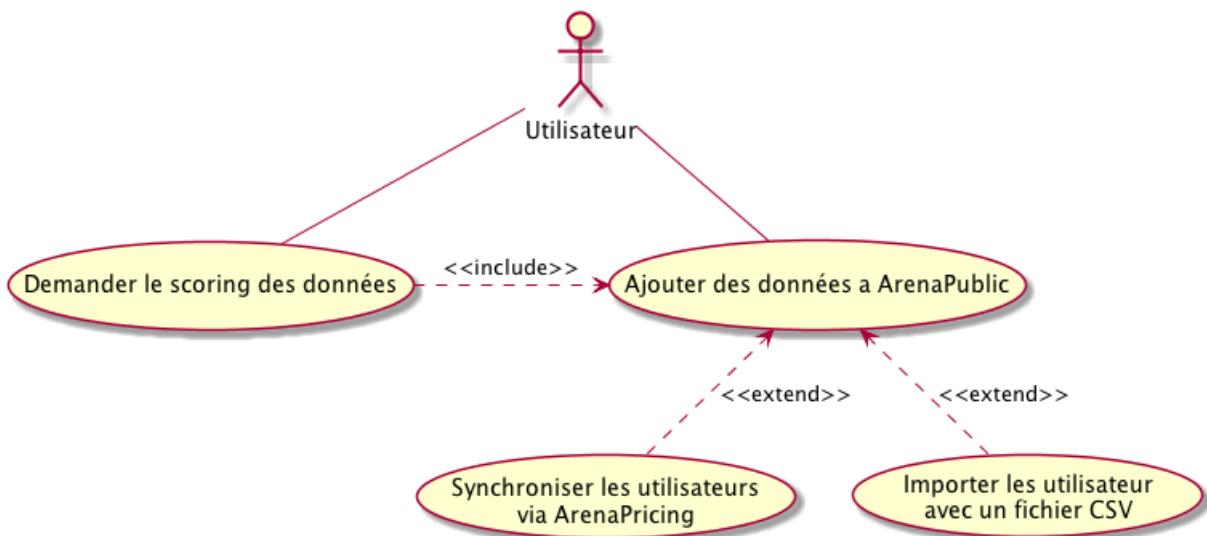


FIGURE 2.11 – Diagramme des cas d'utilisation de ArenaPublic

Le diagramme des cas d'utilisation 2.11 page 23 montre les différentes possibilités pour l'utilisateur d'importer des données relatives aux consommateurs afin de les scorer.

- Synchronisation des données depuis ArenaPricing
- Importation des données depuis un fichier CSV

Pour ma part j'ai essentiellement travaillé sur la synchronisation et l'agrégation des données depuis ArenaPricing.

Pour faire cela j'ai utilisé le langage Python, car il permet de facilement gérer des dictionnaires et de nombreux modules mathématiques sont disponibles pour fournir au statisticien des données aussi complètes que possible. Il ne leur reste ainsi qu'à les analyser.

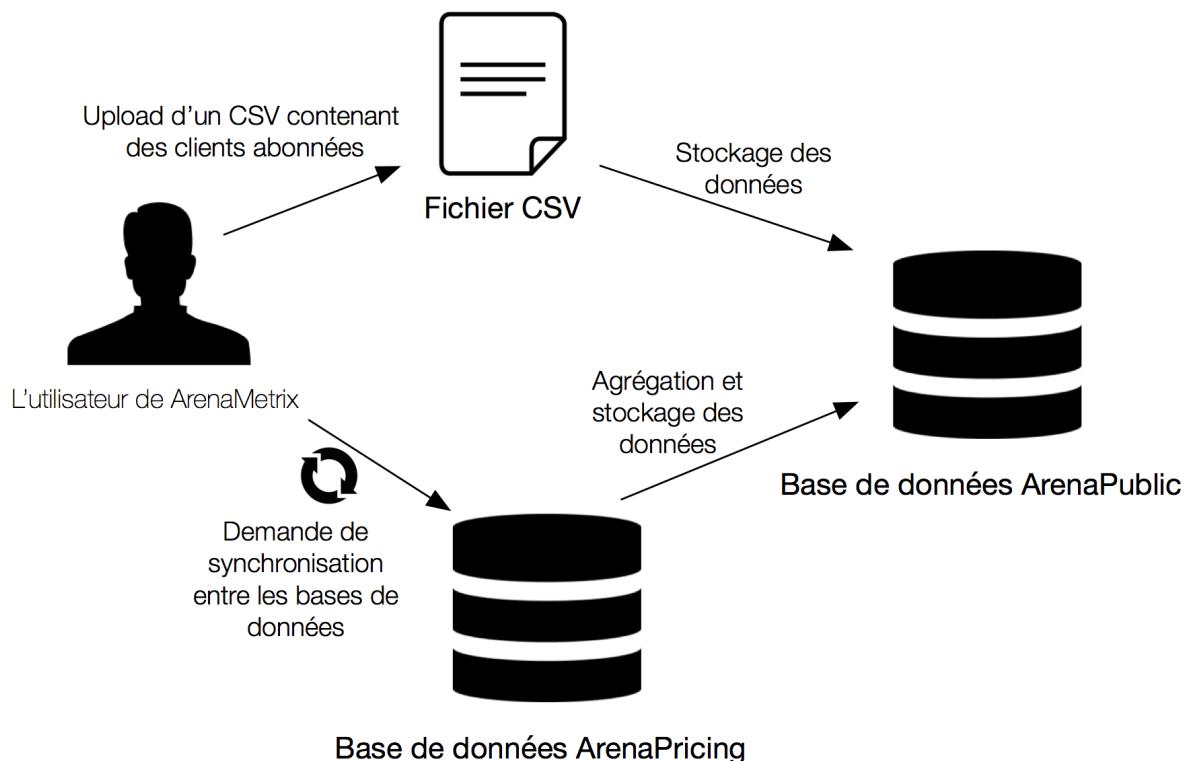


FIGURE 2.12 – Remplissage de la base de données ArenaPicing

Comme le montre la figure 2.12 page 24 il est possible pour l'utilisateur d'envoyer ses données au choix depuis un fichier CSV ou en synchronisant les données avec la base de données ArenaPricing.

La synchronisation des données avec ArenaPricing permet la réalisation d'un CRM à partir de données billetterie, d'un point de vue commercial ArenaPublic est uniquement disponibles pour le client ayant souscrit aux deux offres.

Tech4Team propose ainsi deux offres de scoring différentes :

- Scoring sur le réabonnement : pour les clients nous fournissant des données abonnées.
- Scoring sur l'achat : pour les clients ayant des données dans ArenaPricing

2.2.4 Enrichissement des données

Lors de l'agrégation des données de ArenaPricing dans ArenaPublic en plus de faire les calculs servant aux statisticiens, j'ai enrichi les données disponibles afin d'offrir plus de possibilités au statisticien.

2.2.4.1 Détermination du sexe

Ajout du sexe de la personne en fonction du prénom lorsque l'information n'est pas disponible : pour réaliser cet enrichissement de données, j'ai utilisé le module **SexMachine**. Ce module retourne male, female, mostly_male, mostly_female ou andy en fonction du prénom passé en paramètre.

```
>> import sexmachine.detector as gender
>>> d = gender.Detector()
>>> d.get_gender(u"Bob")
u'male'
>>> d.get_gender(u"Sally")
u'female'
>>> d.get_gender(u"Pauley") # should be androgynous
u'andy'
```

Listing 2.4 – Exemple d'utilisation de SexMachine

Le problème avec le module officiel de SexMachine répertorié sur pypi est qu'il est compatible uniquement avec la version 2.7 de Python. Hors nous utilisons Python 3.X pour l'ensemble de nos logiciels et scripts Python.

J'ai donc modifié SexMachine pour le rendre compatible avec Python 3.X. Le code source de la version modifiée est disponible sur mon repository Github : <https://github.com/ludovicl/sexfmachine>.

2.2.4.2 Calcul de la distance

Pour enrichir les données des clients nous calculons la distance entre l'adresse de l'acheteur et l'adresse de l'infrastructure où a lieu l'événement. Pour faire cela, j'utilise le module geocoder qui permet de wrapper différents sites de géocoding.

Après plusieurs tests j'utilise l'API de tomtom, car elle me donne les meilleurs résultats pour la France et n'a pas de limite d'utilisation.

```
>>> import geocoder
>>> location_customer = geocoder.tomtom('Paris')
>>> location_facility = geocoder.tomtom('12 Rue Thierry Mieg, Belfort')
>>> geocoder.distance(location_customer, location_facility)
359.0865720640995
```

Listing 2.5 – Exemple d'utilisation de geocoder

2.2.5 Implémentation de bout en bout

Pour faire communiquer nos applications de bout en bout nous utilisons Message-Pack : il s'agit d'un format de sérialisation binaire qui permet d'échanger des données comme en JSON.

Ainsi, comme le montre le diagramme 2.13 page 27 pour le scoring :

- Le client demande la synchronisation de sa base de données ticket ArenaPricing en cliquant sur un bouton
- L'interface (Ruby on Rails) lance une requête put une URL d'écoute qui est lancée en Flask depuis DataProc
- DataProc récupère les données d'ArenaPricing, les agrégés et les enrichie
- DataProc enregistre les nouvelles données dans les bases de données PostgreSQL et Riak d'ArenaPublic
- Le client est notifié que les données sont maintenant dans ArenaPublic
- Le client sélectionne les données qu'il veut scorer
- Les données sont récupérées dans la base de données
- Les données sont envoyées au logiciel de scoring écrit en Python. Le logiciel de scoring écoute sur une URL grâce à une instance lancée en Flask
- Le logiciel de scoring retourne les données à la WebInterface
- La WebInterface unpack le MesagePack et affiche les données associées sous forme de graphiques.

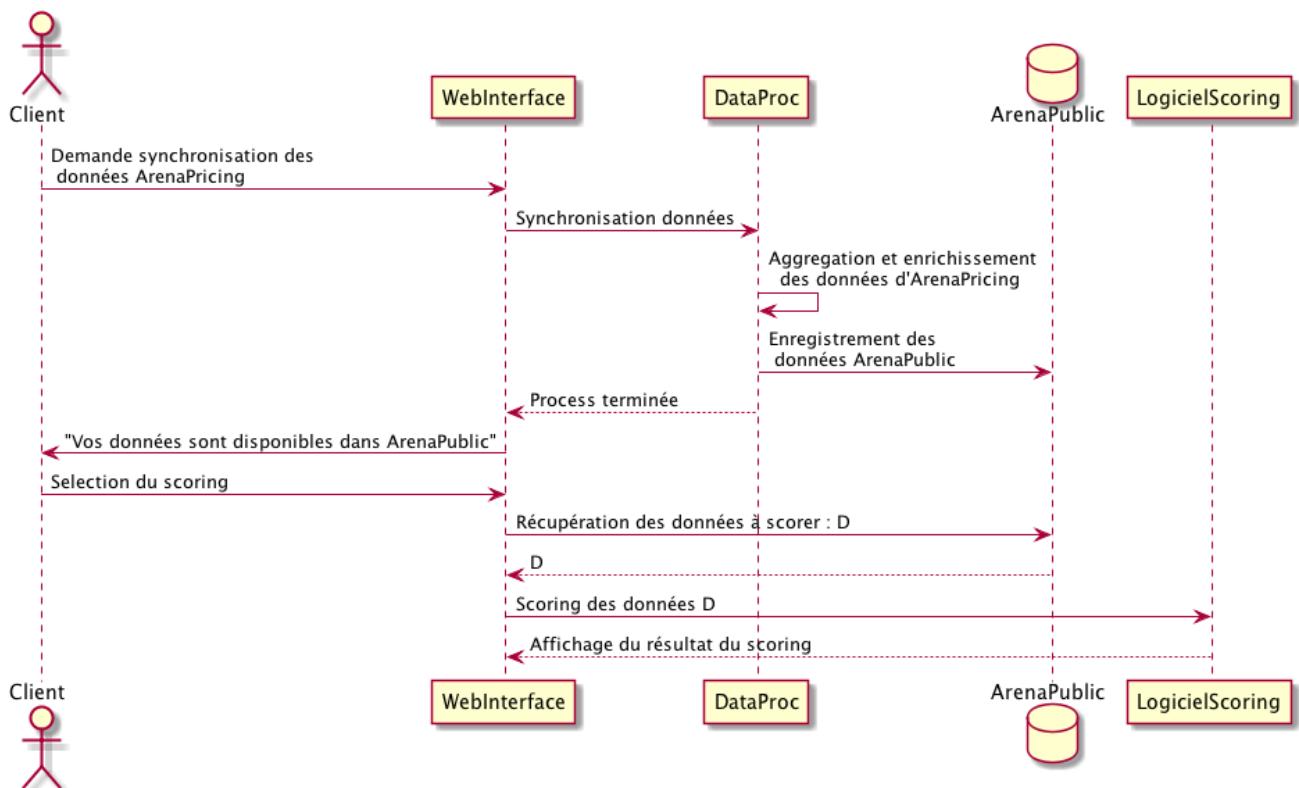


FIGURE 2.13 – Diagramme de séquence du scoring

2.3 Accueil et encadrement de stagiaires

Début juin de nombreux stagiaires ont été recrutés, ces stagiaires étant en première années de l'ETNA je les ai donc managés et encadrée d'un point de vue technique :

- J'ai notamment eu sous ma responsabilité directe un stagiaire back-end qui a travaillé sur du Python et qui a fait la partie récupération des données exogènes comme la météo et le scrapping de certains sites.
- Un autre stagiaire était chargé de faire le front-end de ArenaPublic, j'ai donc directement travaillé avec lui car je m'occupai de la partie Python et back-end Ruby de ce projet. Je l'ai également aidé pour les requêtes SQL à effectuer.

Résultats et conclusion

3.1 Résultats

Au terme de mon stage, nous avons 15 clients en bêta test, la plupart des données sont encore importées à la main avec des fichiers CSV mais plus l'entreprise grandie et gagne en notoriété et plus nous avons accès aux API des logiciels de billetteries.

D'ici quelques mois nous pourrons faire du temps réel sur la synchronisation des données et appliquer les modèles de prévisions à la volée.

3.2 Conclusion

Ce stage m'a permis de me familiariser davantage avec le monde de l'entreprise et cela de plusieurs manières :

- J'ai pu mettre en œuvre les connaissances acquises dans le domaine des bases de données dans un cadre professionnel
- Je me suis initié au management en devant m'occuper de stagiaires
- J'ai découvert le développement d'applications de type SaaS et les contraintes techniques que cela implique notamment au niveau des serveurs.

Après un premier stage effectué dans une entreprise de taille intermédiaire, je peux maintenant comparer ces deux mondes professionnels.

Les entreprises de tailles intermédiaires

L'avantage que je trouve à travailler en tant que stagiaire dans ce genre d'entreprise est l'encadrement. Les ingénieurs de ce type d'entreprise ont plus de disponibilités et peuvent nous guider dans l'exécution de certaines tâches. On peut s'appuyer sur leurs conseils.

On sait exactement où l'on va, le travaille que l'on a faire est les collègues peuvent prendre du temps pour nous aider. Par contre on est juste y engrenage dans une

machine beaucoup plus complexe.

Les start-ups

L'encadrement est beaucoup moins présent, le travail à faire évolue au cours des semaines et des besoins des clients. Le nombre d'employés étant moindre le stagiaire a plus de responsabilités, il faut prendre des décisions, implémenter des fonctionnalités rapidement, on se forme on apprend sur le tas, c'est très formateur. Le seul risque est de prendre des mauvaises habitudes, mais avec tous les logiciels d'analyse de codes et les ressources à notre disposition on s'en sort.

Mon stage s'est terminé le 31 juillet, l'entreprise m'a proposé un CDI que j'ai accepté. Mon rôle dans l'entreprise est de m'occuper de la partie back-end et de la conception des bases de données.

A

Annexes

A.1 Base de données ArenaPricing

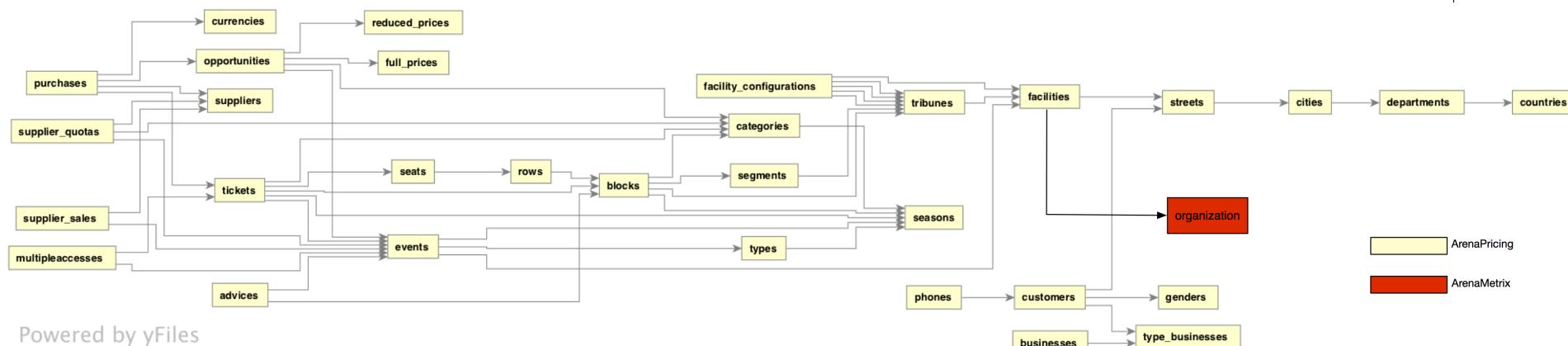


FIGURE A.1 – La base de données ArenaPricing

A.2 Base de données ArenaMetrix

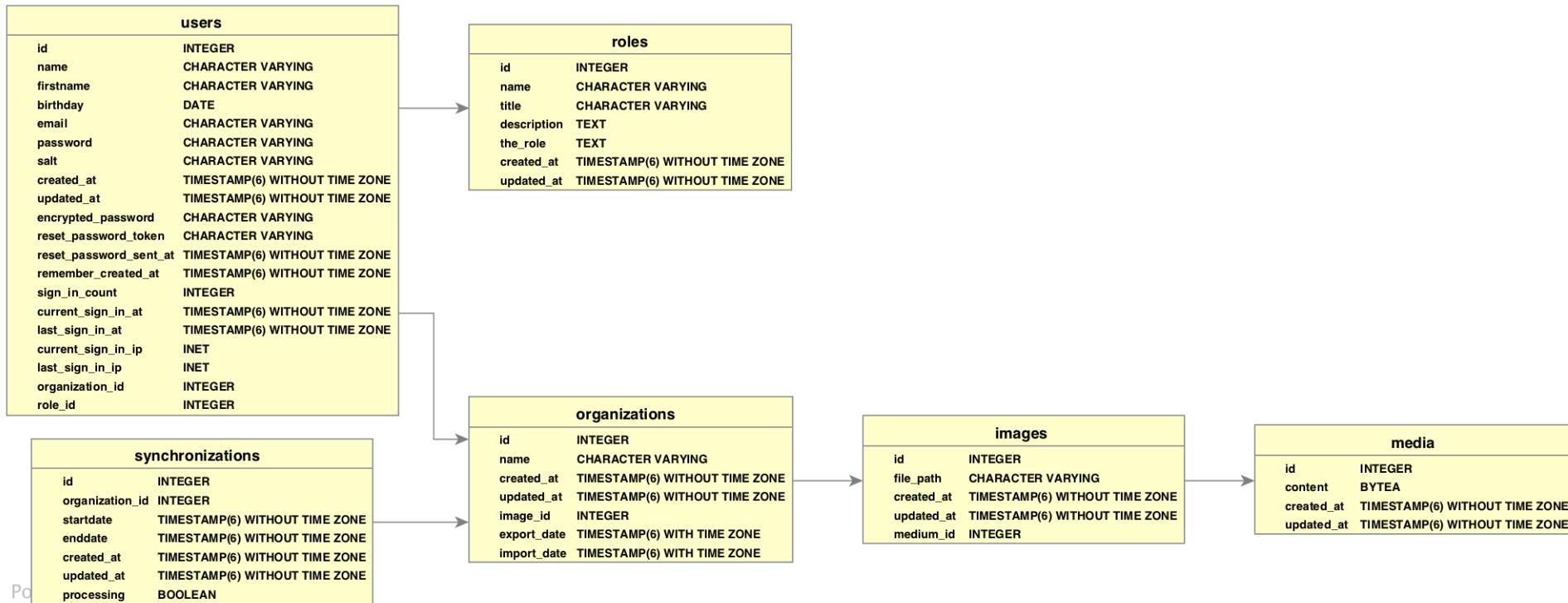


FIGURE A.2 – La base de données ArenaMetrix

A.2. BASE DE DONNÉES ARENAMETRIX

Bibliographie

- [1] Rudi Bruchez. *Les bases de données NoSQL : Comprendre et mettre en oeuvre.* Eyrolles, 2013.
- [2] Brian K. Jones David Beazley. *Python Cookbook, 3rd Edition.* O'Reilly Media, 2013.
- [3] François de Sainte Marie. Bases de données relationnelles et normalisation : de la première à la sixième forme normale.
- [4] Daniel Kehoe. *Learn Ruby on Rails,* 2014.
- [5] Leo S. Hsu Regina O. Obe. *PostgreSQL : Up and Running, 2nd Edition.* O'Reilly Media, 2015.

Mots clefs

Etudes et Conseils – Informatique – Base de données – Logiciel - d'analyse de données

LARDIES Ludovic

Rapport de stage ST50 – P2015

Résumé

Tech4Team a pour but de développer un produit encore inédit sur le marché européen de l'événementiel : le Yield Management.

La promesse est d'augmenter de 5 à 10% le chiffre d'affaires de n'importe quel événement sportif ou culturel.

Tech4Team développe un logiciel permettant de calculer le prix optimal pour chacune des catégories de place en temps réel pour un événement sportif et culturel. Pour cela, l'entreprise exploite les bases clients des organisateurs de spectacle et transforme leurs données en informations utilisables directement pour optimiser leur politique billetterie.

J'ai donc été engagé afin d'intégrer les bases clients brutes à notre base de données interne. J'ai également fait un travail de mise en forme des données et d'agrégation afin de permettre au statisticien d'exploiter les données et de les intégrer directement à leurs modèles mathématiques.

Niveau technologie j'ai travaillé avec du Python pour la récupération et l'agrégation des données. Du Ruby et plus précisément avec le framework Ruby on Rails pour l'insertion des données en base de données. Côté base de données nous utilisons PostgreSQL en base de données relationnelle pour stocker les données clients et Riak en base de données non relationnelle pour stocker les données agrégées.

Tech'4'Team

152 Bd MacDonald

75019 Paris

<http://tech4team.com>