

RAPPORT

Automatisation du réseau AMAP

par Médéric HURIER et Omar EDDASSER



Table des matières

[1. Analyse](#)

[1.1 Expression du besoin](#)

[1.1.1 Mise en relation](#)

[1.1.2 Composition de panier](#)

[1.1.3 Historique](#)

[1.2 Critique du système existant](#)

[1.2.1 Ce que marche](#)

[1.2.2 Ce qui manque](#)

[1.2.3 Évolutivité](#)

[1.3 Solution proposée](#)

[1.3.1 Automatisation des processus](#)

[1.3.2 Technicité](#)

[1.3.3 Résumé](#)

[2. Conception](#)

[2.1 Conceptuel](#)

[2.1.1 Modèle Conceptuel de Données](#)

[2.1.2 Modèles Conceptuels de Traitement](#)

[2.2 Logique](#)

[2.2.1 Modèles Organisationnels des Traitements primaires](#)

[2.2.2 Modèles Organisationnels des Traitements secondaires](#)

[2.3 Physique](#)

[2.3.1 Modèle Physique de Données](#)

[2.3.2 Modèles Physiques de Traitement](#)

[3. Réalisation](#)

[3.1 Choix d'implantation](#)

[3.2 Choix techniques](#)

[3.3 Différence entre la conception et la réalisation](#)

[4. Conclusion](#)

1. Analyse

1.1 Expression du besoin

1.1.1 Mise en relation

Une AMAP est une **association** qui favorise les **échanges sans intermédiaires** entre agriculteurs et groupes de consommateur. Par son action de **proximité**, elle soutient ses partenaires dans une démarche **durable et responsable**.

Pour assurer ses engagements, le système d'information de l'association comprend la gestion et la communication des informations suivantes:

- la promotion de l'agriculture paysanne auprès des consommateurs
- la prise de contact entre le client et l'AMAP la plus proche
- la collecte des propositions auprès des fournisseurs
- la gestion des abonnements et des saisons
- la coopération entre AMAP à échelle nationale

A l'ère des échanges numériques, l'association cherche à automatiser ses échanges pour améliorer sa **fiabilité** et sa **réactivité**. Le système doit permettre des interactions plus **simples** et plus **directes** entre les acteurs.

1.1.2 Composition de panier

La mission principale de l'association est de **composer et de distribuer chaque semaine des paniers auprès de ses abonnés**.

Comme pour une entreprise classique, ce mode opératoire comprend **la gestion des clients et des fournisseurs** ainsi que des **processus de livraison et de commande**. Le système doit assurer **le suivi de ses éléments et des événements associés**.

Afin de varier son service, l'association peut **compléter ses paniers avec des produits d'autres AMAP**. Pour cela, elle contacte son responsable et passe commande en fonction des propositions de ses fournisseurs.

1.1.3 Historique

La pérennité d'une AMAP dépend de la **rentabilité de ses activités**. Mais contrairement à une entreprise, ses membres sont bénévoles qui ne cherchent pas à dégager de bénéfices. Ils doivent estimer au mieux le prix de revient des paniers pour **fixer un prix de vente le plus juste possible**.

Tous les flux et les informations doivent être conservés par le système pour **fournir des statistiques aux gestionnaires**. A court terme, l'association doit savoir si le panier de la semaine est proche du prix moyen de la saison pour adapter ses futures compositions. A long terme, l'AMAP doit savoir si ses saisons ont satisfait aussi bien ses clients que ses fournisseurs pour adapter le prix moyen d'un panier.

1.2 Critique du système existant

1.2.1 Ce qui marche

Le mode de fonctionnement de l'AMAP est **un processus qui plaît** aussi bien aux producteurs qu'aux consommateurs. L'audit a révélé deux éléments intéressants dans les traitements actuels.

Le premier élément concerne **la gestion hebdomadaire et mensuelle des périodes**. A l'échelle d'une semaine, les fournisseurs et les clients peuvent s'organiser pour livrer et récupérer les produits. Cela laisse le temps à tout le monde de se préparer.

Et à l'échelle d'un mois ou d'une saison, les clients payent le service de l'AMAP en ayant la garanti que le prix du panier reste le même. Quant aux fournisseurs, le système de facturation au mois évite les soucis de trésorerie.

Le second élément est **la logistique de l'association**. Les interviews ont révélé que **les adhérents sont très impliqués** et assurent correctement la composition et la livraison des paniers. Ils confirment qu'il n'est pas nécessaire pour le système de faire le suivi de ce processus. De plus, il n'y a pas de problème de livraison avec les fournisseurs, donc pas de gestion de stock. Les paniers non récupérés sont recyclés par divers moyens.

Au niveau national, **la coopération entre systèmes des AMAP** n'est pas encore implémentée, mais les gestionnaire communiquent déjà entre eux pour s'échanger des produits. Ils insistent pour passer directement par l'AMAP et non par ses fournisseurs.

1.2.2 Ce qui manque

Les adhérents de l'association interviennent à toutes les étapes du fonctionnement de l'AMAP, aussi bien dans la gestion des abonnements que des commandes auprès des fournisseurs. Si aucun membre ne peut répondre, ou si l'information est mal remontée, certains produits ou paniers peuvent être oubliés.

Pour améliorer cette gestion, les adhérents et les fournisseurs doivent avoir **une plus grande autonomie** mais **toujours sous contrôle** des membres de l'association. Il faut une **interface** au système d'information capable de **gérer des rôles** pour les différents acteurs.

Malgré les connaissances en feuille de calcul des membres de l'association, l'enchaînement des opérations se représente difficilement dans un tel système et beaucoup de mise à jour sont encore manuelle. Il est nécessaire d'**optimiser les traitements** pour **réduire le nombre d'interaction** avec le système d'information.

1.2.3 Évolutivité

L'audit a confirmé que **les possibilités d'évolutions du système actuel sont très limitées**. La programmabilité des feuilles de calculs ne permet pas d'assurer tous les contrôles et de laisser aux utilisateurs l'autonomie qu'ils réclament.

Pour avancer dans le processus d'automatisation, il faut partir d'**une nouvelle base** en conservant les processus qui fonctionnent et en répondant aux manques applicatifs. Le nouveau système doit tenir compte des exigences utilisateurs, à savoir **une grande liberté d'action** et une **simplicité** qui **facilite la gestion**.

Les données système actuel peut être **facilement reprises** pour alimenter la nouvelle application. Les possibilités d'export des feuilles de calculs sont nombreuses, mais demanderont un traitement post pour assainir les informations stockées.

1.3 Solution proposée

1.3.1 Automatisation des processus

Pour répondre aux attentes du projet, nous vous proposons de réaliser **une application web** accessible via **divers interfaces** par les acteurs du système (client, fournisseur et membre de l'association). Chaque interface, ou module, pourra être réalisé **séparément** mais utilisera une **base commune** pour les traitements.

Voici le découpage que nous envisageons:

- **l'interface producteur** réservée aux fournisseurs afin qu'ils proposent leurs produits
- **l'interface client** qui permet à ses derniers de s'inscrire et de commander des paniers occasionnellement
- **l'interface de composition de panier**, qui comme son nom l'indique servira à concevoir le panier de la semaine et à passer commande aux fournisseurs
- **l'interface d'administration** qui permettra de réaliser le suivi de tous les éléments, dont les commandes et les abonnements. Elle reprendra l'interface précédente, mais accessible uniquement si le membre dispose des permissions nécessaires.

Dans une seconde version, nous pourrions envisager d'ajouter ou de compléter les interfaces avec des sites vitrines pour faire la promotion de l'activité de l'association. Nous imaginons également un site national pour toutes les AMAP avec un système de forum pour faciliter leur collaboration.

Nous mettons en avant l'avantage de cette solution, qui est **totalelement évolutive** et permet la gestion de tous les anciens/nouveaux traitements **avec les contrôles et la liberté nécessaires**.

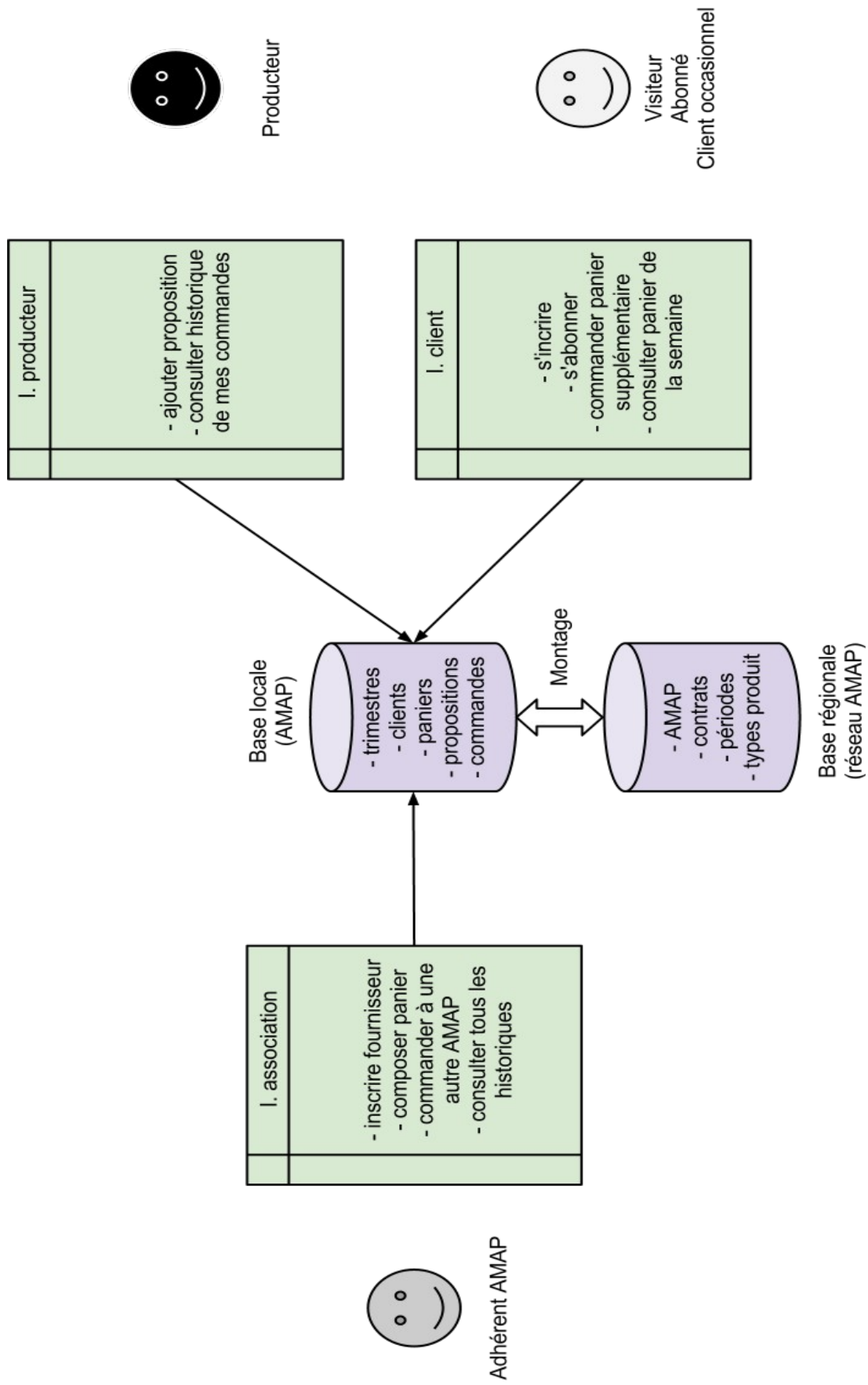
1.3.2 Technicité

L'interview auprès des administrateurs du système actuel a montré que ces derniers étaient capable d'utiliser le système de base de données **PostgreSQL**. Nous comptons nous orienter vers une base applicative en **Django/Python**.

Au niveau de l'organisation des bases, nous comptons mettre en oeuvre 2 bases. L'une sera réservé aux informations locales de l'AMAP et l'autre aux informations globales (nationale) de toute les AMAP.

1.3.3 Résumé

Le diagramme suivant résume notre vision du futur système.



2. Conception

2.1 Conceptuel

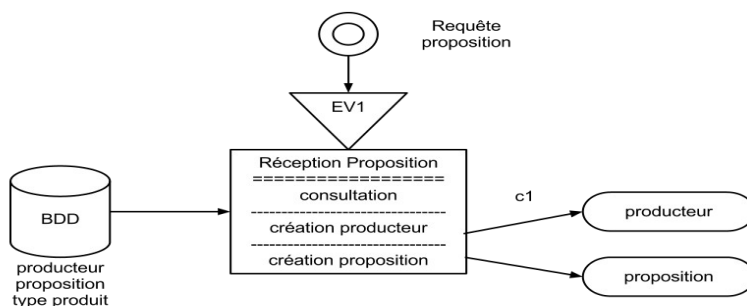
2.1.1 Modèle Conceptuel de Données



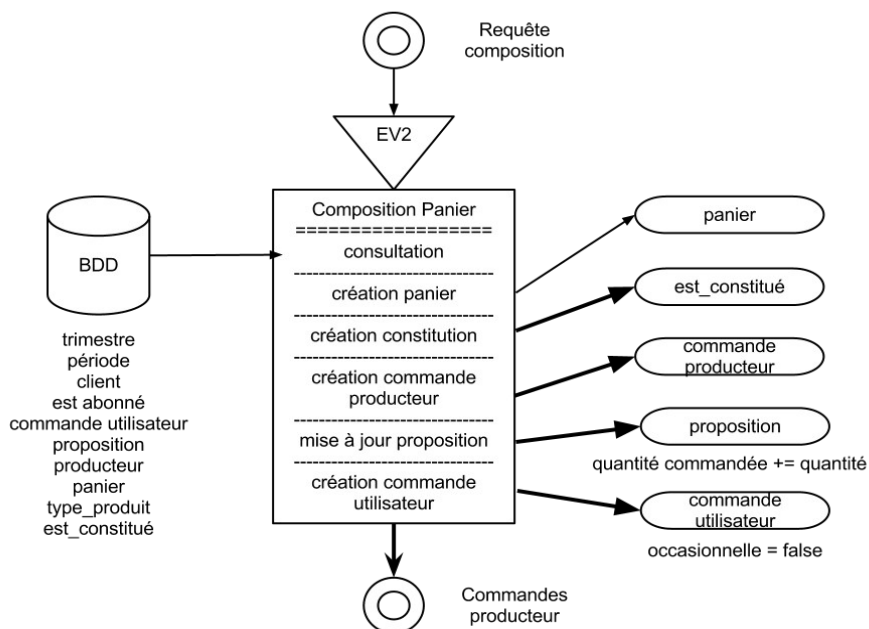
Notre MCD n'a pas beaucoup été modifié par rapport à sa version initial. En effet, hormis l'ajout de certains attributs "secondaire" (tel que "ajoute_le" pour stocker la date de création par exemple) seule la table "CONTRAT" (qui devait initialement faire le lien entre les producteurs et l'AMAP dans laquelle ils sont) a été supprimé car jugée inutile.

2.1.2 Modèles Conceptuels de Traitement

Ci-dessous le diagramme modélisant la réception d'une proposition d'un producteur :

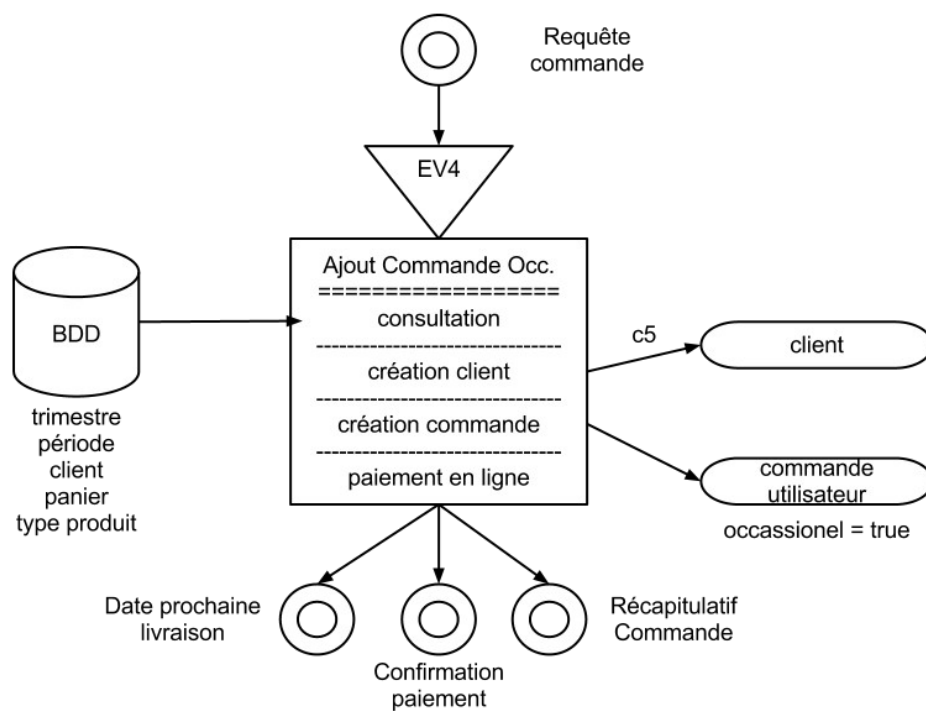


c1: le producteur n'existe pas



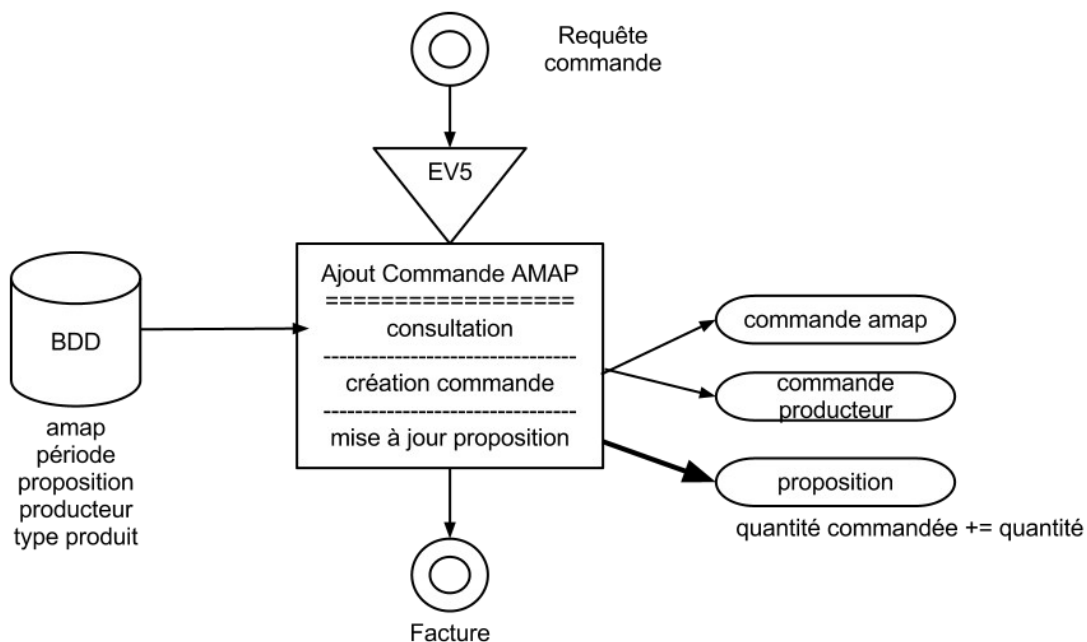
Nous avons choisi pour la modélisation de la composition du panier (comme ci-dessus) d'effectuer toutes les opérations (commande producteur, mise à jour des quantités commandées des propositions et création des commandes des abonnés) s'y rapportant en une fois dans un but de simplification.

Ci-dessous la modélisation de la création d'un abonnement :

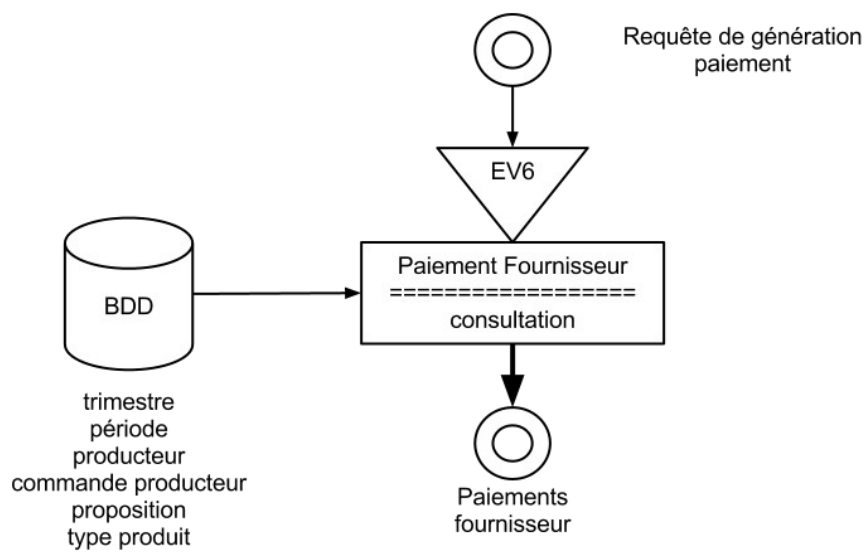


c5: le client n'existe pas

L'ajout d'une commande occasionnelle:



La modélisation du paiement des fournisseurs s'effectue en une étape, comme c'est schématisé ci-dessous :

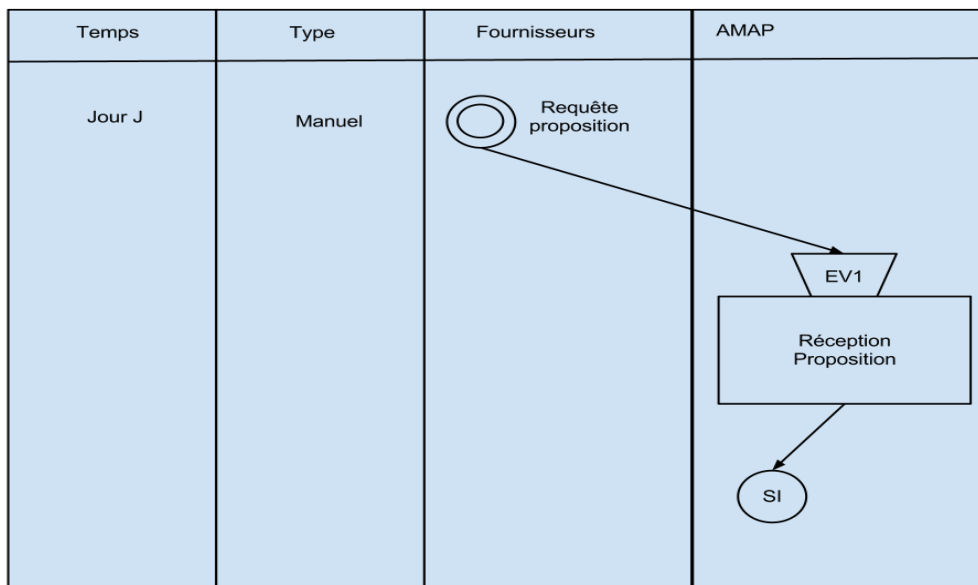


2.2 Logique

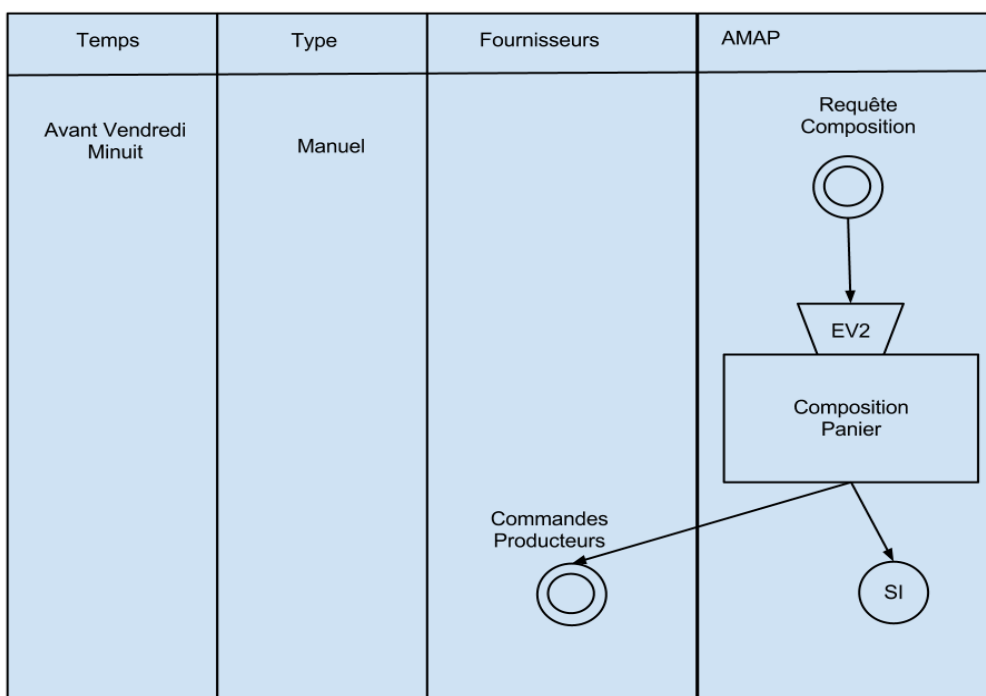
Tout comme pour notre MCD et nos MCT, les MOT qui suivent n'ont pas évolué depuis la première version.

2.2.1 Modèles Organisationnels des Traitements primaires

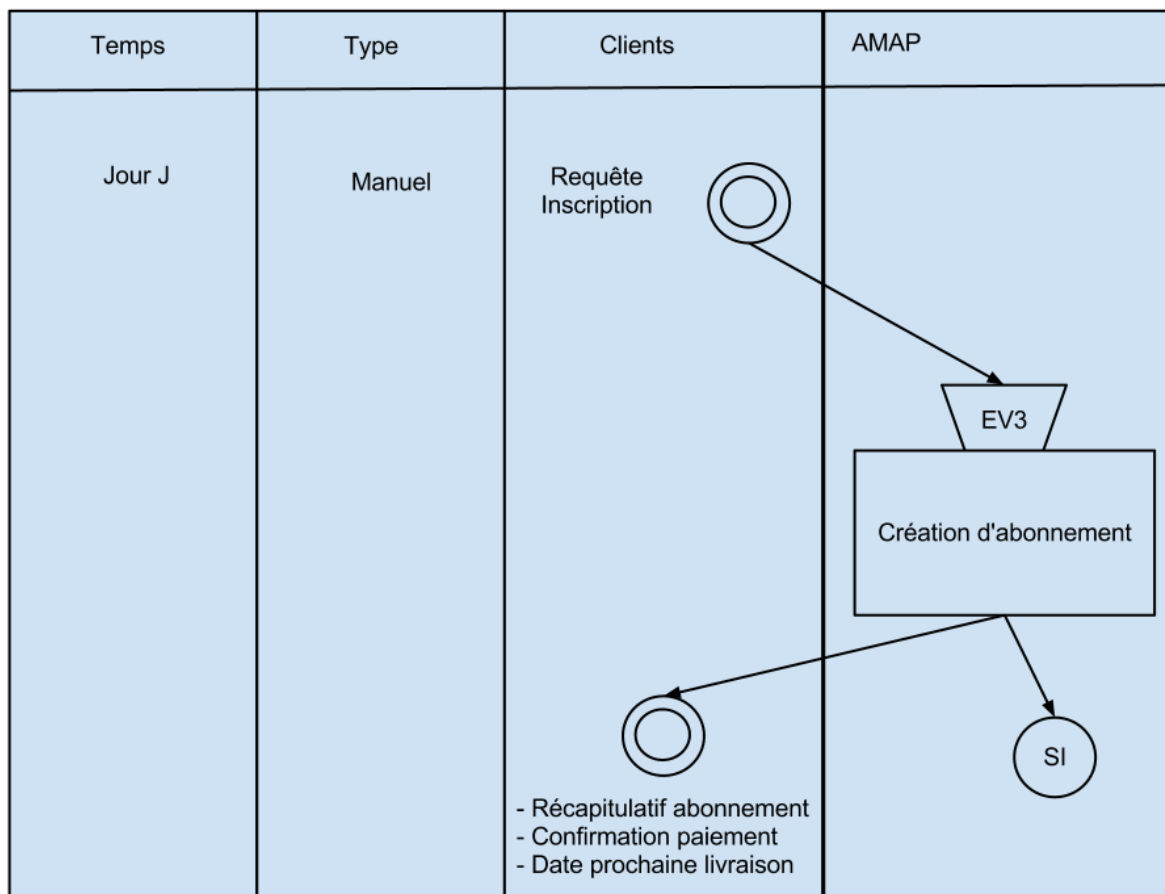
Modélisation de la réception d'une proposition



Modélisation de la composition du panier :

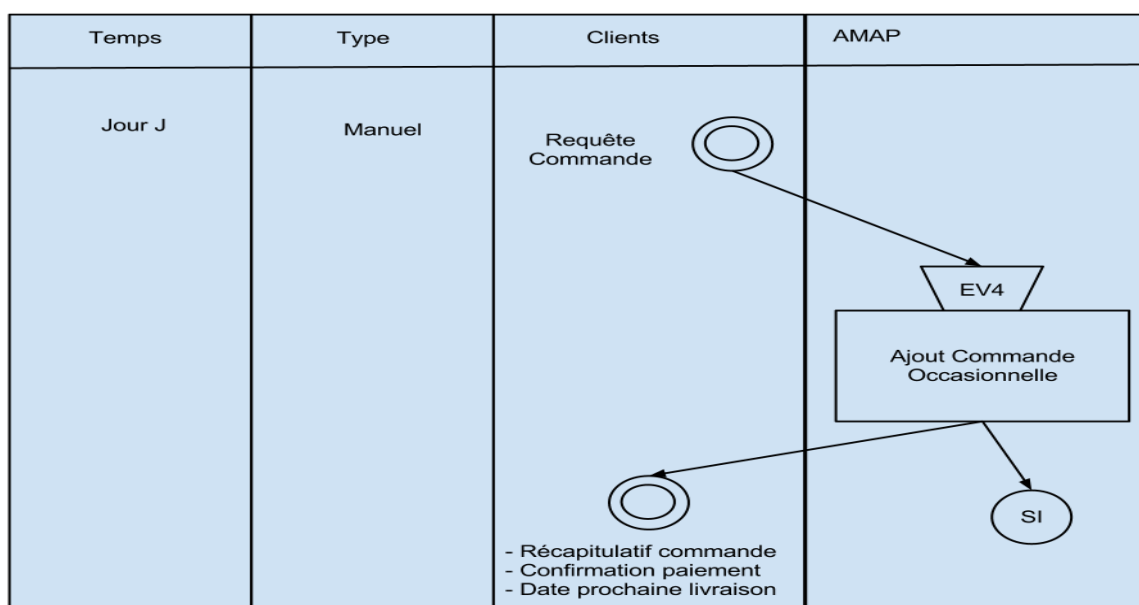


Modélisation de la création d'abonnement :

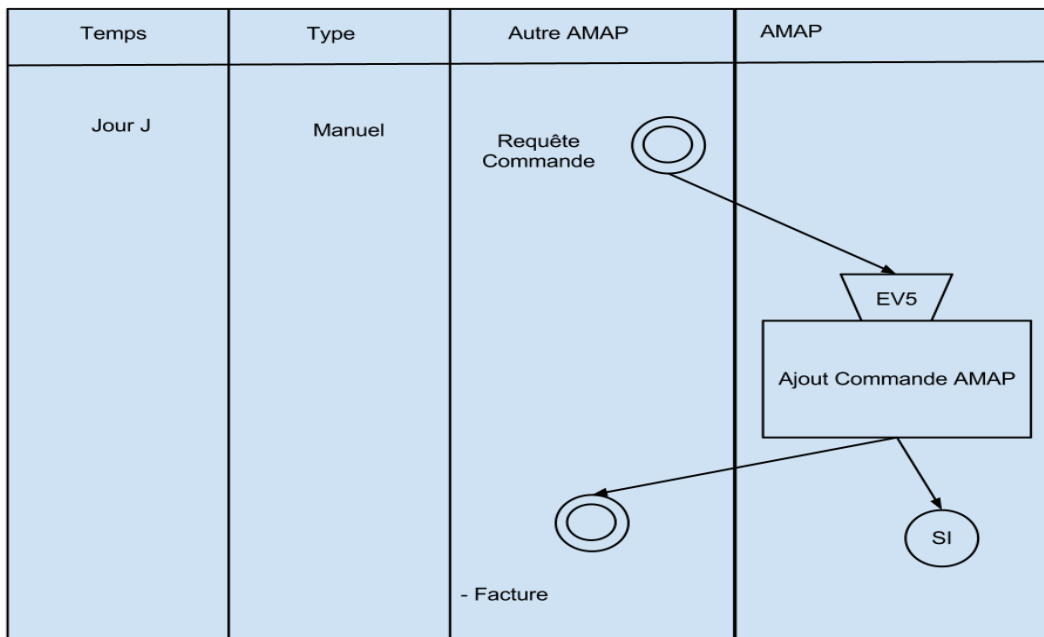


2.2.2 Modèles Organisationnels des Traitements secondaires

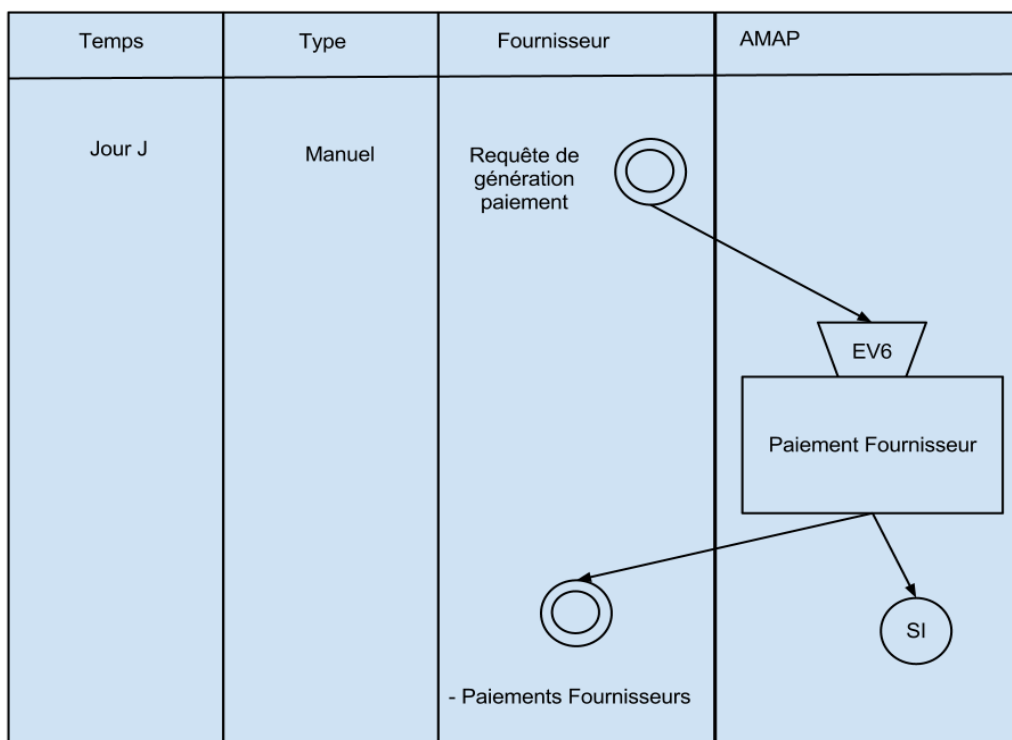
Modélisation de l'ajout d'une commande occasionnelle :



Modélisation de la commande faite par une autre AMAP

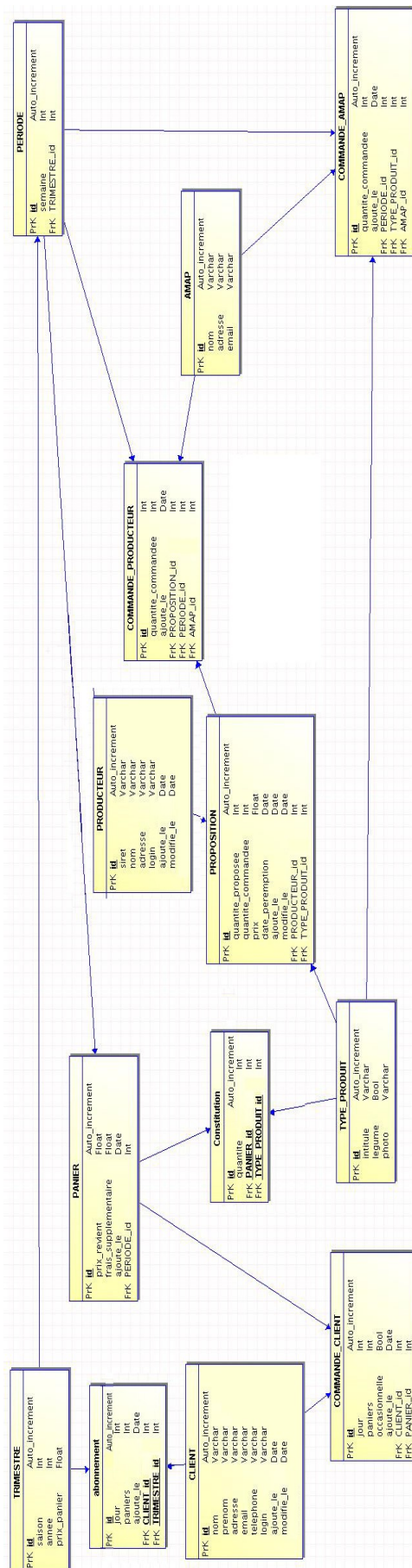


Modélisation du paiement des fournisseurs :



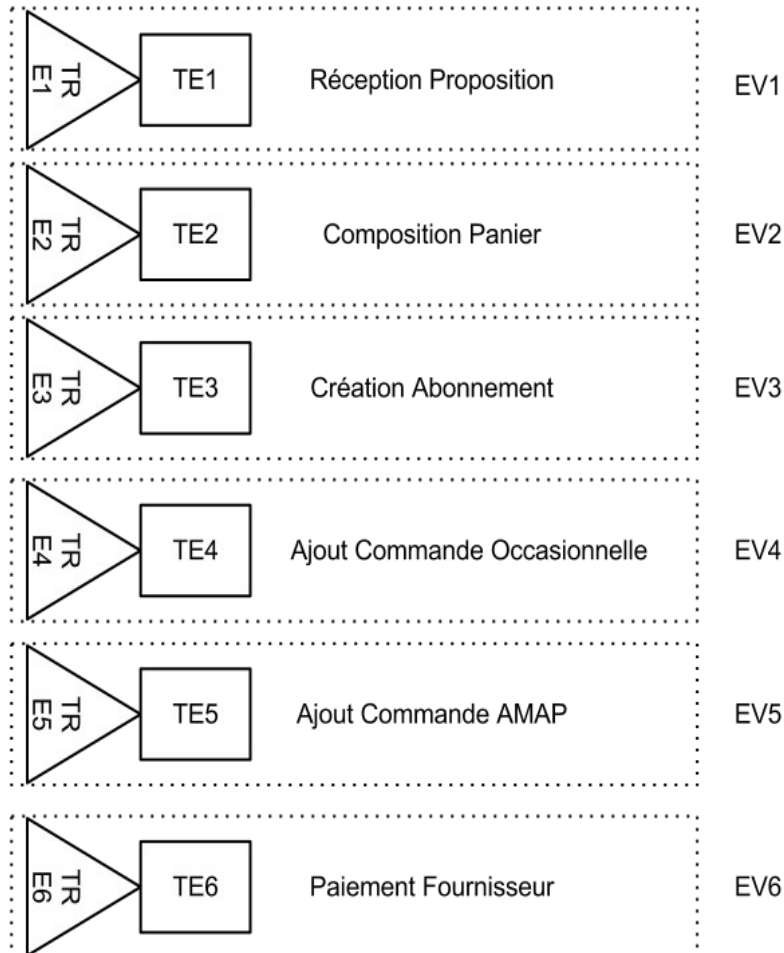
2.3 Physique

2.3.1 Modèle Physique de Données



2.3.2 Modèles Physiques de Traitement

Pour des raisons de simplifications, nous avons choisi de traiter tout les événements en une seule fois (de la manière la plus simple possible), c'est-à-dire en évitant les enchaînements comme le montrent les schémas ci-dessous :



3. Réalisation

3.1 Choix d'implantation

3.1.1 Priorité des traitements

Nous avons séparé les traitements en **2 niveaux de priorités**.

Les traitements indispensables aux fonctionnements du système sont la **Réception des Propositions**, la **Création des Abonnements** et la **Composition du Panier**. Chacune de ces fonctionnalités correspond à une interface utilisateur propre à un acteur du système (client, fournisseur, gestionnaire).

Les traitements secondaires sont l'**Ajout de Commandes Occasionnelles**, l'**Ajout de Commandes AMAP** et le **Païement des Fournisseurs**. Ils ne sont pas nécessaires au fonctionnement de l'AMAP ou peuvent être fait manuellement.

Compte-tenu des contraintes de temps, nous nous sommes concentrés sur la réalisation des traitements principaux pour cette première version. De plus, nous avons implémenté l'**Interface d'Administration** pour que les gestionnaires puissent alimenter la base de données et vérifier les contraintes d'intégrité du modèle.

3.1.2 Déploiement

Afin de vérifier que le réseau des AMAP puisse utiliser notre application, nous avons déployé le site à 2 adresses différentes:

- <http://freemap-main.freaxmind.pro/> : adresse du site principal
- <http://freemap-other.freaxmind.pro/> : adresse du site secondaire

Chaque site comprend une base de données locale avec ses clients, ses fournisseurs et ses commandes. Ils utilisent également une base de données commune contenant un annuaire des types de produits et des AMAP pour faciliter la communication entre les associations.

La base commune correspond la base locale du site principale. C'est un modèle centralisé qui permet au responsable de cette AMAP de gérer les types de produit et les informations de contact des autres AMAP.

Le site secondaire s'y connecte par le réseau et n'a le droit que de lire le contenu de cette base.

Les bases de données sont administrables par une interface en ligne disponible à cette adresse:

- <https://phppgadmin.alwaysdata.com/>

Chaque site dispose de sa propre interface d'administration:

- <http://freemap-main.freaxmind.pro/administration/> : interface d'administration du site principal
- <http://freemap-main.freaxmind.pro/administration/> : interface d'administration du site secondaire

Les logins et mots de passe de ses interfaces se trouvent dans le fichier docs/README.txt de l'archive du projet avec des captures d'écran.

3.2 Choix techniques

3.2.1 Framework applicatif

Les applications que nous avons réalisés sont des **interfaces web** basées sur le **framework Django**. C'est le cadriciel **Python** le plus connu du marché, et qui propulse entre autre:

- [The Washington Post](#)
- [The Guardian](#)
- [Lawrence.com](#)
- [Disqus](#)
- [Pinterest](#)
- [Instagram](#)
- [Mozilla](#)

Pour la partie client, nous avons utilisé **HTML5, CSS3, JQuery et JQuery Mobile**. Cette combinaison permet de réaliser des interfaces portables à la fois sur le bureau et sur les systèmes embarqués.

Enfin, nous utilisons **Mercurial et Redmine** pour synchroniser le code entre les développeurs. **Mercurial** est un **Système de Contrôle de Version (VCS)** similaire à Subversion mais fonctionnant sur un modèle décentralisé. **Redmine** est une interface de **gestionnaire de projet** offrant une vue du dépôt de code et permettant de stocker des documents.

3.2.2 Base de données

Pour palier aux problèmes de connexion rencontrés au lancement du projet, nous avons opté pour une autre base de données que Sybase. Nous utilisons **PostgreSQL**, l'une des bases open-source les plus performantes du marché.

Comme pour Sybase, elle supporte la **gestion avancée des droits, des triggers et des procédures stockées**. Elle sont hébergées sur l'hébergeur de l'auto-entreprise de Médéric Hurier : **AlwaysData**.

Nous avons implémenté trois triggers :

- Un premier qui se déclenche lors de l'ajout d'un nouveau trimestre, celui-ci créé automatiquement toutes les périodes associées au trimestre
- Un second qui se déclenche avant l'ajout d'une commande producteur, il s'assure que :
 - la proposition existe bien,
 - la quantité (de produit) disponible est supérieure à la quantité que l'on souhaite commanderet met à jour la quantité commandé de la proposition
- Et un dernier qui est déclenché lors de l'ajout d'une période et créé un panier vide associé à celle-ci.

3.3 Différence entre la conception et la réalisation

3.3.1 Gestion des logins

Lors de la conception, nous avons prévu que chaque acteur du système (client, fournisseur, gestionnaire) soit représentée par **une table comprenant un login et un mot de passe**. Cette décomposition nous semblait évidente, il suffisait d'associer un gestionnaire d'authentification par interface utilisateur.

Arrivé à l'étape d'implémentation, nous avons réalisé qu'**un système basé sur des ACL (gestion de permissions) serait beaucoup plus efficace et plus rapide** à mettre en place. Ainsi, nous nous sommes basés sur **le module d'authentification de Django** qui gère les utilisateurs, les groupes et les permissions.

La capture suivante montre un exemple du module. Il est accessible en se rendant dans l'interface d'administration de l'application et en cliquant sur le lien "Utilisateurs" ou "Groupe" (voir partie 3.1.2 pour les détails de connexion):

Modification de utilisateur | Site d'administration de Django - Mozilla Firefox

Firefox MCT - Google Drive Rapport - Automatisation... Modification de utilisateur...

freemap-main.freaxmind.pro/administration/auth/user/1/

Information personnelle

Prénom:

Nom:

Adresse électronique:

Permissions

☒ Actif
Précise si l'utilisateur doit être considéré comme actif. Décochez ceci plutôt que de supprimer le compte.

☒ Statut équipe
Précise si l'utilisateur peut se connecter à ce site d'administration.

☒ Statut super-utilisateur
Précise que l'utilisateur possède toutes les permissions sans les assigner explicitement.

Groupes:

clients
fournisseurs
adhérents

 +

Les groupes dont fait partie cet utilisateur. Celui-ci obtient tous les droits de tous ses groupes. Maintenez appuyé « Ctrl », ou « Commande (touche pomme) » sur un Mac, pour en sélectionner plusieurs.

Permissions de l'utilisateur:

Spécifiez permissions for this user. Maintenez appuyé « Ctrl », ou « Commande (touche pomme) » sur un Mac, pour en sélectionner plusieurs.

permissions de l'utilisateur disponible(s)

Filter

gestion	Produit du panier	Can add Produit du panier
gestion	Produit du panier	Can change Produit du panier
gestion	Produit du panier	Can delete Produit du panier
gestion	Panier	Can add Panier
gestion	Panier	Can change Panier
gestion	Panier	Can delete Panier
gestion	Période	Can add Période
gestion	Période	Can change Période
gestion	Période	Can delete Période
gestion	Producteur	Can add Producteur
gestion	Producteur	Can change Producteur
gestion	Producteur	Can delete Producteur
gestion	Proposition	Can add Proposition
gestion	Proposition	Can change Proposition
gestion	Proposition	Can delete Proposition

Choix des « permissions de l'utilisateur »

IGNORE ALL KEYS...

3.3.2 Gestion des clés étrangères sur la base de données commune

La gestion de deux bases de données dans une même application pose plusieurs contraintes que nous n'avons pas anticipé lors de la réalisation.

Les clés étrangères entre les tables de la base externe et les tables de la base locale ne sont pas supportées par le framework applicatif que nous avons choisi. C'est une limitation logique, car une base ne peut pas surveiller le changement d'état d'une clé d'une autre base et appliquées les contraintes qui y sont rattachés (exemple: suppression en cascade).

Pour combler ce problème, **nous nous sommes simplement basés sur des simples champs de type entier sans contrainte de clé étrangère**. Malheureusement, cela nous a privé de beaucoup de fonctionnalités offertes par Django pour la gestion des relations. Nous avons du

coder nous même plusieurs alternatives pour retrouver le fonctionnement initial auquel nous avons pensé.

4. Conclusion

4.1 Captures d'écran de l'application

INTERFACE D'ADMINISTRATION

Administration du site | FreeMAP - Interface d'administration - Mozilla Firefox

Firefox MCT - Google Drive Rapport - Automatisation... Administration du site | F...

127.0.0.1:8000/administration/

FreeMAP - Interface d'administration Bienvenue, **mederic**. Modifier votre mot de passe / Déconnexion

Administration du site

Composer le panier de la semaine

Auth	
Groupes	Ajouter Modifier
Utilisateurs	Ajouter Modifier

Gestion	
Abonnements	Ajouter Modifier
Clients	Ajouter Modifier
Commande AMAP	Ajouter Modifier
Commandes Client	Ajouter Modifier
Commandes Producteurs	Ajouter Modifier
Paniers	Ajouter Modifier
Producteurs	Ajouter Modifier
Produits du panier	Ajouter Modifier
Propositions	Ajouter Modifier
Trimestres	Ajouter Modifier
Types de Produits	Ajouter Modifier

Actions récentes

Mes actions

- [Panier de la semaine 52 en 2012](#)
Panier
- [3](#)
Commande Client
- [4](#)
Commande Client
- [1 Citrouille](#)
Produit du panier
- [2 Citron](#)
Produit du panier
- [3 Cerise](#)
Produit du panier
- [1](#)
Commande Producteur
- [2](#)
Commande Producteur
- [3](#)
Commande Producteur
- [4](#)
Commande Producteur

http://127.0.0.1:8000/administration/ <> [3/3] All

INTERFACE DE COMPOSITION

Composition du panier de la semaine N°51 | FreeMAP - Interface d'administration - Mozilla Firefox

Firefox MCT - Google Drive Rapport - Automatisation du ... Composition du panier de la ...

127.0.0.1:8000/administration/composition/

FreeMAP - Interface d'administration Bienvenue, mederic. Modifier votre mot de passe / Déconnexion


Accueil > Composition

Composition du panier de la semaine N°51


Produits à votre disposition

- Banane
- Carotte
- Fraise
- Patate
- Radis


En cas d'erreur: remettez les produits dans cette boîte.

 x 3


+

 x 5

+

 x 1

+

 x 1

+

...

- 345 Cerise à 10 € de Marc Bonhomme
- 500 Citron à 20 € de Jean Amoureux
- Il manque 75 de Citron !
- 10 Citrouille à 20 € de Marc Bonhomme
- 30 Citrouille à 25 € de Jean Amoureux
- 50 Citrouille à 30 € de Alfred Mousquetaire
- Il manque 25 de Citrouille !
- Il manque 115 de Laitue !

Total des produits	15900.00	€
Total des frais supplémentaires	<input type="text" value="200.00"/>	€
Nombre de paniers	115	
Prix de panier moyen (trimestre)	50,00	€
Prix de revient par panier	140.00	€
Prix de revient pour la période	16100.00	€

Valider le panier de la semaine

http://127.0.0.1:8000/administration/composition/ < [3/4] All

VALIDATION DE LA COMPOSITION

The screenshot shows a web browser window titled "FreeMAP - Interface d'administration - Mozilla Firefox". The address bar shows the URL "127.0.0.1:8000/administration/composition/valider". The page header includes the title "FreeMAP - Interface d'administration" and a welcome message "Bienvenue, mederic. Modifier votre mot de passe / Déconnexion". The breadcrumb trail is "Accueil > Validation".

The main content area displays the title "Panier de la semaine composé avec succès !" followed by a table of statistics:

Nombre de paniers	115
Prix moyen (trimestre)	50,00 €
Prix de revient	140,00 €
Frais supplémentaires	200,00 €
Produits manquants	25 Citrouille 115 Laitue 75 Citron

Below the table, the text "Rendez-vous à la semaine prochaine !" is displayed.

The footer of the page shows the URL "http://127.0.0.1:8000/administration/composition/valider" and pagination information "< [3/4] All".

4.2 Retour d'expérience

Notre bilan du projet est sûrement très différent de celui des autres groupes car nous n'avons pas utilisé les mêmes technologies (Django/Python pour le code, PostgreSQL pour la base de données). Cependant, cet originalité a été l'occasion d'essayer d'implémenter une solution contraignante par d'autres moyens que ceux du cahier des charges.

PostgreSQL est une base simple et très performante. Elle a l'avantage d'être open-source et d'avoir une très bonne documentation ainsi qu'une communauté très présente. Comparé à Sybase qui est une base propriétaire et dont la documentation laisse souvent à désirer, elle offre des fonctionnalités similaires de celles que nous avons vu en cours. Nous n'avons pas pâti de ce choix, qui nous a au contraire offert une appréciable autonomie.

De même que pour Django, le choix d'une technologie innovante c'est révélé être un catalyseur pour notre créativité et notre productivité. Le crédo du framework est: "The Web framework for perfectionists with deadlines" ce qui correspondait parfaitement à notre situation.

La démarche intégratrice que nous avons utilisé tout au long du projet s'est révélé être assez mal adaptée pour un projet de cette taille. Les traitements n'étaient pas assez dépendants les uns des autres pour nécessiter l'emploi de cette technique efficace surtout en cas de complexité forte du système. Comparé à UML, nous avons trouvé que les diagrammes étaient moins expressifs pour décrire aussi bien le besoin que l'implémentation.

Toutefois, ce projet a été très instructif en terme de nouvelles technologies. Nous avons pu bénéficier des cours de CSI et de SGBD avancé pour comprendre les outils que nous utilisons, et nous regrettons seulement de ne pas avoir eu assez de temps pour explorer plus en détails les choix d'implémentations et techniques de ce projet.