	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

### Document présentation


<b>Description</b>	Spécifications détaillées
--------------------	---------------------------

### Document certification

	<i>Name</i>	<i>Fonction</i>	<i>Date livraison</i>
<b>Author</b>	CRASKE Antoine DJEBIEN Tarik STIENNE Rudy MASCOT Mélody RAKOTOBE Sitraka Eric	Chef de projet Responsable technique Responsable fonctionnel Responsable documentation Responsable Qualité	
<i>Decidor</i>	STIENNE Rudy	Responsable fonctionnel	

### Document Version history

<i>Version</i>	<i>Author</i>	<i>Date</i>	<i>Comments</i>
1.0	Antoine Craske	19 jan 2013	Creation

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

## Table des matières

[Table des matières](#)

[Objectif du document](#)

[V1 : Gestion des langues](#)

[IHM](#)

[Service](#)

[Persistence](#)

[Jeux de tests](#)

[V1 : Login](#)

[IHM](#)

[Service](#)

[Persistence](#)

[Jeux de tests](#)

[V1 : Création de compte client](#)

[IHM](#)

[Service](#)

[Persistence](#)

[Jeux de tests](#)

[V3 : Création de compte partenaires/admin](#)

[IHM](#)

[Service](#)

[Persistence](#)

[Jeux de tests](#)

[V1 : Contact par mail](#)


[IHM](#)

[Service](#)

[Persistence](#)

[Jeux de tests](#)

[V1 : Réservation](#)

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

[IHM](#)

[Service](#)

[Persistence](#)

[Jeux de tests](#)

[Versions incrémentales du développement](#)

[V1 : Annulation de réservation](#)

[IHM](#)

[Service](#)

[Persistence](#)

[Jeux de tests](#)

[V1 : Post de commentaires page d'accueil](#)

[IHM](#)

[Service](#)

[Persistence](#)

[Tests](#)

[V1 : Post de commentaires sur services](#)

[IHM](#)

[Service](#)

[Persistence](#)

[Tests](#)

[V2 : Sport / Administrer offre](#)

[IHM](#)

[Service](#)

[Persistence](#)

[Tests](#)

[V2 : Sport / Edition matériel](#)

[IHM](#)

[Service](#)

[Persistence](#)


[Tests](#)

[V2: Sport / Remise matériel](#)

[IHM](#)

[Choix du booking](#)

[Rendu pour un booking](#)

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

[Service](#)

[Persistence](#)

[Tests](#)

[V2 : Remontée mécanique/ Administrer forfait](#)

[IHM](#)

[Service](#)

[Persistence](#)

[Jeux de données](#)

[Tests](#)

[V2 : Remontée mécanique/ Edition des forfaits](#)

[IHM](#)

[Service](#)

[Persistence](#)

[Jeux de données](#)

[Tests](#)

[V3 : Modération des avis](#)

[IHM](#)

[Service](#)

[Persistence](#)

[Jeux de données](#)

[Tests](#)

[V3 : Promotions](#)

[IHM](#)

[Service](#)

[Persistence](#)

[Jeux de données](#)

[Tests](#)

[V3 : Purge](#)

[IHM](#)


[Service](#)

[Persistence](#)

[V3 : Moteur de recherche et fiche client](#)

[IHM](#)

[Service](#)

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

[Persistence](#)

[Tests](#)

[V3 : Statistiques](#)

[IHM](#)

[Service](#)

[Persistence](#)

[Modèle de classe](#)

[Arborescence des pages](#)

[Pistes d'améliorations](#)

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

## Objectif du document

L'objectif de ce document est de permettre le développement des fonctionnalités du projet Proxy-Station en clarifiant les maquettes écrans, les services à développer et les impacts sur le modèle de classes.

Chaque fonctionnalité doit au minimum décrire les impacts IHM, service et persistance. Tout autre diagramme UML ou schéma complémentaire peut être fourni pour les fonctionnalités en ayant besoin.

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

## V1 : Gestion des langues

La langue par défaut est l'anglais.

### IHM

Il faut développer la page du choix de la langue : <http://<server>/proxystat/lang>

Cette page doit afficher un drapeau anglais et un français.



### Service

La gestion des chargements des fichiers de traduction est géré par les frameworks techniques sans service Java à développer.

### Persistence

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

La gestion des langues est géré dans des fichiers de propriétés non stockés en base de données. On aura un fichier sans suffixe qui contiendra les clefs/valeurs en anglais et un fichier suffixé par \_fr pour les clefs/valeurs en français.

Afin de simplifier la maintenance, on utilise uniquement 1 fichier par langue avec les clefs respectant le standard suivant :

- Messages sur toutes les pages en statique (header, veuillez entrer ...) : display.\*
- Messages d'aides : help.\*
- Messages d'exceptions : exception.\*
- Messages de notifications : info.\*

## Jeux de tests

Pour rappel, les jeux de tests ainsi que le code doit être en anglais.

Ici uniquement des tests d'IHM doivent être réalisés :

- Anglais par défaut
  - testEnglishIsDefaultLanguage()
  - Ouvrir l'url sans suffixe
  - Valider que le bouton est "Sign in" (pas "S'authentifier")
- Choix français
  - testFrenchDisplayedOnLanguageChoice()
  - Ouvrir l'url de choix de langue
  - Choisir français
  - Valider que le bouton est "S'authentifier"
- Arrivée sur page d'accueil après choix langue :
  - testArriveOnHomePageAfterLanguageChoice()
  - Ouvrir l'url de choix de langue
  - Choisir anglais
  - Valider l'url ".../index" ou le nom de la page "Index"/"Homepage"
- Pouvoir revenir sur la page de choix de langue
  - Ouvrir la page index
  - Cliquer sur le lien retour choix langue
  - Valider par url ou nom de la page



	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

## V1 : Login

La page de choix de langue et la page d'accueil sont les seuls pages accessibles sans être authentifié.

### IHM

Il faut développer la page index : `http://<server>/proxystat/index`.  
On doit afficher cette page si aucun suffixe d'URL n'est spécifié.

ProxyStat-Gestion header <a href="#">Language choice</a>	Login <input type="text"/> Password <input type="text"/>	<input type="button" value="Sign in"/>
ici on aura les offres de locations d'appartements		


En ayant cliqué sur “Sign in” avec un login et/ou un password valide, le bandeau de login doit être remplacé par un “Welcome <login>”.

En ayant cliqué sur “Sign in” avec un login et/ou un password invalide, il faut afficher un message d'erreur en rouge “Identifiants invalide”.

### Service

L'authentification est gérée par le framework Spring security en utilisant des fichiers de configuration, il n'y a donc pas de service Java à développer.

### Persistence

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

On utilisera la classe User et Role.

Pour la classe User les champs login/password sont utilisés pour valider l'authentification. La requête sera configuré dans le XML de Spring Security.

Sur la V1, on utilise la classe Role et seuls les users avec role "ROLE\_USER" seront authentifiés car les autres rôles amèneront sur des pages encore non développées.

## Jeux de tests

N'ayant pas de service, on effectuera uniquement des tests d'IHM.

- Pouvoir se connecter avec identifiants valide
  - testCanConnectWithCorrectCredentials()
  - Ouvrir page index
  - Renseigner user et password corrects
  - Cliquer sur bouton de connexion
  - Valider message connexion (sur l'id de l'element HTML, pas le texte)
- Pas de connexion avec identifiants invalide
  - testCannotConnectWithBadCredentials()
  - Ouvrir page index
  - Renseigner user et password incorrects
  - Cliquer sur bouton de connexion
  - Valider message erreur affiché (sur l'id, pas texte)

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

## V1 : Création de compte client

Cette page va permettre aux clients de créer un compte afin de pouvoir s'authentifier sur le site et ainsi accéder aux différentes offres proposées dans le but de réserver.

### IHM

Il faut développer la page de création de compte: <http://<server>/proxystat/register>.



**Création d'un compte**

Name

Surname

Email Address

Date of birth

Photo  [Télécharger](#)

Adresse

Password

Password Confirmation

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

Si l'on clique sur "validate" alors que tout les champs ne sont pas renseignés, un message apparait pour signifier que les champs non renseignés doivent être remplis.

Si l'on clique sur "validate" et que tout les champs sont renseignés, un message de validation apparait. Un mail est envoyé au client à l'adresse indiquée.

## Service

Création d'un service permettant de créer un compte en base de données si tout les éléments sont renseignés.

Le service devra vérifier :

- la validité du mail (du type [nom@nom.fr](mailto:nom@nom.fr))
- que le deuxième mot de passe est identique au premier.

/\*\*

\* Interface de validation de compte

\* @author

\* @version 1.0

\*/

interface IServiceValidateRegister {

/\*\*

\* Valide les informations d'un compte client

\* @param Email Email dont le format doit être validé

\* @return vrai(true) si les informations sont valides ou faux(false) si invalides

\*/

Boolean validateEmail (String Email);

/\*\*

\* Valide que les deux mots de passe sont identiques

\* @param password1 Premier mot de passe

\* @param password2 Mot de passe de vérification

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

\* @return vrai(true) si les deux mot de passe sont identiques ou faux(false) si différents  
\*/

Boolean compareTwoPassword (String password 1, String password2);

}

## Persistence

On utilisera les classes Customer, Address et User + Role.  
Tous les champs de ces classes seront utilisés. recherche

Création DAO IRegisterService()

## Jeux de tests

- Pouvoir s'inscrire avec tout les champs renseignés
  - testCanCreateAccountWithAllFieldsInformed()
  - Ouvrir page création de compte
  - Renseigner tout les champs correctement
  - Cliquer sur bouton "validate"
  - Test si le compte n'existe pas
- Ne pas pouvoir créer 2 compte avec les mêmes identifiants
  - testAccountdoesn'tExist()
  - Le client renseigne les champs nécessaires
  - Tous les champs sont correctement renseignés
  - Vérification en base que le compte n'existe pas déjà (même identifiants)
  - Valider message création affiché
  - Envoi d'un mail de validation
- Ne pas pouvoir s'inscrire avec des champs non renseignés
  - testCannotCreateAccountWithOneFieldMisinformed()
  - Ouvrir page création de compte
  - Ne pas renseigner tout les champs
  - Cliquer sur bouton "validate"

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0


- Valider message erreur affiché
- Vérifier que la DAO est bien appelée
  - testServiceValidateRegisterCallDAO() : valider que le service appelle le DAO
- Pouvoir se connecter avec un compte crée
  - unit dao : retrouve bien un compte existant
  - unit service : retrouve bien un compte existant
  - ihm : ouvrir homepage, se connecter, valider connexion avec welcome message

## V3 : Création de compte partenaires/admin

Cette page va permettre aux partenaires et admin de créer un compte afin de pouvoir créer, modifier ou supprimer les offres présentes sur le site et gérer au mieux leurs parties.

### IHM

Il faut développer la page de création de compte: <http://<server>/proxystat/partner.create> ou [create.partner](#)

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

Création d'un compte partenaire ou admin

Name :

Surname :

Job Title :

Type of account : ☐ Admin  
☐ Sport Partner  
☐ Mechanic Partner

Password :

Pass Partner or admin :

Si l'on clique sur "validate" alors que tout les champs ne sont pas renseigné et que le mot de passe n'est pas bon, un message apparait pour signifier que les champs non renseignés doivent être rempli et/ou que le mot de passe n'est pas bon.

Si l'on clique sur "validate" et que tout les champs sont renseignés et que le mot de passe est correct, un message de validation apparait. Un mail est envoyé au client à l'adresse indiquée.

## Service

Création d'un service permettant de créer un compte en base de données si tout les éléments sont renseignés.

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

## Persistence


On utilisera la classe Partner, User, Role

## Jeux de tests

- Pouvoir s'inscrire avec tout les champs renseignés
  - testCanCreateAccountWithAllFieldsInformed()
  - testCanCreateAccountWithCorrectPassword()
  - Ouvrir page création de compte
  - Renseigner tout les champs correctement et utiliser le bon mot de passe
  - Cliquer sur bouton "validate"
  - Valider message création affiché
  - Envoi d'un mail de validation
- Pas de connexion avec des champs non renseignés
  - testCannotCreateAccountWithOneFieldMisinformed()
  - testCanCreateAccountWithBadPassword()
  - Ouvrir page création de compte
  - Ne pas renseigner tout les champs et/ou Utiliser le mauvais mot de passe
  - Cliquer sur bouton "validate"
  - Valider message erreur affiché





	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

Vérification du formulaire : nom et prénom sont des champs obligatoires, l'adresse mail devra être vérifiée afin de savoir si elle respecte les règles d'une adresse mail bien formée (le garant doit pouvoir recontacter la personne qui a envoyé le mail) c'est un champ bien sûr obligatoire, et le champ commentaire sera vérifié pour savoir si il n'est pas vide

A la validation du formulaire les différents champs sont vérifiées, si un ou plusieurs champs ne sont pas correctes suivant ce qui a été défini plus haut alors un message d'erreur apparaît (une pop-up qui indique le champ ou les champs non correctes), si les champs sont tous correctes alors un message de validation apparaît (une pop-up qui indique que le message a bien été envoyée).

## Service

nom de l'interface : `IServiceSendMail`

méthodes de l'interface : **checkMail**(String suname,String name,String email,String category,String message);

**sendMail**();

Contrôle : la méthode `checkMail` vérifiera la validité des champs comme indiqué plus haut, puis appellera la méthode `sendMail`() afin d'envoyer le mail vers la boîte mail du gérant de la société.

```
public class MailMail
{
    private MailSender mailSender;

    public void setMailSender(MailSender mailSender) {
        this.mailSender = mailSender;
    }

    public void sendMail(String from, String to, String subject, String msg) {

        SimpleMailMessage message = new SimpleMailMessage();

        message.setFrom(from);
```

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

```

        message.setTo(to);
        message.setSubject(subject);
        message.setText(msg);
        mailSender.send(message);
    }
}

```

Ensuite le service de contact utilisera ce service de la façon suivante après avoir validé les données en input :

```

mm.sendMail("com.miage.dreamit@gmail.com",
            "com.miage.dreamit@gmail.com",
            "Contact from <user>",
            "<message from web contact>");

```

Utilisation du SMTP gmail avec le compte suivant :

Login: com.miage.dreamit

Pass : ProxyStat2013

Utilisation librairie javamail

```

<dependency>
    <groupId>javax.mail</groupId>
    <artifactId>mail</artifactId>
    <version>1.4.3</version>
</dependency>

```

Déclaration du bean

```

<bean id="mailSender" class="org.springframework.mail.javamail.JavaMailSenderImpl">
    <property name="host" value="smtp.gmail.com" />
    <property name="port" value="587" />
    <property name="username" value="com.miage.dreamit" />
    <property name="password" value="ProxyStat2013" />

    <property name="javaMailProperties">
        <props>
            <prop key="mail.smtp.auth">true</prop>

```

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

```

<prop key="mail.smtp.starttls.enable">true</prop>
</props>
</property>
</bean>

```

En attendant d'avoir l'email de contact proxyStation, on forcera le "TO" des emails à ce même email pour des raisons de test. Nous sommes conscients de la dépendance à internet pour l'envoi d'email.

En résumé :

1. On récupère de l'interface web les paramètres de la page de contact dans le contrôleur
2. On vérifie la validité des informations reçues
3. On appelle le service d'envoi d'email
4. On affiche une notification sur la page web pour montrer que le mail a bien été envoyé

## Persistence

Pas d'utilisation bdd. Fichier de config smtp.properties (serveur smtp utilisé, port, mail destinataire, authentification)

## Jeux de tests

Jeux de tests unitaires

- testSendMailWithCorrectFields()
- testSendMailWithFieldSurnameEmpty()
- testSendMailWithFieldNameEmpty()
- testSendMailWithFieldEmailEmpty()
- testSendMailWithFieldEmailMalFormed()
- testSendMailWithFieldMessageEmpty()

Jeux de tests IHM

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

- `testCreateMessageWithCorrectFields()`
  - remplir l'ensemble des champs
  - remplir le champ mail bien formé
  - cliqué sur le bouton envoyé
- `testCreateMessageWithFieldSurnameEmpty()`
  - remplir l'ensemble des champs en omettant le champ surname
  - remplir le champ mail bien formé
  - cliqué sur le bouton envoyé
- `testCreateMessageWithFieldNameEmpty()`
  - remplir l'ensemble des champs en omettant le champ name
  - remplir le champ mail bien formé
  - cliqué sur le bouton envoyé
- `testCreateMessageWithFieldEmailEmpty()`
  - remplir l'ensemble des champs en omettant le champ email
  - remplir le champ mail bien formé
  - cliqué sur le bouton envoyé
- `testCreateMessageWithFieldEmailMalFormed()`
  - remplir l'ensemble des champs
  - remplir le champ mail sans @ par exemple
  - cliqué sur le bouton envoyé
  
- `testCreateMessageWithFieldMessageEmpty()`
  - remplir l'ensemble des champs en laissant le champ message vide
  - remplir le champ mail bien formé
  - cliqué sur le bouton envoyé

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

## V1 : Réservation

La réservation va permettre aux clients de réserver l'intégralité de leur séjour de la location d'appartement à l'abonnement de remontée mécanique.

### IHM

Il faut développer la page de création de compte: <http://<server>/proxystat/booking>.

Pour accéder à cette page, il faut que le client soit authentifié sur le site.

Le client arrive sur la page de réservation, il saisit la période sur laquelle il veut réserver, le nombre de personnes, ainsi que le prix souhaité afin de choisir son appartement parmi la liste qui lui ai proposée.

En période scolaire, les semaines se font uniquement du samedi au samedi. Il est possible de réserver plusieurs semaines consécutives.


Hors période scolaire, il est possible de réserver un appartement à la journée.

Seuls les appartements libres et correspondant aux critères sont présents dans la liste:

- libre pour la période demandée
- pour le nombre de personnes renseigné
- pour le prix demandé : être inférieur ou égal au prix saisi

La liste des appartements se rafraîchit automatiquement à chaque changements de critères.

Cette liste sera une datatable avec un bouton "réserver" pour chaque appartement.

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0




Lorsque l'appartement est choisi (clique sur le boutons "réserver"), on passe sur la page suivante pour la réservation du matériel et des forfaits de remontées mécaniques. La réservation du matériel et des forfaits se fait sur la même page.

Il faut développer la page de création de compte: <http://<server>/proxystat/equipment>

On réserve autant de fois le matériel et le forfait qu'il y a de personnes qui ont été saisies sur la page "booking".

*ex: 3 personnes pour la réservation de l'appartement = 3 passage sur la page "equipment"*

La liste des équipements se rafraîchit en fonction des critères renseignés.

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

Cette liste sera une datatable avec des “checkbox” qui permettront aux clients de sélectionner ce qu’ils souhaitent en pouvant modifier leurs choix tant que ceux ci n’ont pas été validés.

Les forfaits de remontées mécaniques se décompose en 3 parties:

- Les forfaits ski alpin
- les forfaits ski nordique
- le télésiège

Si le nombre de jour est à zéro et que le forfait saison n’est pas sélectionné, cela veut dire que le client n’est pas intéressé par le forfait.

Si le forfait saison est coché, on ne prend pas en compte le nombre de jour indiqué.

Le prix total de la réservation du séjour se met à jour au fur et à mesure de la sélection du client afin qu’il puisse savoir combien il devra payer au final.





	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

Lorsque les équipements et les forfaits de remontées ont été sélectionnés et réservés pour chaque membres participant au séjour, on passe au récapitulatif de la réservation afin de confirmer et passer au paiement.

Il faut développer la page: <http://<server>/proxystat/recap>


Cette page présente le récapitulatif de toute la réservation :

- appartement
- équipements
- forfaits

On peut retrouver le prix total de la réservation ainsi que le prix à payer de suite (calcul = 20 % du montant de la réservation).

Des options telles que l' option annulation sont proposées à ce moment de la réservation.

Si l' option est cochée, le prix se met à jour en fonction du supplément..

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0



**Réserver un séjour**

Réserver un appartement >> Réserver le matériel et le forfait >> **Récapitulatif** >> Paiement

Récapitulatif de toute la réservation  
appartement  
équipements  
forfaits

Prix totale de la réservation : 980 Euros Acompte à payer : 196 Euros

<input checked="" type="checkbox"/> Option linge (draps, serviettes)	<input checked="" type="checkbox"/> Assurance perte de matériel
<input checked="" type="checkbox"/> Option ménage	<input checked="" type="checkbox"/> Assurance neige
<input checked="" type="checkbox"/> Lit bébé	
<input checked="" type="checkbox"/> Location d'un garage	<input checked="" type="checkbox"/> Option annulation

**Confirmer la réservation**

Lorsque le client a vérifié le récapitulatif de sa réservation et choisit les options qu'il souhaite, il confirme sa réservation en cliquant sur le bouton "Confirmer la réservation".

Il est alors redirigé vers la page de paiement : <http://<server>/proxystat/paiement>

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0



La réservation est réellement prise en compte après paiement de l'acompte.  
 Les vérifications sont effectués pour savoir si les informations bancaires sont bonnes et si le client peut payer.  
 Si tout est correct, la réservation est enregistrée en base de données. Les stocks des équipements sont mis à jour.  
 Une page de confirmation de la réservation s'ouvre avec un message de remerciement.

## Service

/\*\*

- \* Interface permettant d'enregistrer la réservation
- \* @author
- \* @version 1.0

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

```

*/
interface ServiceValidateSearchBookingParameters {

    /**
     * Valide que la taille saisie est possible
     * @param GuestSize Taille saisi dans le formulaire
     */
    void validateGuestSize(float GuestSize);

    /**
     * Valide que la pointure saisie est possible
     * @param ShoesSize pointure saisi dans le formulaire
     */
    void validateGuestShoesSize(integer ShoesSize);
}

```

## Persistence

On utilisera les classes concernant Customer, booking et Partner

Réutilisation/Création DAO IBookingService

Réutilisation/Création DAO IPartnerService

## Jeux de tests

Jeux de tests unitaires


- testPriceChangeWhenChoiceChange()
  - Le prix total de la réservation est affiché
  - Le client change son choix (coche ou décoche un élément)
  - Le prix change en fonction du choix du client
- testCalculationOfPricelsGood()
  - Le client fait ses choix

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

- Le prix est mis à jour avec les bons calculs
- testStockUpdateWhenBookingsIsValidate()
  - Le client A validé sa réservation et payer l'acompte
  - Le stock d'équipements réservés se met à jour en base chez les partenaires
- testFlatBecomeUnavailableWhenBookingsIsValidate()
  - Le client A validé sa réservation et payer l'acompte
  - Mise à jour en base. L'appartement est réservé donc plus disponible.
- testPaymentIsAccepted()
  - Le client saisit ses informations de carte bancaire
  - Le service vérifie la validité de la saisie
  - Le paiement est effectué

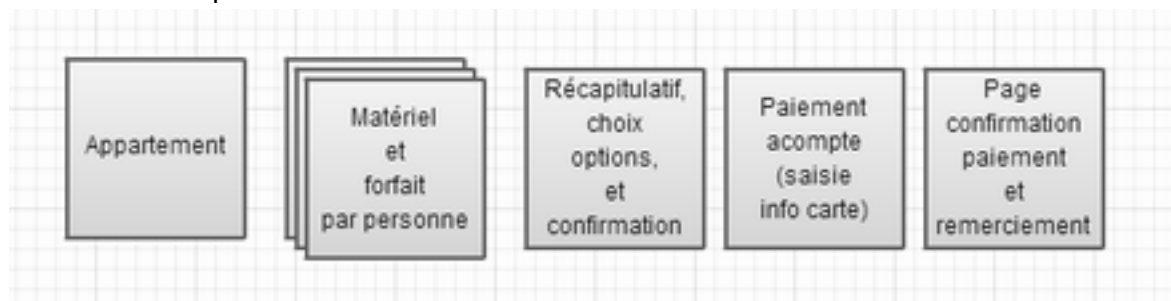
#### Jeux de tests IHM

- testRadioButton1IsUncheckedWhenButton2IsChecked()
  - Le bouton 1 du radio bouton est coché
  - L'utilisateur coche le bouton 2
  - Le bouton1 se décoche
- testCalendarComeUpWhenClickIntoArrowInDateField()
  - L'utilisateur veut saisir une date dans les champs appropriés
  - Il clique sur la flèche pour déployer les possibilités
  - Un calendrier apparait
- testPriceChangeWhenSliderMove()
  - L'utilisateur fait bouger le curseur sur la ligne des prix
  - Le prix affiché en dessous correspondant à la position du curseur change

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0


## Versions incrémentales du développement

Le workflow de paiement est le suivant :




Les règles de gestion étant nombreuses, l'implémentation du processus de réservation sera lotti en versions incrémentales afin d'en faciliter le développement. Chaque version devra faire l'objet de tests unitaires et d'intégration afin d'en assurer la régression lors du développements des versions supérieures. Le détails des versions est le suivant :

Version	Appartement	Matériel et forfait par personne	Récapitulatif Choix options Confirmation	Paiement acompte en saisissant info carte	Page confirmation paiement et remerciement
1	<ul style="list-style-type: none"> <li>* Tout appartement listés, pas de filtre</li> <li>* Pas d'affichage des commentaires, notation</li> <li>* Pas de filtre sur le prix possible</li> <li>* Choix date que sur une semaine période vacances scolaires</li> </ul>	Non développé dans la V1	<ul style="list-style-type: none"> <li>* Récapitulatif avec le choix de l'appartement</li> <li>* Faire apparaitre montant acompte</li> <li>* Possibilité activer option annulation et revalorisation du panier en direct (3% prix), afficher description option</li> </ul>	<ul style="list-style-type: none"> <li>* Saisie information carte bancaire, afficher montant acompte qui sera payé</li> <li>* Développer un service de paiement qui retourne toujours vrai</li> </ul>	<ul style="list-style-type: none"> <li>* Affichage simple anglais ou français, remercier achat et mettre un lien sur la page d'accueil</li> </ul>

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

2	* Ajouter champ pour spécifier nombre de personnes	* Génère bien N pages suivant nombre de personnes sélectionnées avant * Click sur bouton "Ne pas réserver de matériel pour cette personne" à chaque page fait juste passer à la page suivant	* Ajouter dans récapitulatif 1 ligne par personne de la réservation, dans autre version on listera tout ce que la personne à choisi		
3		* Gestion des réservations matériel de sport	* Update recap pour faire apparaitre les résa matériel		
4		* Gestion des réservations de forfaits	* Update recap pour faire apparaitre les résa forfait		
5				* Affichage de toutes les options avec revalorisation en cliquant sur bouton * Option sport grisées si on réserve uniquement un appartement	
6	* datatables pour pouvoir filtrer * ajout filtre prix qui filtre	* datatables pour pouvoir filtrer * ajout filtre prix qui filtre en direct	* datatables pour pouvoir filtrer * ajout filtre prix qui filtre en direct		

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

7	* Ajout bouton pour voir tous les commentaires sur un service	* Ajout bouton pour voir tous les commentaires sur un service	* Ajout bouton pour voir tous les commentaires sur un service		
---	---	---	---	--	--



	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

\* affichage moyenne note par service

## V1 : Annulation de réservation

Cette partie va permettre aux clients d'annuler une réservation en cas de désistement.  
Pour accéder à l'annulation, il faut que le client soit authentifié.

### IHM

Il faut développer la page d'annulation de réservation: <http://<server>/proxystat/annulation>


Annulation de la réservation

Récapitulatif de la réservation

Annuler cette réservation

Pour pouvoir atteindre la page d'annulation, il faut passer par le récapitulatif en détail de la réservation accessible via ces pages et cliquer sur le bouton "Annuler cette réservation".

- /myHistory : Cette page contiendra les commandes passées par le client de façon exhaustive avec un lien par booking permettant d'accéder au détail

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

- /myHistoryDetail : Cette page détaillera un booking en particulier et permettra de noter chacun des services qui ont été effectués sur le booking

Lors de l'annulation, un calcul est fait pour savoir s'il doit rembourser ou non une partie de la réservation.

Cela dépend de différents critères :

- Annulation avant X jours
- Souscription de l'assurance annulation

Le stock de matériel est également remis à jour lors de l'annulation de la réservation.

Les équipements qui avaient été réservés sont remis en stock.

Lors de l'annulation de la réservation, un mail de notification est envoyé aux partenaires.

## Service

Création d'un service permettant de vérifier que le client a le droit d'annuler la réservation. Il permettra ensuite d'annuler cette réservation.

Réutilisation du service de paiement ServicePayment

```
/**
 * Interface d'annulation de réservation
 * @author
 * @version 1.0
 */
interface ServiceDeleteBooking {

    /**
     * Valide que la réservation peut être annulée */
    Boolean validateAnnulationAuthorized();
}
```

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

```

/**
 * Calcul le montant du remboursement s'il y en a un, en fonction de la date et de l'option
 annulation*/
float CalculateReimbursement();

/**
 * Annuler la réservation en base de données
 Remettre à jour le stock des équipements*/
void deleteBooking();

}

```

## Persistence

On utilisera les classes Customer, booking

Réutilisation/Création DAO IBookingService

Réutilisation/Création DAO IPartnerService

## Jeux de tests

- testDisplayAllBookingOfCustomer()
  - Le client ouvre la page des réservations
  - Toutes les réservations qu'il a effectué sont affichées
- testCustomerAuthorized()
  - Le client demande l'annulation en cliquant sur le bouton
  - Vérification que le client est bien autorisé à annuler

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

- Demande de confirmation d'annulation au client +  
Message annonçant le montant du remboursement s'il y en a un.
- testCustomerWantConfirmAnnulation()
  - Le client demande l'annulation en cliquant sur le bouton
  - Vérification que le client est bien autorisé à annuler
  - Le message de demande de confirmation apparait +  
Message de montant du remboursement
  - Le client clique sur "confirmer" l'annulation
  - La page pour le paiement du remboursement s'ouvre
- testCustomerWantStopAnnulation
  - Le client demande l'annulation en cliquant sur le bouton
  - Vérification que le client est bien autorisé à annuler
  - Le message de demande de confirmation apparait +  
Message de montant du remboursement
  - Le client clique sur "Quitter"
  - Le client est redirigé sur la page de détail de la réservation
- testCustomerNotAuthorized()
  - Le client demande l'annulation
  - Vérification que le client est autorisé
  - Message "Autorisation impossible" affiché
- testCustomerReimburseAnnulation()
  - Le client a confirmer l'annulation
  - Détail du remboursement à effectuer
  - Paiement par le client
  - Mail de confirmation
- testBookingsdeleted()
  - Le client annule la réservation après vérification des droits
  - Le client paie le remboursement
  - Message "Réservation annulée"
  - Mail envoyée + Réservation supprimer en base
- testStockUpdateWhenBookingsDeleted()
  - Le client A validé sa réservation et payer l'acompte
  - Le stock d'équipements réservés se met à jour en base chez les partenaires
- testFlatBecomeAvailableWhenBookingsDeleted()

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

- Le client A validé sa réservation et payer l'acompte
- Mise à jour en base. L'appartement est réservé donc plus disponible.

## V1 : Post de commentaires page d'accueil

Cette fonctionnalité permettra aux utilisateurs de donner leur avis et de noter les services qu'ils ont utilisés. Cette fonctionnalité nécessite d'être authentifié en tant que role customer.

### IHM

Il faut ajouter sur la homepage :


- un affichage des 5 derniers posts
- un champ message
- un bouton de soumission du message

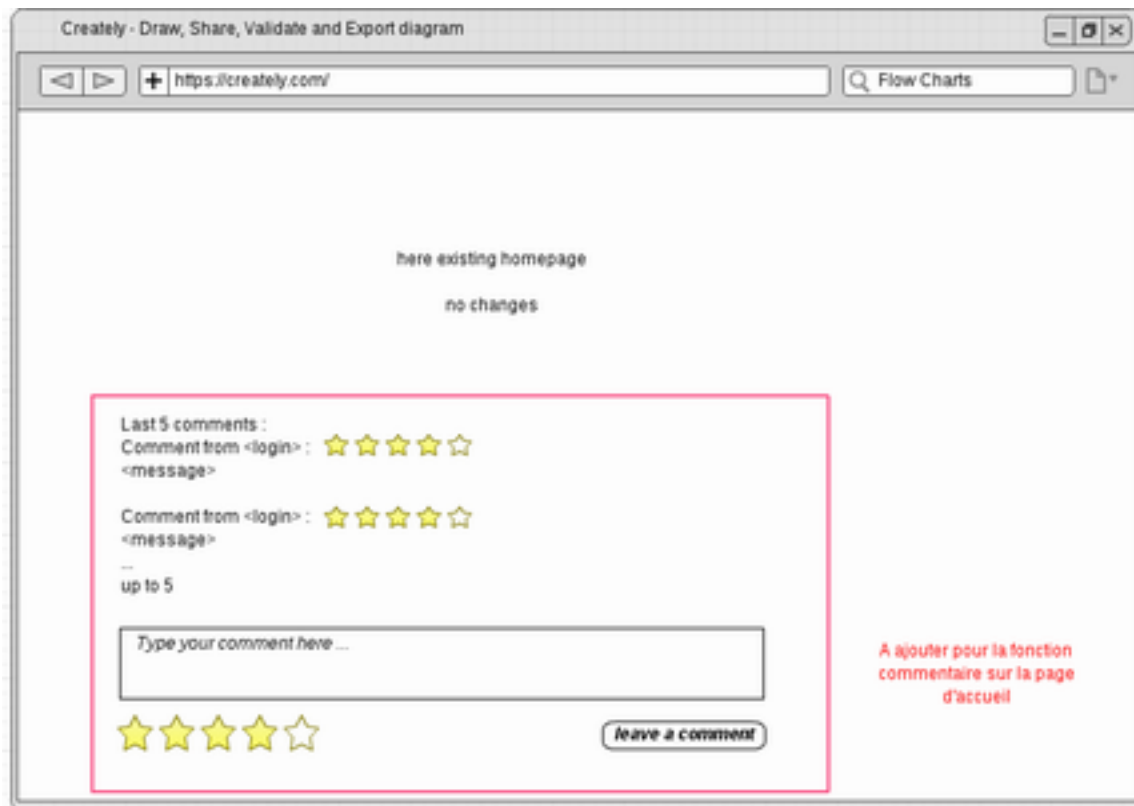
Il faut pouvoir afficher tous les commentaires en cliquant sur un bouton déroulant nommé "See all comments".

Il n'y a pour l'instant pas de vérification du contenu du message à faire (ordre SQL ou injures).

On peut noter uniquement avec les valeurs suivantes : 1,2,3,4,5.

La sélection s'effectue à l'aide de 5 étoiles.

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0



## Service

Création du service IRatingService (ou bien à mettre dans ICustomerService ???)

- Création : Récupération des commentaires à afficher

/\*\*

\* Retourne la liste des Rating qui n'ont pas de lien avec un service

\*/

List<Rating> getHomeRatings()

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

- Création : Sauvegarde d'un commentaire

/\*\*

- \* Sauvegarder un commentaire pour un client sur la homepage
- \* Il n'y a pas de lien avec un service car c'est un commentaire global
- \*/

Rating saveHomeRating(Rating rating)

## Persistence

Les commentaires sont dans l'entité Rating.

Les commentaires à récupérer sur la page d'accueil sont les Rating qui ne sont liées à aucune entité Service.

Les commentaires à sauvegarder effectueront un lien entre un Rating et un Customer en mettant le champ Date à la date du jour. Le type date est du java.util il y a donc bien la notion de date/heure prise en compte.

Utilisation du DAO IRating

- Création : Récupération des commentaires à afficher

/\*\*

- \* Retourne la liste des Rating qui n'ont pas de lien avec un service
- \*/

List<Rating> getHomeRatings()

- Réutilisation : Sauvegarde d'un commentaire

/\*\*

- \* Sauvegarder un commentaire pour un client sur la homepage
- \* Il n'y a pas de lien avec un service car c'est un commentaire global
- \*/

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

Rating saveHomeRating(Rating rating)

## Tests

### Tests unitaires service

- testGetOnlyHomeRatings() : valider appel DAO effectué
- testSaveHomeRatingSetDateToToday() : valider qu'on met à l'attribut Date à la date du jour en appelant le service et en comparant la valeur de l'attribut Date avec la date du jour (sans les heures)
- testSaveHomeRatingNotLinkToService() : valider en appelant le service que l'attribut Service du Rating est null

### Tests unitaires dao

- testGetOnlyHomeRatings() : valider en mettant 1 rating lié service et un autre non qu'on retourne uniquement le rating non lié au service
- testSaveHomeRatingLinkedToCustomer() : valider que l'attribut customer n'est pas vide après sauvegarde
- testSaveHomeRatingNotLinkToService() : valider que l'attribut Service est vide après sauvegarde


### Tests d'intégration

- testDisplayLast5Ratings()
  - Ouvrir la homepage
  - S'identifier avec un client
  - Valider que la frame contient les commentaires affichés
- testDisplayAllRatings()
  - Ouvrir la homepage
  - S'identifier avec un client
  - Cliquer sur le bouton pour afficher tous les commentaires
  - Valider que le frame contient plus de 5 commentaires
- testCannotSubmitRatingWithoutMessage()
  - Ouvrir la homepage
  - S'identifier avec un client



	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

- Cliquer sur “Leave a message”
  - Valider message d’erreur s’affiche
- testDefaultRatingIs3()
  - Ouvrir la homepage
  - S’identifier avec un client
  - Valider que le rating par défaut est à 3 étoiles
- testCanSubmitAndSeeRating()
  - Ouvrir la homepage
  - S’identifier avec un client
  - Remplir le champ de commentaires avec “Test online comment”
  - Cliquer sur “Leave a message”
  - Valider message “Your rating have been submitted”
  - Valider qu’on retrouve le message en haut de l’affichage des commentaires

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

## V1 : Post de commentaires sur services

Cette fonctionnalité permettra aux clients d'évaluer les services qu'ils ont utilisés en ayant fait une commande.

Les commentaires et rating seront affichés lors du workflow de réservation dans un second temps sur chaque service proposé.

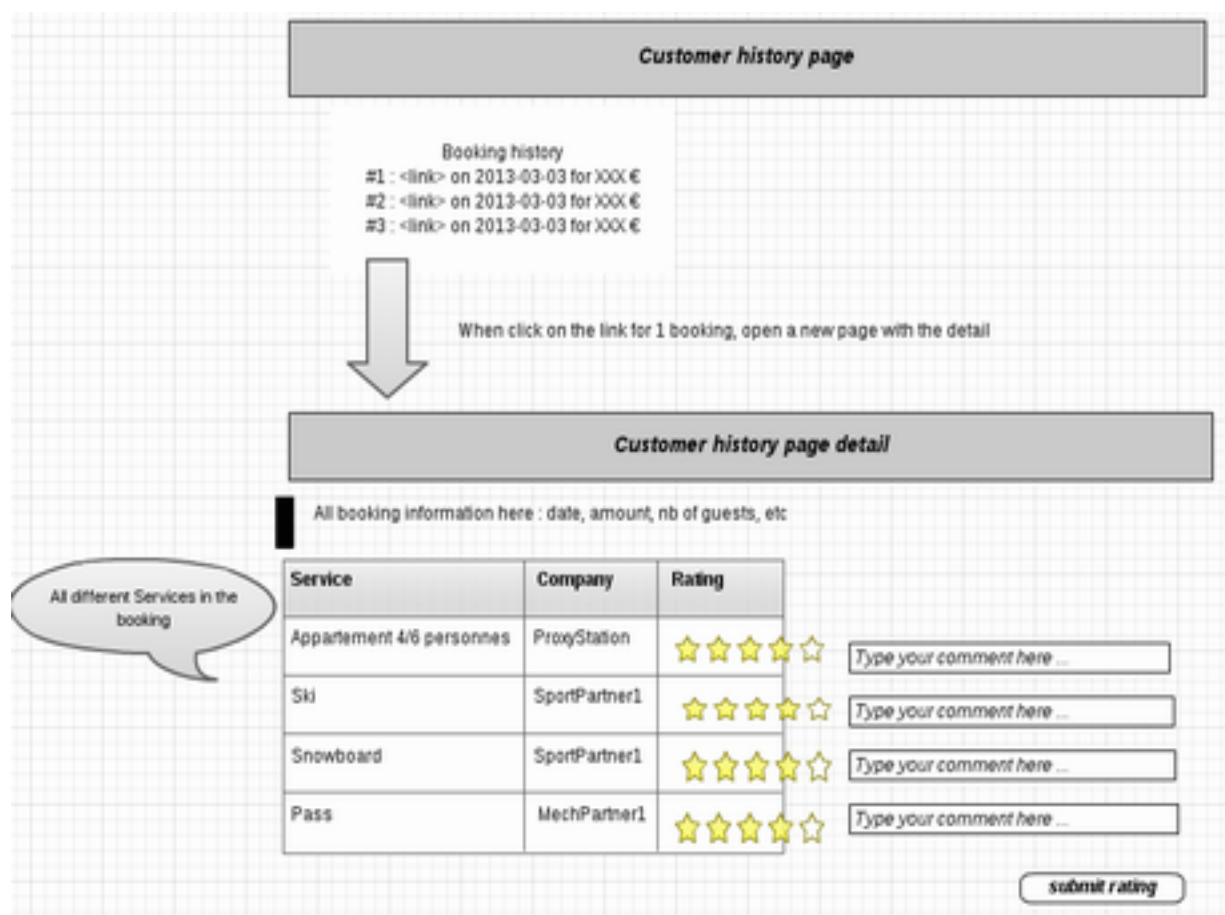
### IHM

Deux nouvelles pages sont à créer :

- /myHistory : Cette page contiendra les commandes passées par le client de façon exhaustive avec un lien par booking permettant d'accéder au détail
- /myHistoryDetail : Cette page détaillera un booking en particulier et permettra de noter chacun des services qui ont été effectués sur le booking

Il faut ajouter un lien vers cette page depuis la homepage dans le menu existant.

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0



## Service

Réutilisation de l'interface ICustomerService

- Création

/\*\*

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

\* Save the rating with a customer and service link

\*/

Rating save(Rating rating);

/\*\*

\* Permet de sauvegarder plusieurs ratings configurés depuis l'IHM

\*/

boolean saveAllRatings(List<Rating> ratings)

Réutilisation de l'interface IBookingService

- Création

/\*\*

\* Retrieve booking for a customer

\*/

List<Booking> getBookingsBy(Customer customer)

## Persistence

Réutilisation DAO IDAORating

Rating save(Rating rating)

Réutilisation/Création DAO IBookingService

List<Booking> getBookingsBy(Customer customer)

## Tests

Tests unitaires service

- testSaveAllRatingsCallMultipleSaveRating() : valider que l'appel à ce service appelle plusieurs fois le service unitaire saveRating
- testSaveNullRatingListsThrowsException() : valider qu'en passant un paramètre null on récupère une exception

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

- testSaveAllRatingsWith2RatingsReturnsTrue() : valider retour à true en passant une liste de 2 ratings correctement renseignés
- testGetBookingsByCustomerCallDAO() : valider que le service appelle le DAO correspondant

#### Tests unitaires DAO

- testGetBookingsByCustomerReturnEmptyListWithNullCustomer() : valider qu'on récupère une liste initialisée mais vide en passant un paramètre null
- testGetBookingsByCustomerReturnEmptyListWithCustomerWithoutBooking() : valider qu'en passant un client qui n'a jamais fait de commande (booking) la liste retournée est bien renseignée mais vide
- testGetBookingsByCustomerReturnOnlyCustomerBookings() : valider qu'en passant un client qui a déjà commandé on ne récupère dans la liste uniquement ses bookings et pas ceux d'un autre client

#### Tests d'intégration

- testDisplayBookingHistoryCorrectly()
  - Ouvrir homepage
  - Se connecter avec un compte client qui a déjà commandé
  - Cliquer sur le lien de la page d'historique client
  - Valider l'affichage des liens
- testDisplayDefaultMessageWhenNoBookingForCustomer()
  - Ouvrir homepage
  - Se connecter avec un compte client qui n'a jamais commandé
  - Cliquer sur le lien de la page d'historique client
  - Valider l'affichage de la div qui spécifie pas d'historique commande "You don't have any order yet, click here <link to booking workflow>"
- testDisplayBookingDetailFromBookingHistory()
  - Ouvrir homepage
  - Se connecter avec un compte client qui a déjà commandé
  - Cliquer sur le lien de la page d'historique client
  - Cliquer sur le lien du premier booking
  - Valider qu'on arrive sur la page de détail
  - Valider que le détail de la commande est bien affiché

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

- Valider qu'il y a le tableau de service affiché
- testCannotLeaveRatingWithoutAtLeastOneMessage()
  - Ouvrir homepage
  - Se connecter avec un compte client qui a déjà commandé
  - Cliquer sur le lien de la page d'historique client
  - Cliquer sur le lien du premier booking
  - Cliquer sur submit - si aucun commentaires précédemment renseigné, on doit afficher un message d'erreur car les champs textes doivent être renseignés
- testCanLeaveRatingWithDifferentNotations()
  - Ouvrir homepage
  - Se connecter avec un compte client qui a déjà commandé au moins 2 services
  - Cliquer sur le lien de la page d'historique client
  - Cliquer sur le lien du premier booking
  - Laisser un rating à 3
  - Laisser un rating à 4
  - Envoyer rating
  - Valider message de prise en compte "Rating submitted"
  - Valider rating 3 et 4 mis à jour dans l'IHM
- testCustomerHistoryRatingSetTo3ByDefault()
  - Ouvrir homepage
  - Se connecter avec un compte client qui a déjà commandé au moins 2 services
  - Cliquer sur le lien de la page d'historique client
  - Cliquer sur le lien du second booking
  - Valider notation à 3 par défaut

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

## V2 : Sport / Administrer offre

Cette fonctionnalité doit permettre aux utilisateurs de profil Partenaire sport de configurer leur offre de matériel disponible en ligne.

### IHM

Dans le workflow, l'utilisateur est sur la page d'accueil et après une identification réussie, il est directement routé sur une page d'accueil de backoffice lui présentant les actions suivantes :

- - Administrer mon offre
- - Consultation et édition du matériel à préparer
- - Rendu de matériel


Les pages à développer dans cette fonctionnalité seront accessibles en cliquant sur le lien "Administrer mon offre".

2 périmètres d'administration sont identifiés :

1. Catégories du matériel
2. Equipement de sport

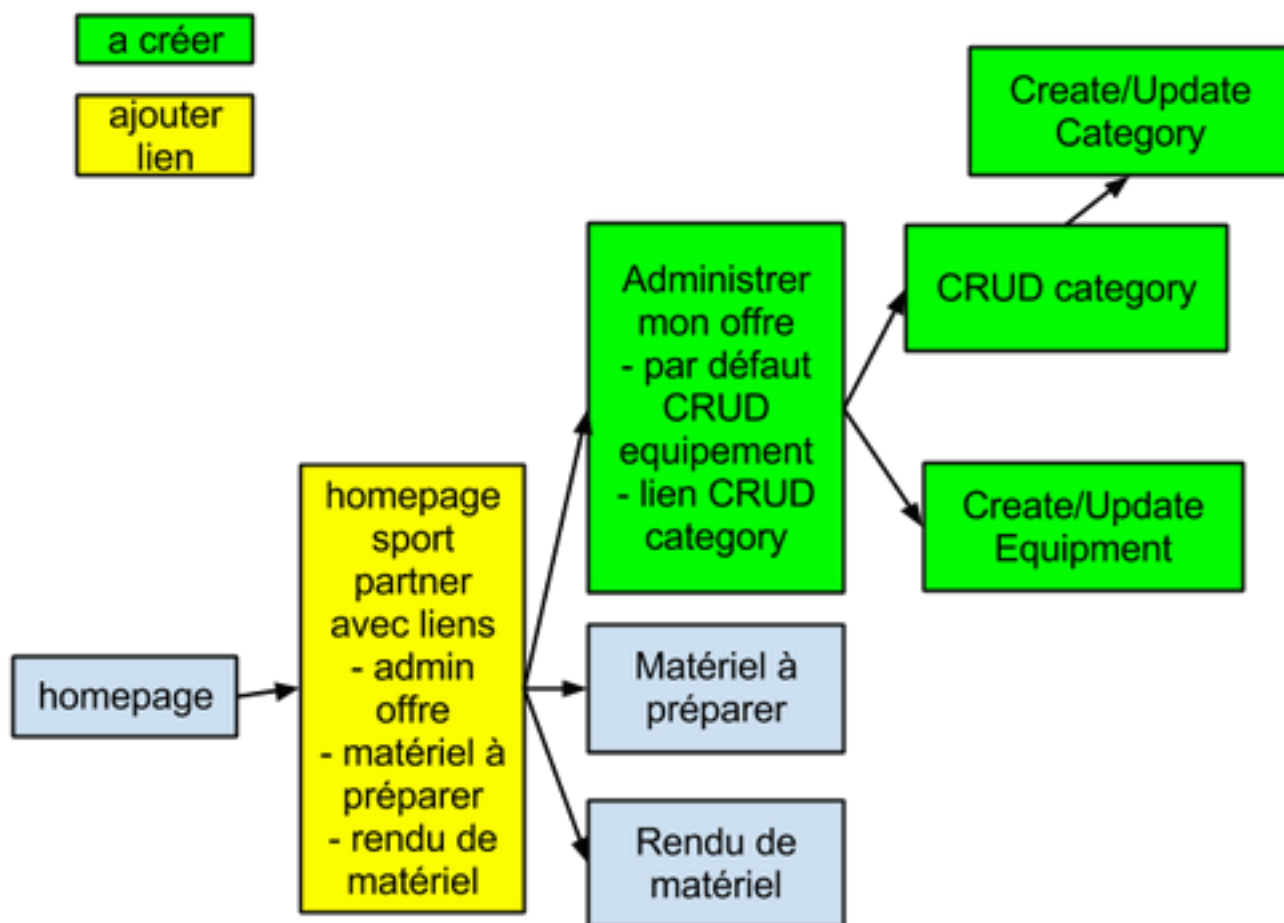
Par défaut en cliquant sur le lien "Administrer mon offre" on arrive sur la page CRUD Equipment et on a un lien "Administrer les catégories" permettant d'accéder à la page CRUD Category.

On ne développera pas de page pour administrer les catégories d'âge car celles-ci ne sont pas amenées à changer régulièrement et les règles de gestion sont plus simple à configurer directement dans la table. Il y aura en effet entre 5 et 10 lignes (bébé, enfant, étudiant, adulte, senior, age d'or, etc).

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

Le workflow et l'organisation des pages est la suivante :

Légende :








	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

Dans une optique de standardisation et de facilitation d'usage de l'interface d'administration, toutes les pages CRUD auront le style suivant. Ce qui est important est la gestion des opérations :

- create et update sont un seul et même formulaire
- la recherche s'effectue avec le plugin js datatable directement, il n'y a rien à implémenter
- la suppression est sur chaque ligne, pas de gestion de checkbox

Name (job title)	Age	Nickname	Employee		
Giacomo Guizzoni Founder & CEO	36	Peldi	<input checked="" type="checkbox"/>		
Marco Botton Tuttofare	34		<input checked="" type="checkbox"/>		
Mariah Maclochan Better Half	37	Potato	<input checked="" type="checkbox"/>		
Valerie Liberty Head Chef	3	Val	<input checked="" type="checkbox"/>		
Guido Jack Guizzoni	6	The Guide	<input type="checkbox"/>		

 Create

<< < 1 2 3 4 5 6 7 8 9 10 > >>


L'administration des équipements de sport s'effectuera sur le même modèle d'IHM. Cependant, l'entité Equipment a des liens avec d'autres entités : elle est rattachée à une Category, potentiellement un ageCategory et liée à un SportPartner.

Il faut donc développer des pages CRUD pour AgeCategory, le CRUD Sport Partner fait partie de la V3 et est réservé à l'administrateur du site.

### Catégories de matériel

Cette page sera une page CRUD classique permettant d'ajouter, modifier, supprimer des catégories en renseignant les champs de l'entité sauf l'ID.

Il est possible de supprimer une catégorie dans laquelle des équipements sont encore disponibles. Le lien catégorie dans l'objet Equipment correspondant sera mis à null, normalement géré par Hibernate.

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

L'IHM sera donc celle ci-dessus. Le style est à titre indicatif. Il faudra utiliser le plugin js datatables pour afficher les données. Les données à afficher seront celles de l'entité "Category".

Par défaut au chargement de la page toutes les catégories sont chargées.

En cliquant sur l'icone/bouton delete, un message de confirmation doit demander la validation à l'utilisateur.

En cliquant sur le bouton create, un formulaire doit apparaître dans une nouvelle page permettant de saisir les informations de création.

En cliquant sur le bouton update, le même formulaire que celui de création doit apparaître mais avec les champs de l'entité pré-rempli avec les informations de l'entité choisie.

## Equipement

Cette page permettra d'effectuer les opérations CRUD sur les équipements de la société Sport Partner du user connecté.

Les champs minimumHeight et Quantity ne peuvent pas être inférieurs à 0.

Le champ size devra être de type String pour les vêtements (S, XL, etc) et non Integer. Il n'y a pas de restriction sur le champ quality, la partenaire est responsable de la saisie correcte.

Il faut afficher et pouvoir modifier uniquement l'offre de la société auquel appartient l'utilisateur connecté.

La page CRUD equipment sera basée sur la même maquette que ci-dessus mais pour les liens avec les autres entités Category, AgeCategory il faut afficher une liste déroulante basée sur la méthode toString de l'objet ou afficher les informations les plus pertinentes (nom + description).

Spécificités de la page CRUD Equipement :

- il faut lister uniquement les Equipement appartenant au SportPartner de l'utilisateur profil partenaire connecté. En effet, un partenaire ne doit pouvoir administrer que son offre

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

- il ne pas pouvoir afficher/modifier le lien SportPartner dans cette IHM CRUD, le lien doit être fait automatiquement lors de la création de l'équipement et ne doit plus être modifiable (il ne faut donc pas l'afficher dans le formulaire de création/update ni dans le tableau)
- champs devant être renseignés Size, minimumHeight, ageCategory.
- Le champ stockQuantity sera mis à 0 si non renseigné.
- Le lien Catégorie restera à null si non renseigné.

## Service

Création interface et implémentation service IAdministrationSportPartnerService  
+ création des factory si nécessaire pour utilisation dans les controllers afin de passer des objets directement aux services

Signatures

```
/** Create or update
Category saveEquipmentCategory(Category)
```

```
void deleteEquipmentCategory(Category)
```

```
List<Category> findAllCategory()
```

```
/** Create or update
Equipment saveEquipment(Equipment)
```

```
void deleteEquipment(Equipment e)
```

```
List<Equipment> findAllEquipment()
```

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

## Persistence

Utilisation des méthodes existantes dans DAO génériques

Entités :

- Administration catégorie d'équipement : Category
- Administration Equipement : principalement Equipment avec liens AgeCategory, SportPartner, Category

Il manque un extends PeriodService sur l'entité Equipment

## Tests

Tests unitaires service

- testCannotCreateEquipmentCategoryWithEmptyName() : test si champ isempty ou "" throws exception
- testCannotCreateEquipmentCategoryWithNullCategory()
- testCannotCreateEquipmentCategoryWithNullName()
- testCannotCreateEquipmentCategoryWithNullDescription() : test si champ null throws exception
- testCannotCreateEquipmentCategoryWithEmptyDescription() : test isempty throws exception
- testCreateEquipmentCallPersistenceLayer()
- testCannotUpdateEquipmentCategoryWithEmptyName() : fait dans la méthode save
- testCannotUpdateEquipmentCategoryWithNullName() : fait dans la méthode save
- testCannotUpdateEquipmentCategoryWithEmptyDescription() : fait dans la méthode save
- testCannotUpdateEquipmentCategoryWithNullDescription() : fait dans la méthode save
- testUpdateEquipmentCategoryCallPersistenceLayer() : fait dans la méthode save
- testCanListEquipmentCategoriesByCallingPersistenceLayer()
- testCanDeleteEquipmentCategoryByCallingPersistenceLayer()
- testCannotDeleteEquipmentCategoryWithEmptyName() : test si champ isempty ou "" throws exception
- testCannotDeleteEquipmentCategoryWithNullCategory()

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0


- testCannotDeleteEquipmentCategoryWithNullName()
- testCannotDeleteEquipmentCategoryWithNullDescription() : test si champ null throws exception
- testCannotDeleteEquipmentCategoryWithEmptyDescription() : test isempty throws exception
- + same test as for creation (null, empty for name and description)
- testCannotSaveNullEquipment()
- testEquipmentStockQuantityIsSetTo0IfNotFeededOnSave() : ce test doit valider que le champ stockQuantity de l'entité est mis à 0 s'il est à null lors de l'appel du service de création
  - testEquipmentStockQuantityIsSetTo0IfInferiorToZeroOnSave()
  - testEquipmentMinimumHeightIsSetToZeroIfNotFeededOnSave() : ce test doit valider que le champ minimumHeight de l'entité est mis à 0 s'il est à null lors de l'appel du service de création
    - testEquipmentMinimumHeightIsSetToZeroIfInferiorToZeroOnSave()
  - testEquipmentIsAffectedToCorrectCompanyOnCreate() : il faut valider qu'on affecte à la bonne SportCompany lors de la création d'un équipement. Il faudra pour cela avoir 2 sport company en base et valider qu'on affecte bien le matériel à la compagnie de l'utilisateur connecté.: testDAO non fait, délégué au framework de persistance
  - testEquipmentIsAffectedToCorrectCompanyOnCreate() : délégué au framework de persistance
  - testCannotCreateEquipmentCategoryWithSameName() : délégué au framework de persistance
  - testCanSaveEquipmentByCallingPersistenceLayer();
  - testCanDeleteEquipmentByCallingPersistenceLayer()
  - testCanFindAllEquipmentByCallingPersistenceLayer()

#### Tests unitaires DAO

- Les contraintes sont exprimées et implémentées dans les services. Nous utiliserons les méthodes DAO disponibles dans le DAO générique ce qui nous dispense de tests unitaires sur cette partie.

#### Tests intégration IHM

- testSportPartnerCanAccessToSportAdminPage()
  - ouvrir homepage

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

- se connecter avec compte client
  - valider présence lien édition du matériel
- testSportPartnerCanOnlyAccessToHisOffer()
  - ouvrir homepage
  - se connecter avec compte partenaire sport
  - cliquer lien édition du matériel
  - valider que le partenaire ne voit que son offre
- testCustomerCannotAccessToSportAdminPage()
  - ouvrir homepage
  - se connecter avec compte client
  - valider pas de lien édition du matériel
  - charger url en hardcoded de la page d'édition matériel
  - valider erreur 500
- testMechanicPartnerCannotAccessToSportAdminPage()
  - ouvrir homepage
  - se connecter avec mecanic partner
  - valider pas de lien édition du matériel
  - charger url en hardcoded de la page d'édition matériel
  - valider erreur 500
- testSportPartnerCanAddEquipementCategory()
  - Scénario standard avec compte partenaire sport
  - Valider message de confirmation
- testSportPartnerCannotAddEquipementCategoryWithoutName()
  - Scénario standard avec compte partenaire sport
  - Valider message d'erreyr
- testSportPartnerCannotAddEquipementCategoryWithoutDescription()
  - Scénario standard avec compte partenaire sport
  - Valider message d'erreur
- testSportPartnerCanListEquipementCategory()
  - Scénario standard avec compte partenaire sport
  - Valider message de confirmation
- testSportPartnerCanDeleteEquipementCategory()
  - Scénario standard avec compte partenaire sport
  - Valider message de confirmation

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

- testSportPartnerCanUpdateEquipementCategory()
  - Scénario standard avec compte partenaire sport
  - Valider champs préchargés
  - Valider message de confirmation
- testSportPartnerCannotUpdateEquipementCategoryWithEmptyNameAndDescription()
  - Scénario standard avec compte partenaire sport
  - Valider message d'erreur

Same integration test for equipment (CRUD and no empty name or description)

## V2 : Sport / Edition matériel

Cette fonctionnalité permettra aux utilisateurs partenaires sport d'éditer la liste du matériel à préparer.

### IHM

La page sera accessible via un lien placé sur la page d'accueil après s'être identifié comme partenaire sport.

En cliquant sur le lien, on affichera pour chaque équipement la quantité à préparer pour chaque date d'arrivée. Le périmètre de date est à partir du jour courant + 7 jours soit une semaine complète.

Un bouton "Show all" sera disponible pour rafraichir la page en affichant tout le matériel à préparer de la date la plus ancienne à la date la plus lointaine disponible pour les bookings.

L'édition du matériel se fera par impression au format PDF en utilisant le navigateur, il n'y a donc pas d'éditeur de fichier PDF à utiliser. Il faut indiquer en italique ou clairement sous forme d'aide à l'utilisateur qu'il peut imprimer cette page web en utilisant son navigateur web.

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

Il faut afficher un tableau avec le plugin js datatables contenant les colonnes suivantes : Date, Equipement, Quantité. Pour la colonne équipement, on affichera les champs suivants de l'entité Equipment :

- Category.name
- PeriodService.name
- quality
- size

## Service

Création d'un dto SportPartnerListing contenant :

- Equipment equipment
- Date startDate
- Integer quantity

Utilisation du service ISportPartnerService

Ce service aura une dépendance avec le DAO décrit ci-dessous.

```
/**
 * Retourne la quantité de matériel de sport à préparer par date
 * Si startDate null, appelle getSportPartnerListingTo()
 * Si endDate null, appelle getSportPartnerListingFrom()
 */
List<SportPartnerListing> getSportPartnerListingBetween(Date startDate, Date endDate);
```

```
/**
 * Retourne la quantité de matériel de sport à préparer par date
 * à partir de @startDate à J+7
 * Si startDate null, appelle getSportPartnerListingForComingWeek()
 */
List<SportPartnerListing> getSportPartnerListingFrom(Date startDate);
```



	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

```
/**
 * Retourne la quantité de matériel de sport à préparer par date
 * à partir d'aujourd'hui inclus à @endDate
 * Si endDate null, appelle getSportPartnerListingForComingWeek()
 */
List<SportPartnerListing> getSportPartnerListingTo(Date endDate);
```

```
/**
 * Retourne la quantité de matériel de sport à préparer par date
 * à partir d'aujourd'hui inclus à J+7
 */
List<SportPartnerListing> getSportPartnerListingForComingWeek();
```

## Persistence

Dans interface DAO ISportPartnerDAO – création signature suivante

```
List<SportPartnerListing> getSportPartnerListingBetween(Date startDate, Date endDate);
```


Pour rappel le DTO possède les attributs suivants : Equipment e, Date date, Integer quantity.

On utilisera les entités :

- Equipment : récupération des équipement
- Booking : récupération des dates de début de séjour pour chaque Guest ayant un GuestEquipment
- Guest : utilisé pour récupérer les liens GuestEquipment
- GuestEquipment : chaque occurrence comptera pour une quantité de Equipment correspondant à préparer

Exemple du calcul à réaliser :

- 1 Booking avec 2 personnes à partir du 10 février.

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

- a. Une personne a réservé des skis
  - b. Une personne a réservé un snowboard
  - c. Les deux personnes ont pris une combinaison complète
- 1 Booking avec 4 personnes à partir du 11 février.
  - a. 2 personnes ont réservé des skis
  - b. 1 personne a réservé du matériel de ski de fond
  - c. 1 personne n'a pas pris de matériel de ski
  - d. 2 personnes ont pris une combinaison complète
  - e. 2 personnes ont pris uniquement des après-skis.

Dans ce cas, le retour du DAO sera le suivant :

Equipment	Date	Quantity
Ski	10/02/2013	1
SnowBoard	10/02/2013	1
Combinaison	10/02/2013	2
Ski	11/02/2013	2
Ski de fond	11/02/2013	1
Combinaison complète	11/02/2013	2
Après-ski	11/02/2013	1

Pour simplification d'exemple, la colonne Equipment contient uniquement le nom de l'Equipment mais le dto contient bien l'objet avec tous les attributs chargés : AgeCategory, Category, description etc)

Au niveau de l'accès aux DAOs il faut donc :

1. Joindre les entités Booking, Guest, GuestEquipment, Equipment
2. Grouper par Booking.startDate et Equipment

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

3. Faire un count sur GuestEquipment
4. Order by Booking.startDate

## Tests

Tests unitaires service

- GetSportPartnerBetween
  - testCallListingToWhenStartDateNull() : Si startDate null, appelle getSportPartnerListingTo()
  - testCallListingFromWhenEndDateNull() : Si endDate null, appelle getSportPartnerListingFrom()
  - testCallListingDAOToGetSportListing() : Retourne la quantité de matériel de sport à préparer par date en appelant DAO – test mock appelle bien dao
- List<SportPartnerListing> getSportPartnerListingFrom(Date startDate);
  - testListingForComingWeekFillterToNextWeek() : Vérifier qu'on appelle DAO avec paramètre endDate = J+7
  - testListingForComingWeekCallDAOFromTodayIncluded() : fait dans testListingForComingWeekFillterToNextWeek()
  - testListingForComingWeekCallDAOTodayPlusSevenDays() : fait dans testListingForComingWeekFillterToNextWeek()
  - testIfStartDateAndEndDateFilledCallPersistenceWithThoseParameters()

Tests unitaires DAO

- testNoMatchingCriteriaReturnEmptyList() : valider que si périmètre date sans données (exemple startDate 1900-01-01 et endDate 1900-01-02), on retourne bien une liste sans objets et pas null
- testCountOneQuantityPerGuestEquipment() : valider qu'on compte bien un equipment à préparer par guestEquipment et pas 0 ou 2.
- testIfOneMatchingBookingReturnOneListing()
- testOneBookingWithOneGuestAndTwoEquipmentsReturnTwo()
- testOneBookingWithTwoGuestsAndSameEquipmentReturnOneRowAndTwoQuantity()
- testTwoBookingsWithOneGuestsHavingSameEquipmentAndOneAdditionnalForOneReturnTwoEquipmentOneWithQuantityOneAndCommonWithTwo()

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

- testNoStartDateCriteriaWhenStartDatelsNull() : responsabilité du service de ne pas appeler avec paramètres null
- testNoEndDateCriteriaWhenEndDatelsNull() : responsabilité du service de ne pas appeler avec paramètres null
- testNoDateCriteriaWhenNullStartDateAndEndDate() : responsabilité du service de ne pas appeler avec paramètres null

#### Tests intégration IHM

- testSportPartnerCanAccessSportListingPage() : Valider l'accès à la page avec un compte partenaire sport, fait dans testSportPartnerGetEquipmentListing()
  - Ouvrir homepage
  - Se connecter avec compte partenaire sport
  - Cliquer sur lien page "Matériel à préparer"
  - Valider affichage page en checkant URL et la présence du tableau
- testMechanicPartnerCannotAccessSportAdminPage() : Valider pas d'accès au lien vers la page d'affichage de stock en se connectant avec un compte partenaire mécanique : déjà fait pour administration offre
  - Ouvrir homepage
  - Se connecter avec compte partenaire mécanique
  - Valider pas de lien Matériel à préparer
  - Changer l'URL en dur pour celle du lien Matériel à préparer
  - Valider message d'erreur / erreur HTTP 500
- testSportPartnerGetEquipmentListing() : Valider l'affichage du tableau avec le matériel à préparer
  - ouvrir homepage
  - se connecter avec compte partenaire sport
  - cliquer sur le lien édition du matériel à préparer
  - valider affichage du tableau et du contenu
- testSportPartnerGetFullEquipmentListing() : appuyer sur show all et valider plus de lignes affichées


## V2: Sport / Remise matériel

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

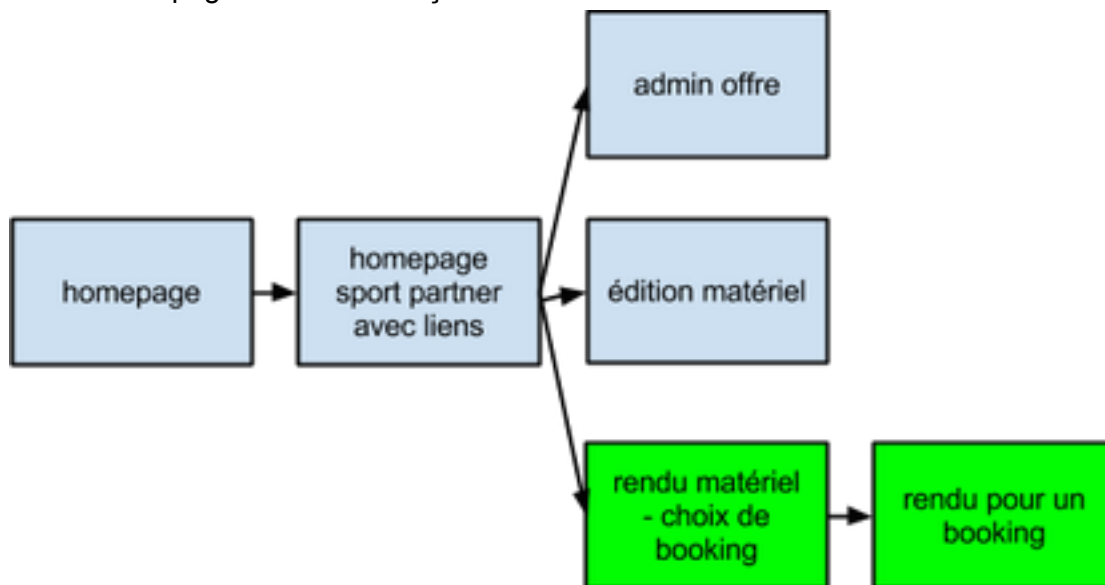
Cette fonctionnalité doit permettre à un partenaire de sport de valider le retour de matériel par des clients. Concrètement, les clients sont soit directement en face du partenaire et lui rendent le matériel et la personne saisit les informations directement ou bien le partenaire choisit d'accéder à l'application une fois par jour.

Le fonctionnement reste néanmoins le même et la page doit être pensée dans cette optique :

- un client arrive et donne son numéro de réservation ou son nom
- le partenaire recherche la réservation, il doit pouvoir la trouver
- le partenaire accède uniquement aux informations nécessaires au rendu du matériel : pour chaque personne de la réservation, le matériel de sport emprunté avec la date d'emprunt, et pour chaque équipement il peut saisir la date/heure de retour et si l'état du matériel rendu est accepté.

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

L'accès à la page se fait de la façon suivante :



pages à développer en vert

## IHM

Deux pages sont à développer :

1. Choix du booking
2. Rendu pour un booking

	<b>Project ProxStat- Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

### Choix du booking

Cette page doit afficher un tableau avec les informations suivantes :

- Numéro de réservation
- Nom et prénom du client
- Nombre de personnes
- Date de début et fin de réservation
- Rendu de matériel qui va contenir une icone permettant d'accéder à la page de rendu de matériel pour le booking de la ligne

Par défaut on chargera les booking dont la date de fin est supérieure ou égale à la date du jour.  
Un bouton showAll permettra de charger toutes les réservations.

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0


### Rendu pour un booking

Cette page va afficher les informations du booking sélectionné à la page précédente. On affichera pour chaque Guest, un tableau avec la liste d'équipement et le partenaire pourra saisir la date de retour, l'acceptation ou non du retour et éventuellement saisir un commentaires.

On remplira le champ return date avec la date/heure du jour par défaut et la checkbox pour la colonne accepted sera décoché (sinon forcer le false dans factory ou bien afficher décoché si valeur nulle, plus propre de mettre faux à la création dans factory).



Si trop compliqué de tout sauvegarder en une seule fois, faire des boutons save sur chaque ligne.





	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

Booking ID #####  
Customer name + surname  
Booking Start date & end date  
Nb of guests

Guest name & surname

Equipment	return date	accepted	comments
Ski	1/1/2009 	<input type="checkbox"/> Checkbox	
Combinaison	1/1/2009 	<input type="checkbox"/> Checkbox	

Guest name & surname


Equipment	return date	accepted	comments
Snowboard	1/1/2009 	<input type="checkbox"/> Checkbox	broken
Combinaison	1/1/2009 	<input type="checkbox"/> Checkbox	

## Service

Création du service ISportPartnerService

/\* Retourne tous les booking dont date de fin à partir de la date du jour incluse

\* utilisation DAO findByProperty ou Criteria en mettant un critère de date sur date de fin

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

```
List<Booking> findIncomingReturns();
```

```
/* Retourne tous les booking, à remonter du DAO Booking
List<Booking> findAll();
```

```
/* Save/update booking, à remonter du DAO Booking
Booking save(Booking)
```

## Persistence

Cette fonctionnalité impacte principalement l'entité GuestEquipment sur laquelle il faut renseigner les attributs returnDate et returnStateAccepted. Il ne faut pas créer/supprimer d'entité.

On ajoutera un attribut comments de type String dans l'entité GuestEquipment afin de permettre au partenaire de saisir des informations complémentaires en cas de retour invalidé.

Utilisation des DAOs existant save, findAll et findByCriteria dans genericDao.

## Tests

Tests unitaires service

- testFindIncomingReturnDontReturnBookingWithEndDateBeforeToday()
- testFindIncomingReturnSearchFromCurrentDateNotTomorrow()

Tests unitaires DAO


- réutilisation du DAO générique donc pas de nouveaux tests

Tests intégration IHM

- testSportPartnerCanAccessToReturnEquipmentPage() : accès validé dans tests précédents
  - ouvrir homepage

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

- se connecter avec sport partner
  - valider présence lien retour équipement
- testMechanicPartnerCannotAccessToReturnEquipmentPage() : faits dans tests précédents - autre fonctionnalité
  - ouvrir homepage
  - se connecter avec mechanic partner
  - valider pas de lien retour équipement
  - charger url en dur pour accéder retour équipement
  - valider erreur HTTP 500 restricted access
- testCustomerCannotAccessToReturnEquipmentPage() : faits dans tests précédents - autre fonctionnalité
  - ouvrir homepage
  - se connecter avec client
  - valider pas de lien retour équipement
  - charger url en dur pour accéder retour équipement
  - valider erreur HTTP 500 restricted access
- testReturnPageDisplayBookingStandardInformation()
  - ouvrir homepage
  - se connecter avec sport partner
  - cliquer sur lien retour équipement
  - valider affichage tableau avec id, start/end date, nom client et nombre de guests
- testDefaultReturnDatelsCurrentDateByDefault() : fait avec choix datepicker
  - ouvrir homepage
  - se connecter avec sport partner
  - cliquer sur lien retour équipement
  - valider date de retour si vide, par défaut est celle du jour
- testCanReturnGuestEquipmentForOneBooking()
  - ouvrir homepage
  - se connecter avec sport partner
  - cliquer sur lien retour équipement
  - choisir le premier booking
  - changer une date
  - mettre un booléen à faux
  - mettre un commentaires

	<b>Project ProxStat- Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

- cliquer sur save
- valider message de confirmation affiché

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

## V2 : Remontée mécanique/ Administrer forfait

Cette fonctionnalité doit permettre aux utilisateurs de profil Partenaire remontée mécanique de configurer leur offre de forfait disponible en ligne. Le service de remontées mécaniques pourra décrire pour chaque saison l'ensemble des forfaits possibles aussi bien pour le ski alpin, le ski de fond, le VTT ou encore les piétons avec les périodes concernées.

### IHM

Dans le workflow, l'utilisateur est sur la page d'accueil et après une identification réussie, il est directement routé sur une page d'accueil de backoffice lui présentant les actions suivantes :

- - Administrer mon offre
- - Consultation et édition des forfaits à présenter
- 

Les pages à développer dans cette fonctionnalité seront accessibles en cliquant sur le lien "Administrer mon offre".

Par défaut en cliquant sur le lien "Administrer mon offre" on arrive sur la page CRUD Forfait.

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

DataTable affichant les types de forfait avec leur coût suivant la période choisie.

Ecran :

Type de forfait	Tarif/Période	Supprimer	Enregistrer
Ski alpin	<ul style="list-style-type: none"> <li>• 20€/vacances février</li> <li>• 15€/vacances Avril</li> </ul>	<u>Delete</u>	Save
Ski de fond		<u>Delete</u>	Save
VTT		<u>Delete</u>	Save
Piétons		<u>Delete</u>	Save

Add a new Pass

## Service

Création interface et implémentation service

IAdministrationPassMechanicPartnerService

+ création des factory si nécessaire pour utilisation dans les controllers afin de passer des objets directement aux services

Signatures

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

```

/** Create or update
Pass savePass(Pass)

void deletePass(Pass)

List<Pass> findAllPassByMechanicPartner(int id_mechanicPartner)

Track saveTrack(Track)

void deleteTrack(Track)

List<Track> findAll()

```

## Persistence

Utilisation des méthodes existantes dans DAO génériques  
Entités : Pass, PeriodService (pour le price)

## Jeux de données

Ici inutile puisque justement on va pouvoir tester en insérant les données depuis le « formulaire », c'est-à-dire la datatable.

## Tests

Tests unitaires service

- testCannotCreatePassWithoutName() : test si champ null ou "" throws exception

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

- testCannotCreatePassWithoutDomain() : test si champ null ou "" throws exception
- testCannotCreatePassWithoutType() : test si champ null ou "" throws exception
- testCannotCreatePassWithoutPrice() : test si champ null ou throws exception
- testCanSaveOrUpdate()
- testCanListPass()
- testCanDeletePass()
- // idem pour les track


#### Tests unitaires DAO

- Les contraintes sont exprimées et implémentées dans les services. Nous utiliserons les méthodes DAO disponibles dans le DAO générique ce qui nous dispense de tests unitaires sur cette partie.


#### Tests intégration IHM

- testMechanicPartnerCanAccessToPasstAdminPage ()
  - ouvrir homepage
  - se connecter avec compte partenaire remontée mécanique
  - valider présence lien administrer les forfaits
- testMechanicPartnerCanOnlyAccessToHisOffer()
  - ouvrir homepage
  - se connecter avec compte partenaire remontée mécanique
  - cliquer lien administrer les forfaits
  - valider que le partenaire ne voit que son offre
- testCustomerCannotAccessToMechanicAdminPage()
  - ouvrir homepage
  - se connecter avec compte client
  - valider pas de lien administrer les forfaits
  - charger url en hardcoded de la page d'édition matériel
  - valider erreur 500
- testSportPartnerCannotAccessToMechanicAdminPage()
  - ouvrir homepage
  - se connecter avec compte sport partner
  - valider pas de lien administrer les forfaits
  - charger url en hardcoded de la page d'édition matériel



	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

- valider erreur 500
- testMechanicPartnerCanAddPass()
  - Scénario standard avec compte partenaire remontée mécanique
  - Valider message de confirmation
- testMechanicPartnerCannotAddPassWithoutName()
  - Scénario standard avec compte partenaire remontée mécanique
  - Valider message d'erreur
- testMechanicPartnerCannotAddPassWithoutDomain()
  - Scénario standard avec compte remontée mécanique
  - Valider message d'erreur
- testMechanicPartnerCannotAddPassWithoutType()
  - Scénario standard avec compte remontée mécanique
  - Valider message d'erreur
- testMechanicPartnerCannotAddPassWithoutPrice()
  - Scénario standard avec compte remontée mécanique
  - Valider message d'erreur
- testMechanicPartnerCanListPass()
  - Scénario standard avec compte remontée mécanique
  - Valider message de confirmation
- testMechanicPartnerCanDeletePass()
  - Scénario standard avec compte remontée mécanique
  - Valider message de confirmation
- testMechanicPartnerCanSaveOrUpdatePass()
  - Scénario standard avec compte remontée mécanique
  - Valider champs préchargés
  - Valider message de confirmation

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

## V2 : Remontée mécanique/ Edition des forfaits

Cette fonctionnalité doit permettre aux utilisateurs de profil Partenaire remontée mécanique d'éditer les forfaits qui ont été réservé. Le service de remontées mécaniques pourra donc éditer les forfaits en vérifiant bien la composition de la famille.


### IHM

La page sera accessible via un lien placé sur la page d'accueil après s'être identifié comme partenaire remontées mécaniques.

En cliquant sur le lien, on affichera pour chaque type de forfait (ski alpin, fond, etc ...) la quantité à préparer pour chaque date d'arrivée. Le périmètre de date est à partir du jour courant + 7 jours soit une semaine complète.

Un bouton "Show all" sera disponible pour rafraichir la page en affichant tous les forfaits à préparer de la date la plus ancienne à la date la plus lointaine disponible pour les bookings.

L'édition du matériel se fera par impression au format PDF en utilisant le navigateur, il n'y a donc pas d'éditeur de fichier PDF à utiliser. Il faut indiquer en italique ou clairement sous forme d'aide à l'utilisateur qu'il peut imprimer cette page web en utilisant son navigateur web.

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

Ecran :

Type de forfaits	Date d'arrivée	Détail personne (nom, <u>prenom</u> )	Numéro de réservation
Ski alpin	01 /03/2013	Tarik <u>Djebien</u>	1234
Ski de fond	01 /03/2013	Eric <u>Rakotobe</u>	1234
Ski alpin	01 /03/2013	Kevin <u>Sansen</u>	2345

Show All

Tableau développé sous forme de DataTable avec chaque colonne triable, afin de voir en un clic le volume par type de forfaits, ou aussi par famille

## Service

Création d'un dto MechanicPartnerListing contenant :

- Pass pass
- Date startDate

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

- Booking booking ->Guest guest

Utilisation du service IAdministrationFlatMechanicPartnerService

Ce service aura une dépendance avec le DAO décrit ci-dessous.

/\*\*

\* Retourne la quantité de forfaits à préparer par date

\* Si startDate null, appelle getMechanicPartnerListingTo()

\* Si endDate null, appelle getMechanicPartnerListingFrom()

\*/

List<MechanicPartnerListing> getMechanicPartnerListingBetween(Date startDate, Date endDate);

/\*\*

\* Retourne la quantité de forfait à préparer par date

\* à partir de @startDate à J+7

\* Si startDate null, appelle getMechanicPartnerListingForComingWeek()

\*/

List<MechanicPartnerListing> getMechanicPartnerListingFrom(Date startDate);

/\*\*

\* Retourne la quantité de forfaits à préparer par date

\* à partir d'aujourd'hui inclus à @endDate

\* Si endDate null, appelle getMechanicPartnerListingForComingWeek()

\*/

List<MechanicPartnerListing> getMechanicPartnerListingTo(Date endDate);

/\*\*

\* Retourne la quantité de forfait à préparer par date

\* à partir d'aujourd'hui inclus à J+7

\*/

List<MechanicPartnerListing> getMechanicPartnerListingForComingWeek();

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

## Persistence

On utilisera les entités :

- Pass : récupération des forfaits (type de forfait,...)
- Booking : récupération des dates de réservations pour chaque Guest
- Guest : utilisé pour récupérer les infos (nom, prenom) (chaque Guest est un forfait à préparer, puisque chaque personne aura un forfait, en tout cas chaque Guest attaché à un pass)

## Jeux de données

Guest et booking ayant Pass en paramètres

## Tests

Tests unitaires service

- testCallListingToWhenStartDateNull() : Si startDate null, appelle getMechanicPartnerListingTo()
- testCallListingFromWhenEndDateNull() : Si endDate null, appelle getMechanicPartnerListingFrom()
- testCallListingDAOToGetSportListing() : Retourne la liste de forfaits à préparer
- testListingFromStartDateFilterToNextWeek() : Vérifier qu'on appelle DAO avec paramètre endDate = J+7
- testCallListingForWhenStartDateNull() : Si startDate null, appelle getMechanicPartnerListingForComingWeek()
- testListingToStartFromTodayIncluded() : valider appel DAO avec startDate = today

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

- testCallListingForComingWeekWhenEndDateNull() : Si endDate null, appelle getMechanicPartnerListingForComingWeek()
- testListingForComingWeekCallDAOFromTodayIncluded()
- testListingForComingWeekCallDAOTodayPlusSevenDays()

#### Tests unitaires DAO

- testNoMatchingCriteriaReturnEmptyList() : valider que si périmètre date sans données (exemple startDate 1900-01-01 et endDate 1900-01-02), on retourne bien une liste sans objets et pas null
- testCountOneQuantityPerGuestEquipment() : valider qu'on compte bien un pass à préparer par guest.
- testNoStartDateCriteriaWhenStartDatesIsNull()
- testNoEndDateCriteriaWhenEndDatesIsNull()
- testNoDateCriteriaWhenNullStartDateAndEndDate()

#### Tests intégration IHM

- testMechanicPartnerCanAccessPassAdminPage() : Valider l'accès à la page avec un compte partenaire partenaire remontée mécanique
  - Ouvrir homepage
  - Se connecter avec compte partenaire remontée mécanique
  - Cliquer sur lien page "Forfait à préparer"
  - Valider affichage page en checkant URL et la présence du tableau
- testSportPartnerCannotAccessPassAdminPage() : Valider pas d'accès au lien vers la page d'affichage des forfaits à préparer en se connectant avec un compte partenaire sport
  - Ouvrir homepage
  - Se connecter avec compte partenaire sport
  - Valider pas de lien Forfaits à préparer
  - Changer l'URL en dur pour celle du lien Forfait à préparer
  - Valider message d'erreur / erreur HTTP 500
- testMechanicPartnerGetPassListing() : Valider l'affichage du tableau avec les forfaits à préparer

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

- ouvrir homepage
- se connecter avec compte partenaire remontée mécanique
- cliquer sur le lien édition des forfaits à préparer
- valider affichage du tableau et du contenu

## V3 : Modération des avis

Cette fonctionnalité est prévue pour les administrateurs du site. Elle leur permet d'avoir la visibilité sur tous les avis que les utilisateurs ont mis sur le site. Il leur est possible de supprimer tout avis qu'il juge inapproprié. Le commentaire est alors supprimé et n'est plus visible des autres utilisateurs.

### IHM

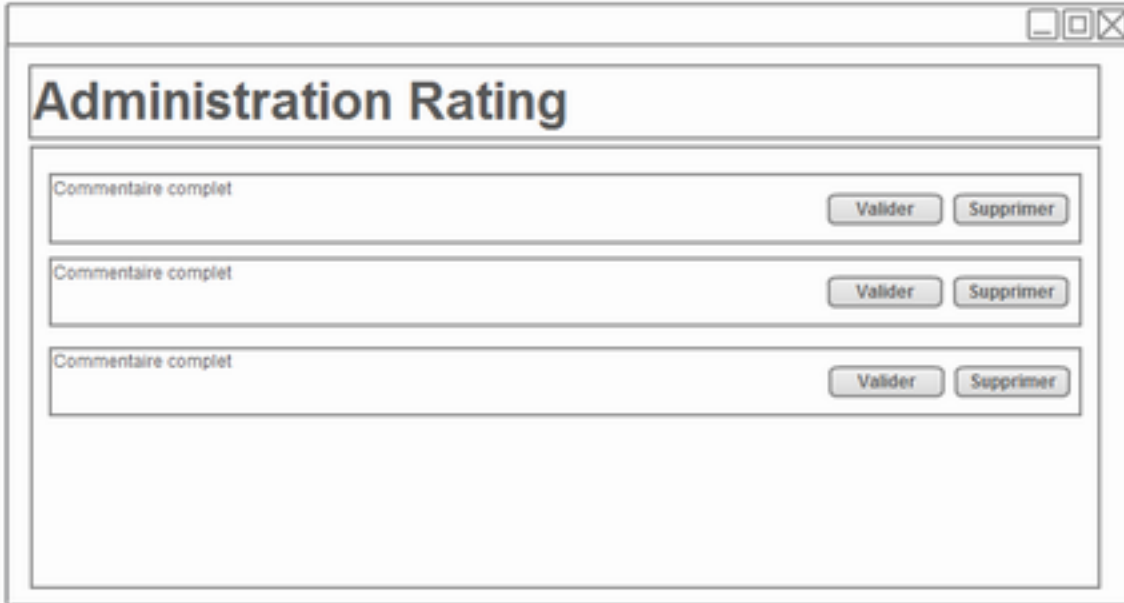
La page sera accessible via un lien placé sur la page d'accueil après s'être identifié comme "Admin".

En cliquant sur le lien, on affichera une liste des commentaires ajoutés par les utilisateurs. Sur chaque commentaire, on retrouvera un bouton "supprimer".

Si l'admin appuie sur "supprimer", le commentaire est retiré et n'est plus visible sur le site.

Ecran :

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0



The screenshot displays a web application window titled "Administration Rating". Inside the window, there is a table with three rows. Each row contains a text input field with the placeholder text "Commentaire complet" and two buttons labeled "Valider" and "Supprimer".

Tableau développé sous forme de DataTable.

## Service

Utilisation du service existant : IRatingService

Développement des méthodes :

```
/* Retourne la liste de tous les commentaires à valider */
private List<Rating> findAllRatingsToValidate() { }
```

```
/* Supprime le commentaire afin qu'il ne soit pas publié sur le site */
private void deleteRatingSelected () { }
```



	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

## Persistence

On utilisera l'entité : rating pour la récupération des commentaires qui ont été créés

## Jeux de données

## Tests


### Tests unitaires service

- testRatingsToValidateAreListing() :Vérifie que tous les commentaires qui sont à valider sont affichés dans la liste.
- testRatingsValidateArePublished () : Les commentaires validés sont publiés sur le site
- testRatingsNotValidateAreDelete () : Les commentaires non validés sont supprimer et n'apparaissent pas sur le site.
- testRatingsValidateOrDeletedAreNotListing () : Les commentaires qui ont été validés ou supprimés n'apparaissent plus dans la liste des commentaires à valider
- testCallDaoToListingRatings () : appel à la DAO

### Tests unitaires DAO

### Tests intégration IHM

- testAdministrationRatingCanAccessPassAdminPage() : Valider l'accès à la page avec un compte admin
  - Ouvrir homepage
  - Se connecter avec compte admin
  - Cliquer sur lien page "Modération d'avis"

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

- Valider affichage page en checkant URL et la présence du tableau
- testOtherUserCannotAccessPassAdminPage() : Valider pas d'accès au lien vers la page d'affichage des avis à valider si ce n'est pas un admint
  - Ouvrir homepage
  - Se connecter avec compte autre que admin
  - Valider pas de lien "Modération d'avis"
  - Changer l'URL en dur pour celle du lien "Modération d'avis"
  - Valider message d'erreur / erreur HTTP 500
- testAdminGetRatingListing() : Valider l'affichage du tableau avec la liste des avis
  - ouvrir homepage
  - se connecter avec compte admin
  - cliquer sur le lien "Modération d'avis"
  - valider affichage du tableau et du contenu

## V3 : Promotions

Cette fonctionnalité va permettre aux admin de mettre en place des promotions sur les produits proposés sur leur site.

### IHM

Un écran uniquement accessible aux admin va leur permettre de positionner des promotions sur les articles de leur choix.

Pour cela, on va avoir la liste de tous les produits proposés sur le site avec le prix actuel du produit ainsi qu'une colonne réservé au prix après promotions. Si la colonne promotion est vide, ou null c'est le prix normal qui s'affiche. S'il y a un prix dans la colonne promotions, le prix normal s'affiche barré et le prix après promotions est affiché en rouge.

Cet page est accessible via un bouton sur la page d'accueil d'un compte admin.

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

## Service

Création du service : IAdministrationPriceService

## Persistence

On utilisera les entités suivantes :

- price : pour retrouver les prix avant et après promotions
- periodeService : pour faire le lien avec les différents produits (équipement, forfaits, appartements)
- on utilisera aussi toutes les entités en rapport avec les produits

## Jeux de données

## Tests

Tests unitaires service

Tests unitaires DAO

Tests intégration IHM

- testAdministrationPriceCanAccessPassAdminPage() : Valider l'accès à la page avec un compte admin

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

- Ouvrir homepage
- Se connecter avec compte admin
- Cliquer sur lien page “Promotions”
- Valider affichage page en checkant URL et la présence du tableau
- testOtherUserCannotAccessPassAdminPage() : Valider pas d’accès au lien vers la page d’affichage des promotions si ce n’est pas un admin
  - Ouvrir homepage
  - Se connecter avec compte autre que admin
  - Valider pas de lien “Promotions”
  - Changer l’URL en dur pour celle du lien “Promotions”
  - Valider message d’erreur / erreur HTTP 500

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

## V3 : Purge

L'exigence non-fonctionnelle sur ce projet consiste à faciliter la purge des données dans l'application en définissant un périmètre des données à supprimer sur un intervalle.

Le périmètre défini au départ est celui des commandes sur 3 ans. Toutes les commandes et les liens éventuels de plus de 3x365 jours seront supprimés de la base de données. L'archive de la base de données est effectué par le client qui pourra recharger ses données pour des besoins de consultation d'archivages.

### IHM

Il n'y a pas d'accès utilisateur pour cette fonctionnalité, elle sera automatisée toutes les semaines.

### Service

Le service supprimera de la base de données

- Booking dont bookingDate supérieure 365jx3
  - les payment liés aux bookings
  - les guests et GuestEquipment associées aux bookings à supprimer
- Rating dont date supérieure 365jx3
- Les saisons ne sont pas supprimés car elles supprimeraient des servies qui peuvent rester en place plus de 3 ans et qui plus est ne surcharge pas la volumétrie de la base de données

dans AdministrationService

service : void purgeData();

### Persistence

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

Pas de modification de la base de données, on utilisera les entités décrites ci-dessus.

## Jeux de données

Test DAO

Utilisation des données en base avec rollback après passages des tests unitaires

Test service : mock du DAO pour retourner les cas mentionnés ci-dessous

## Tests

- testDontPurgeBookingWithLessThanThreeYearsOfHistory : ne supprime pas un booking avec crée dans les 3 années en cours - cas date du jour
- testPurgeBookingWithMoreThanThreeYearsOfHistory : supprime un booking avec booking date il y a 4 ans
- testPurgeTwoBookingsWithMoreThanThreeYearsOfHistory : supprime 2 bookings avec booking date il y a 4 ans
- testDontPurgeRatingWithLessThanThreeYearsOfHistory : ne supprime pas un rating avec crée dans les 3 années en cours - cas date du jour
- testPurgeRatingWithMoreThanThreeYearsOfHistory : supprime un rating avec booking date il y a 4 ans
- testPurgeTwoRatingWithMoreThanThreeYearsOfHistory : supprime 2 ratings avec booking date il y a 4 ans

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

## V3 : Moteur de recherche et fiche client

Une moteur de recherche client sera implémenté afin de pouvoir récupérer les informations clients ainsi que ces anciennes réservations (historique). Cette recherche pourra se faire à partir du nom du client ou des autres informations le concernant. Cette écran donnera également accès à la fiche client qui reprendra toutes les informations le concernant.

### IHM

L'écran sera accessible à partir du compte admin et sera disponible dans le menu sous le nom "Recherche client".

L'écran sera composé de trois parties:

- la partie "paramètres de recherche" où l'admin pourra renseigner le nom et le prénom sur lesquels il souhaite faire la recherche du client
- la partie "informations client" qui reprendra les informations générales du client (adresse, numéro de téléphone ...), autrement dit la fiche client, avec un lien sur la fiche amenant vers une page où se trouvera la datatable affichant toutes les réservations que le client a effectué
- la partie "historique de réservation" qui reprendra toutes les réservations que le client a effectué. C'est à dire une data table avec les informations des différents booking du client (date, montant, infos sur la réservation)

### Service

Utilisation des services existants suivants:

- IBookingService : pour retrouver les réservations liées à un client
- ICustomerService : pour retrouver les informations concernant un client

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

Développement des méthodes :

- Dans ICustomerService :

/\* Retourne le client correspondant à son nom et prenom \*/

Customer findByNameAndSurname(String name, String surname);

- Dans IBookingService

/\* Retourne les réservations que le client a déjà effectué \*/

List<Booking> findBookingByCustomer (Customer customer) ;

## Persistence

On utilisera les entités suivantes :

- USER : pour retrouver le client en fonction de son login
- CUSTOMER : pour rechercher toutes les informations le concernant
- ADRESS : pour ses coordonnées géographiques
- BOOKING : pour avoir l'historique de ses réservations

## Jeux de données

DAO : voir jeux de données requis ci-dessous

Service : mock du DAO avec retour des cas de tests mentionnés ci-dessous

## Tests

Tests unitaires service

- testFindCustomerWithNameNull()




	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

- testFindCustomerWithSurnameNull()
- testFindCustomerWithNameAndSurnameNotInBdd()
- testFindCustomerWithNameAndSurnameInBdd
- testFindBookingByCustomerWithCustomerNull()
- testFindBookingByCustomerWithCustomerWithBooking()
- testFindBookingByCustomerWithCustomerWithoutBooking()

#### Tests intégration IHM

- testAdminCanAccessCustomerResearchPage() : Valider l'accès à la page avec un compte admin
  - Ouvrir homepage
  - Se connecter avec compte admin
  - Cliquer sur lien page "Recherche client"
- testSportPartnerCannotAccessAdminPage() : Valider pas d'accès au lien vers la page Recherche client en se connectant avec un compte partenaire sport
  - Ouvrir homepage
  - Se connecter avec compte partenaire sport
  - Valider pas de lien "Recherche client"
- testMechanicPartnerCannotAccessAdminPage() : Valider pas d'accès au lien vers la page Recherche client en se connectant avec un compte partenaire mechanic
  - Ouvrir homepage
  - Se connecter avec compte partenaire remontée mécanique
  - Valider pas de lien "Recherche client"
- testAdminResearchCustomer() : Valider la recherche d'un client donné par son nom et prenom
  - ouvrir homepage
  - se connecter avec compte admin
  - renseigner le formulaire avec nom et prenom puis valider
  - valider affichage de la fiche client

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

- testAdminResearchCustomerWithDataFalse() : Valider l’affichage d’un message erreur si aucune fiche client n’est remontée (test à décliner sur le nom, prenom, les deux champs à la fois, champs vide)
  - ouvrir homepage
  - se connecter avec compte admin
  - renseigner le formulaire avec nom et prenom non disponible en base puis valider
  - valider affichage d’un message de retour indiquant que le client n’existe pas
  
- testAdminGetBookingListingByCustomer() : Valider l’affichage du tableau avec les réservations pour un client donné
  - ouvrir homepage
  - se connecter avec compte admin
  - charger une fiche client
  - cliqué sur le lien vers ses booking
  - valider affichage de la dataTable avec les booking
  
- testAdminGetBookingListingByCustomerWithoutBooking() : Valider l’affichage d’un tableau vide si un client donné n’a aucune réservations qui lui sont attachées
  - ouvrir homepage
  - se connecter avec compte admin
  - charger une fiche client
  - cliqué sur le lien vers ses booking
  - valider affichage de la dataTable sans aucune ligne

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

## V3 : Statistiques

Un écran de statistiques sera mis à la disposition de l'admin afin qu'il puisse analyser les chiffres et améliorer ses ventes.

### IHM

L'écran de statistiques sera disponible uniquement pour un admin via un bouton figurant dans son menu et donnera accès aux tableaux de bord décrits ci-dessous.

On affichera sur une première lignes les indicateurs :

- nombre de réservations prévues
- montant moyen de commande

Seront ensuite disposés en dessous les différents graphiques de façon quadrillé.

Sauf pour les indicateurs, le périmètre filtré par défaut sera sur les 5 semaines précédentes mais on donnera la possibilité à l'utilisateur de choisir un périmètre pouvant aller jusqu'à 1 an avec des échelons suivants :


- par défaut (5 semaines) => l'axe X présentera les semaines
- trimestre précédent : date courante - 90 jours ) => l'axe X présentera les mois
- semestre précédent : date courante - 180 jours ) => l'axe X présentera les mois
- année : date courante - 365 jours ) => l'axe X présentera les mois

exemple


- filtre par défaut : S1, S2, S3, S4, S5 les valeurs par semaine

- filtre trimestre : mois1, mois2, mois3 les valeurs par mois


le group by est donc différent mais permettant de ne pas afficher des données trop détaillées.

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0


Nom	Besoin	X	Y	Valeurs	Règle
Indicateur - Nombre de réservations prévues pour la semaine prochaine	Connaître nombre réservations pour semaine suivante	Aucun - 1 seule valeur à afficher	Aucun - 1 seule valeur à afficher	count Booking.id  ex: 49	Périmètre = tout la base
Indicateur - Montant moyen de commande	Connaître montant moyen payé par client	Aucun - 1 seule valeur à afficher	Aucun - 1 seule valeur à afficher	avg(booking.totalPrice)  ex: 380 €	Périmètre = tout la base
Commandes - Nombre	Suivre activité par évolution nombre de commandes	Semaine ou mois en fonction du filtre	Nombre de commandes	count Booking.id  ex: 28	Graphique de suivi  ne pas prendre en compte si booking annulé  filtre sur booking dates

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

Commandes - CA	Suivre activité par CA	Semaine ou mois en fonction du filtre	Montant de commande	sum(Booking. totalPrice)  ex: 18000€	Graphique de suivi  ne pas prendre en compte si booking annulé  filtre sur booking date
Commandes - Marge	Savoir si rentable	Semaine ou mois en fonction du filtre	Marge en €	Booking.total Margin  ex: 3000€	Graphique de suivi
Services - Taux d'occupation appartement	Mesure pas de surcapacité	Semaine ou mois en fonction du filtre	Taux d'occupation	count(flat in booking) / count(* flat)  ex: 80%	Graphique de suivi  ne pas prendre en compte si booking annulé  filtre sur booking date

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

Services - Types d'appartemen t	Savoir quels appartements sont les plus réservés	aucun car pie chart	aucun car pie chart	service .name, count(bookin g.id)  where periodService = 'FLATTYPE'  group by service.name  ex: 40% 4 pers 60% 2 pers	Piechart
Services - Types d'équipement	Savoir quels équipements sont les plus demandés	aucun car pie chart	aucun car pie chart	service .name, count(bookin g.id)  where periodService = 'EQUIPME NT'  group by service.name  ex:  30% combi 30% ski 30% snow 10% gants	Piechart

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

Services - Types de forfait	Savoir quels forfaits sont les plus demandés	aucun car pie chart	aucun car pie chart	service.name, count(booking.id)  where periodService = 'PASS'  group by service.name  ex: 40% diamant 60% or	Piechart
Services - Note moyenne par service (rating)	Connaître le ressenti des clients	1 barre par service	Note : échelle de 0 à 5	service.name , avg(rating)  group by service.name  ex: Appartement 4 pers = 4/5  Appartement 2 pers = 3,5/5  Combinaison ski = 5/5	Bar chart
Client - Nombre de clients	Connaitre le nombre de nouveaux clients dans le temps	Semaine ou mois en fonction du filtre	Nombre de clients	count(distinct booking.customer.id)  ex: 20	Graphique de suivi

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

Client - Répartition age	Connaitre la répartition d'age des clients	aucun car pie chart	aucun car pie chart	g.ageCategory, count(g.guest) )  from Guest g  group by g.ageCategory  ex: 15% -12 ans 15% 12-18 20% 18-25 40% adulte 10% senior	Piechart
--------------------------	--	---------------------	---------------------	--	----------

Les suivis de trafic et des pages vues seront effectuées à l'aide de l'outil google analytics.  
Le compte est déjà configuré, le client pourra accéder aux données une fois son application disponible via internet.

#### Règles spécifiques

- Montant moyen de commande

#### **1/ solution la plus simple pour faciliter requetage :**

ne plus avoir champ transient mais le remplir au moment où on insère commande avec factory  
(et la faire somme des periodService dans Basket ou Booking) pour les champs

- totalPrice
- totalMargin

règle pour faire totalPrice

somme de

- flat.flatType.price.price
- for options : option => option.price.price
- for guests : guest => pass.price.price



	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

- for guests : guest => for guestequipment : ge => ge.equipment.price.price

règle pour totalMargin

somme de

- flat.flatType.price.discountPrice
- for options : option => option.price.discountPrice
- for guests : guest => pass.price.discountPrice
- for guests : guest => for guestequipment : ge => ge.equipment.price.discountPrice

dans ce cas la requête devient

select avg(totalPrice) from Booking b

## 2/ si dans entité

moyenne du montant de


pour chaque booking :somme de

- flat.flatType.price.price
- for options : option => option.price.price
- for guests : guest => pass.price.price
- for guests : guest => for guestequipment : ge => ge.equipment.price.price

## Service

Utilisation des services existants suivants:

- IBookingService : pour retrouver les réservations liées à un client et ainsi récupérer les dates et le matériel liés à cette date
- ICustomerService : pour retrouver les informations concernant un client

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

## Persistence

Les entités à utiliser sont décrites dans le tableau ci-dessous.

On réutilisera les DAOs créés pour le prototype qui utiliseront des objets DTO pour conserver les données à afficher.


## Jeux de données

DAO : utilisation des données en base qui sont déjà avec des types différents de réservations

Service : mock des DAO avec retours pour les cas de tests mentionnés ci-dessous

## Tests

- 1 test par graphique pour valider son affichage
- manuel : pour chaque graphe
  - valider nombre axe abscisse comme attendus
  - valider affichage légende
  - valider titre présent
  - valider unités présentes
  - valider échelle lisible
  - valider chiffres lisibles
- manuel : pour chaque piechart
  - valider les pourcentages
  - valider affichage légende
  - valider titre présent
- pour les indicateurs : valider les unités et les valeurs
- valider valeur à 0 si pas de données : pas de bug d'affichage

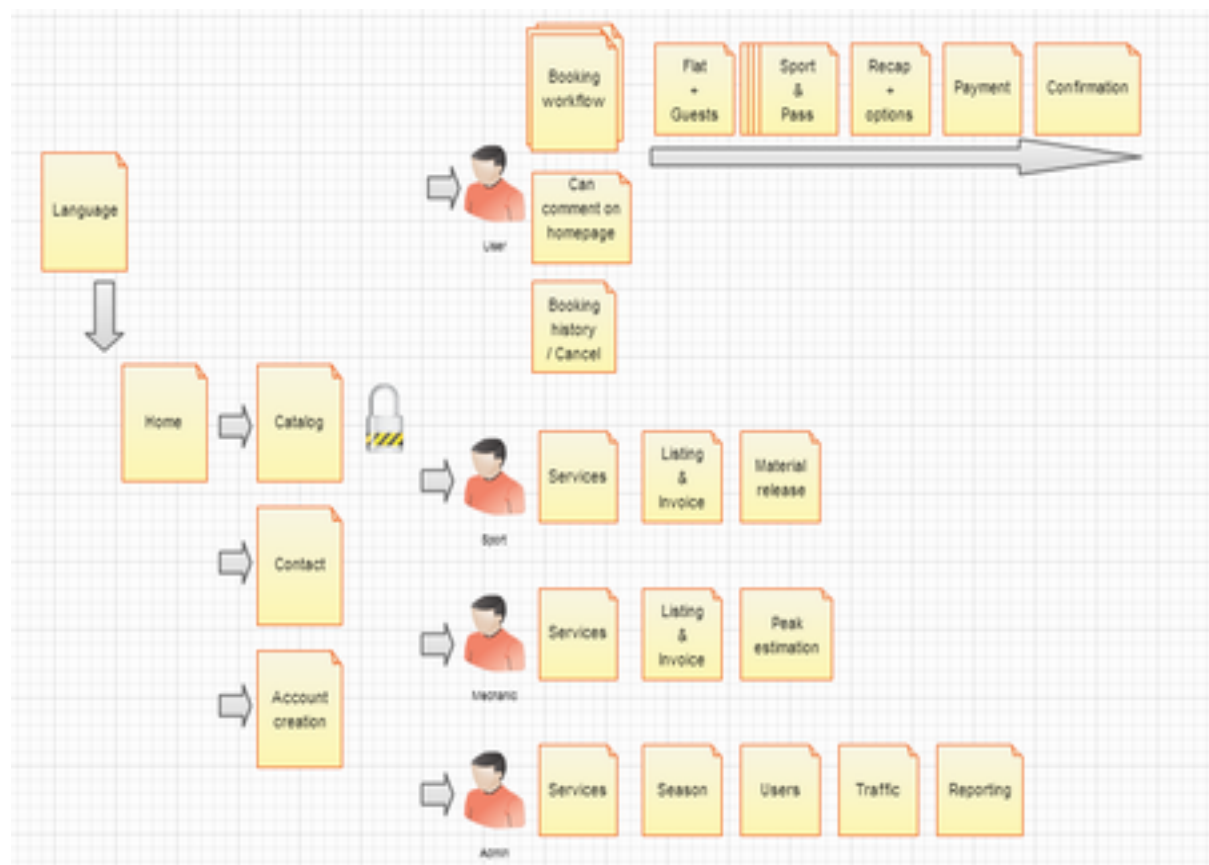
	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0


## Modèle de classe

Disponible dans google drive 02 Specs / ProxyStat-Gestion entity classes V1

## Arborescence des pages

	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0



	<b>Project ProxStat-Gestion</b>	Domaine
Latest update : 28 jan 2013	Spécifications détaillées	Current doc version : 1.0

## Pistes d'améliorations

- afficher tous les types de services dispo dans homepage  
(meme ceux pas dispo pour vraiment avoir vue catalogue)
- ajouter email au niveau PartnerUser pour qu'ils puissent recevoir matériel  
à préparer par email chaque jour de façon automatisée (sport + mécanique)