

Détails des choix technologiques, Description de l'application

Chaîne d'hôtels nationale située en France uniquement , et la réservation peut se faire jusqu'à J+60jrs.

Le choix du lieu de séjour se fera depuis un formulaire disponible sur toutes les pages du site.

Recherche d'un hotel par ville, département, ou par région à partir d'une carte de France.

Recherche de disponibilité en indiquant une date d'arrivée, un nombre de nuits et un nombre de chambres: single, double ou appartements.

Une fois tous les renseignements pris, il faut afficher la liste des hôtels correspondants et/ou disponibles.

Le client choisit un hôtel dans la liste ou, si un seul résultat le détail de l'hotel s'affiche directement.

Sur la page de l'hôtel, au moins une photo de l'hôtel, les coordonnées, un plan d'accès, les services de l'hôtel (piscine, restaurant, parking fermé, accès WiFi...), les commentaires des personnes ayant séjournées dans l'hôtel ainsi que des infos touristiques de la région.

Il faut indiquer les prix des différents types de chambres et prestations.

Le client peut avoir accès à la météo prévue dans la ville de son séjour. (webservice)

Un lien pour envoyer un mail à l'hôtel sera accessible.

Lors de la réservation, le client choisit le type de chambre qu'il désire (single, double, appartement ...).

L'application fonctionne sur Internet Explorer et sur Mozilla Firefox. Je me suis imposée cette contrainte car je pense qu'il est important que le site soit compatible sur les deux principaux navigateurs utilisés.

Détails des choix technologiques:

- **Choix du xslt** pour l'affichage des villes et des départements dans le formulaire de recherche. Ce choix se justifie par l'affichage des villes et département par ordre alphabétique, et en cas de requête asynchrone (utilisation d'ajax lors du choix d'un département pour l'affichage des ville correspondantes), on ne peut pas faire de requete à la base de données, ce qui d'ailleurs serait beaucoup plus long. J'ai donc décidé de créer un fichier xml qui contient toutes les données par rapport aux régions, département villes

Le fichier regions.xml:

Contient toutes les regions de France, et tous les noms et numero des departements contenant des hôtels, ainsi que toutes les villes contenant des hotels. Ce fichier contient également un lien vers un flux rss pour chaque région.

Structure du fichier:

```
<region id="n°" rss="lien rss">
  <departement numero="n°">
    <ville>
      nom de la ville
    </ville>
  </departement>
</region>
```

Il est utilisé:

- Via la classe de création du formulaire (Formulaire) pour écrire les départements, en utilisant Xslt pour trier les département par numéro croissant.

- Via la classe de resultatRecherche pour récupérer le rss correspondant à la région de l'hôtel (comme expliqué plus bas)

- Via la classe Recherche en cas de recherche par région (clic sur une région en page d'accueil)
- Via Ajax, et le DomJavascript:
 - pour l'affichage des villes dans le formulaire de recherche dans l'ordre croissant (appel via Ajax d'un fichier php qui effectue la transformation XSLT du fichier xml)
 - pour l'affichage des villes dans le formulaire correspondant au département sélectionné par l'utilisateur .
- Pour l'affichage du rss correspondant à la région où est situé l'hôtel. (cela évite d'avoir des champs redondants: tous les hôtels d'une même région ont le même rss)

- **Le flux rss:** d'après le département de l'hotel (= les deux premiers chiffres de son code postal), via php, le dom et xpath, récupération du lien correspondant à la région. Utilisation du dom + Xpath pour l'affichage des 6 dernières news. Cela aurait été mieux de le faire avec le parsing sax, mais dans ce cas, on ne peut gérer ni le nombre de réponses souhaitées, ni l'affichage dans les bulles (le début de la news apparaît lors du rollover sur le lien de la news). L'arbre doit donc être en mémoire.

- **Le livre d'or en xml:** Evite l'appel à la base de données, pour la lecture et l'écriture. Dans le dossier 'guestbook', le nom du fichier correspond à l'Id de l'hôtel. C'est un xml de la forme:

```
<message id="1">
  <date>Date en anglais</date>
  <texte>Message</texte>
  <prenom>Prénom</prenom>
  <nom>Nom</nom>
</message>
```

Vérification de la validité du fichier avec un fichier xsd ('xml/guest.xsd') pour éviter les problèmes d'affichage de date par exemple.

Pour le lire, utilisation de Sax, bien adapté à ce type de fichier car toutes les données sont lues, dans l'ordre du fichier. Même si ce fichier est peu volumineux, j'ai choisi de le traiter avec Sax.

- **Le plan d'accès et la photo de l'hôtel** ont eux aussi des noms en fonction de l'Id de l'hôtel, et sont contenus dans le dossier 'photos' pour la photo, et 'plans' pour les plans. Il m'a paru inutile de stocker en base des adresses de fichiers, surtout qu'il faut être sûr qu'ils aient un nom unique, ce qui est forcément le cas ici

- **Edition du récapitulatif en pdf:** Utilisation de xslt fo, bien adapté à la génération de fichier pdf. Les données viennent d'un fichier xml généré avec la session xml. Le récapitulatif est généré après la validation de la réservation, lors de l'ouverture de la page du récapitulatif. Son nom sera composé de l'id de la réservation Il sera stocké (dans le dossier 'pdf') jusqu'à la date d'arrivée à l'hotel, effacé en cas d'annulation de la réservation, régénéré en cas de modification de la réservation.

- **mysqli:** Utilisation bien adaptée de mysqli dans la recherche de disponibilité de l'hôtel, car la recherche est faite pour chaque jour demandé. La préparation de la requête se fait en dehors de la boucle, et la requête s'exécute autant de fois qu'il y a de jours. Sauf si un des résultats est false, auquel cas on sort de la boucle.

- **Session en simpleXML:** J'ai choisi de stocker dans une session à part les données de l'hôtel, pour éviter l'appel à la base sur les pages suivantes, et parce qu'il servira à la génération du pdf si la réservation se fait jusqu'au bout, puis de stocker les données de la réservation (lorsqu'elle est arrivée avec succès jusqu'au bout) dans le même simple xml, qui sera ensuite enregistré sur le serveur pour pouvoir créer le xml. Je n'ai pas réussi à créer le pdf d'après la seule session (ni avec le simpleXml, ni en l'important en objet dom), ce que je voulais faire au départ.

Voici la structure de cette session xml:

```

<session>
  <hotel>
    <tarif>
      <prix type="single">30</prix>
      <prix type="double">40</prix>
      <prix type="appart">50</prix>
      <prix type="Pdej">4</prix>
    </tarif>
    <idHotel>1</idHotel>
    <nomGerant>nom du gerant</nomGerant>
    <adresse>adresse hotel</adresse>
    <cp>69000</cp>
    <ville>laVille</ville>
    <telHotel>00-00-00-00-00</telHotel>
    <mailHotel>hsdjk@snjf.fr</mailHotel>
    <presta>00101</presta>
    <description>description</description>
  </hotel>
  <resa>
    <nom>leNom</nom>
    <prenom>leprénom</prenom>
    <email>jdhf@hfdjdf</email>
    <dateReservation>2007-06-30</dateReservation>
    <dateArrivee>2007-06-23</dateArrivee>
    <sejour nbreNuits="1">
      <nombre type="single">3</nombre>
      <nombre type="double">2</nombre>
      <nombre type="appart">1</nombre>
      <nombre type="Pdej">5</nombre>
      <total>480</total>
    </sejour>
    <IdRes>12</IdRes>
    <nTel>00-00-00-20-00</nTel>
  </resa>
</session>

```

- **Utilisation du Dom Javascript** pour la vérification des formulaires. Evite de recharger la page en cas d'erreur. Les vérifications de date sont gérées par php, il me semblait beaucoup plus fastidieux de le gérer par javascript, le javascript ne possédant pas de fonction checkdate.

Difficultés rencontrés: Internet Explorer et Firefox ne gère pas de la même façon les nœuds, il m'a été difficile de rendre mon code compatible pour les 2 navigateurs. L'affichage des nœuds d'erreur sur le formulaire de reservation final au niveau des coordonnées bancaires (page reserver.php) avec IE, n'est pas celui attendu.

- **Utilisation du Dom Javascript** pour les bulles, permettant l'affichage des news rss sur la page hotel.php, ainsi que l'affichage du plan d'accès lors du survol du lien adresse (toujours sur cette page)

La base de données:

Base de données mysql avec des tables de type innnoDB, afin de pouvoir utiliser les Foreign Key.

- **Table hotels:**

```
`Id` int(3) NOT NULL auto_increment,
```

```

`libVille` varchar(50) NOT NULL default "",
`libDesc` text NOT NULL,
`libNbSingle` int(3) NOT NULL default '0',
`libNbDouble` int(3) NOT NULL default '0',
`libNbAppart` int(3) NOT NULL default '0',
`libNom` varchar(50) NOT NULL default "",
`libAdresse` varchar(200) NOT NULL default "",
`libCP` int(5) NOT NULL default '0',
`libTel` varchar(16) NOT NULL default "",
`libMail` varchar(50) NOT NULL default "",
`libPrestations` varchar(5) NOT NULL default "",
`libPrixSingle` float NOT NULL default '0',
`libPrixDouble` float NOT NULL default '0',
`libPrixAppart` float NOT NULL default '0',
`libPrixPdej` float NOT NULL default '0',
PRIMARY KEY (`Id`)

```

libPrestations est un champ à 5 caractères 0 ou 1 correspondant à la présence ou non de certaines prestations: baignoire,restaurant,piscine,parking,wifi.On ne fera la recherche sur une prestation que lorsqu'elle est cochée. Autrement dit les résultats peuvent contenir des hôtels qui incluent une prestation non demandée par l'internaute.

- La table planning:

J'ai fait le choix d'utiliser l'id de l'hotel et le type de chambre pour écrire les attributs de la table. La table est de la forme:

```
dateJour(PK) date NOT NULL default '0000-00-00',
```

et pour chaque hotel:

```

`1single` int(2) NOT NULL default '0',
`1double` int(2) NOT NULL default '0',
`1appart` int(2) NOT NULL default '0',.....

```

Cette table contient par défaut le nombre de chambres disponibles par jour suivant le type de chambre, et est décrémentée à chaque réservation.

- La table clients:

```

`IdClient` int(11) NOT NULL auto_increment,
`libNom` varchar(50) NOT NULL default "",
`libPrenom` varchar(50) NOT NULL default "",
`libMail` varchar(50) NOT NULL default "",
`libTel` int(10) NOT NULL default '0',
`mdp` varchar (15) NULL default '0',
PRIMARY KEY (`IdClient`)

```

- La table reservations:

```

`IdResa` int(11) NOT NULL auto_increment,
`IdClient` int(11) NOT NULL,
`libJourR` date NOT NULL default '0000-00-00',
`libNbSingle` int(1) default NULL,
`libNbDouble` int(1) default NULL,
`libNbAppart` int(1) default NULL,
`libNbPdej` int(2) NOT NULL default '0',
`libJourA` date NOT NULL default '0000-00-00',
`libNbNuits` int(1) default NULL,
`libPrix` int(11) NOT NULL default '0',

```

```
`IdHotel` int(11) NOT NULL default '0',
`libCB` int(11) NOT NULL default '0',
`libCrypto` int(3) NOT NULL default '0',
`libExpir` int(6) NOT NULL default '0',
PRIMARY KEY (`IdResa`),
FOREIGN KEY IdHotels REFERENCES hotels(Id),
FOREIGN KEY IdClient REFERENCES clients(IdClient)
```

Ces Foreign Key assurent l'intégrité des données: entre l'hotel et ses réservations, entre le client et sa ou ses réservations.

Les données de la carte bleue sont stockées pour permettre de le paiement de la 1ere nuit si la personne ne se présente pas à l'hôtel et n'a pas annulé sa réservation.

Description de l'application:

- Affichage des formulaires:

- **Classe Formulaire ('classes/crea_form.php')** qui gère l'écriture des formulaires de recherche et de réservation.

- `ecrireSelectMois()`: Utilisation d'un tableau php pour l'affichage des noms des mois, associée à leur valeur numérique. Mois actuel affiché par défaut

- `ecrireSelectJour()`: Jour actuel affiché par défaut

- `ecrireCaseCochée()`: Écriture des cases à cocher correspondant aux prestations demandées

- `ecrireListeNum()`: Pour le choix du nombre de nuits, du nombre de chambres single, chambres double, appartements et petits déjeuners.

- `expirCB()`: Écriture des mois et années de l'expirationCB dans le formulaire de réservation

- `ecrireListeString`: Écriture du formulaire département et du formulaire ville.

- Pour les départements, utilisation d'un fichier 'xml/regions_dpt.xml' pour le tri des départements par ordre alphabétique (par numéro de département). Utilisation du domphp et import en simpleXML pour l'affichage

- Pour les villes affichage simplement de la valeur par défaut du formulaire (Sélectionnez), l'affichage des villes se fait par javascript (voir ci-dessous)

- **L'affichage des villes** se fait à partir du fichier 'regions.xml' décrit plus haut:

Fonctionnement:

- le script javascript: 'script_js/form_ville_dep.js'

- 2 événements déclenchent la fonction `affiche_ville()`:

- Le chargement de la page,

- La sélection d'un département (ou retour à la valeur de départ 'Sélectionnez') de

l'internaute

- On efface les villes du formulaire si il y en a.

- Si aucun champ n'est sélectionné:

- on récupère les données du fichier regions.xml.

- Appel par Ajax du fichier 'xml/result_villes_xml.php'. Ce fichier affiche toutes les villes de 'xml/regions.xml' dans l'ordre alphabétique via une transformation xslt: 'xml/regions.xsl'. Le fichier php permet de faire la transformation.
- Affichage des ville dans le formulaire 'ville'.
- Si un département est sélectionné:
 - On va directement dans le fichier 'xml/regions.xml' récupérer les villes correspondant au numero du département.

J'ai eu beaucoup de difficultés à créer ces deux fonctions avec Ajax, car plusieurs fichiers sont en cascade, et l'utilisation du xslt, du DomPhp, du xpath, du dom javascript, le tout avec Ajax est assez complexe à mettre en œuvre.

- Contrôle des formulaires:

- Contrôle de la présence des champs:

Par javascript via le Dom: fichier ('script_js/form_rech_verif.js')

- fonction verif() effectuée lors de la soumission du formulaire de recherche:
 - On verifie la partie demande de disponibilité avec la fonction verif_dispo().
 - Aucun des champ, ou au moins un type de chambre + un nombre de nuit est requis pour une recherche
 - Au moins un type de chambre + un nombre de nuit est requis pour une demande de réservation .
 - On vérifie que le champ département ou le champ ville est rempli.

- Classe ctrlFormulaire ('classes/ctrl_formulaire.php'):

- Contrôle par php de la validité de la date verifDispo():
 - Qui ne doit pas être inférieure à la date du jour,
 - Qui doit être valide (pas de 31 Juin par ex)
- Si ok, on trouve l'année de la demande (non demandée sur le formulaire). Par défaut c'est l'année en cours, sauf si on est en fin d'année.
- Mise en session des données de recherche de disponibilités (si elles existent)
- Si pas de message erreur Date, creaSession() :
 - Appel de la fonction presta() pour transformer les cases de demande de prestations non remplies en 0.
 - Concatenation des valeur: champ à 5 caractères de 0 et de 1.
 - Mise en session des champs departement ville et prestations.
 - On passe à la page result_rech:
- Si message d'erreur, on reste sur la même page.

Recherche et choix hôtel:

- Classe recherche ('classes/recherche.php'):

- Si recherche par région (clic sur la carte de France en page d'accueil), on ouvre le fichier 'xml/regions.xml' avec le domphp, et avec l'identifiant de la région passé en GET, on récupère avec une fonction xpath les numéro des départements correspondant à la région. Création de la requête où on comparera les numéro des départements avec les deux premiers chiffres des codes postaux des hotels
- Si recherche provenant du formulaire, création de la requete en fonction des données remplies précédemment par l'internaute:
 - rechercherVille(): si la ville est renseignée (et pas la demande de dispo).
 - rechercherDep(): si le département est renseigné (et pas la ville ni la demande de dispo).
 - rechercherVilleDispo(): si la ville et la demande de dispo sont renseignées.
 - rechercherDepDispo():si le département et la demande de dispo sont renseignés. (et pas la ville). Dans ces deux derniers cas on effectue d'abord une recherche par ville ou département, puis pour chaque hotel trouvé, on effectue une recherche de disponibilité sur la table planning pour chaque jour demandé avec mysql. Renvoi de true ou false et on insère dans un tableau php les ids de chaque hôtel

disponibles, Ensuite on crée une nouvelle requête où on cherche les hotels par Id pour afficher les résultats .

- Si on vient de la liste d'hôtels affichée par une requête précédente, création d'une requête avec l'id de l'hotel passée en GET

- On effectue la requête finale avec faireRequete(), on récupère le nombre de résultats, et l'exécution de la requête.

- Si on est dans la phase de réservation on effectue une requête, rechercheDispo(), avec l'id de l'hôtel. Cette requête renvoie true ou false et conditionne la suite de la réservation.

- Classe resultatRecherche ('classes/resultatRecherche.php'):

- En fonction du nombre de résultat de la requête, fonction resultatRequete():

- Pas d'hôtels, message d'erreur et on retourne à l'accueil

- Plusieurs hôtels, on reste sur resultat_rech.php. Appel de la fonction plusieursReponses qui affiche la liste des résultats. Affichage des images des prestations avec la fonction ecrirePrestas().

- Un seul hôtel: Appel de la fonction uneReponse() qui crée un simpleXML avec les données de l'hôtel, le mets en session, puis envoie sur la page hotel.php pour voir le détail de l'hôtel:

- RecupereLeResultat(), permet la récupération du xml en session pour l'affichage des détails de l'hôtel.

- recupRss(), permet de récupérer avec simpleXML et xpath (avec le numéro du département) le bon flux rss inscrit dans le fichier 'xml/regions.xml' , que l'on affiche ensuite avec afficherRss(). Ici on utilise le domphp pour accéder à l'arbre xml du fichier. On affiche une liste de lien avec la date de publication (à mettre en français). On inclut une fonctionnalité pour afficher dans une bulle le début de la news lors du survol du lien. Cette bulle est générée par javascript: 'script_php/bulle.js' avec la fonction popup(). Cette fonction sert également à l'affichage du plan d'accès lors du survol du lien adresse. cachePop() permet la disparition de la bulle lors du rollout.

- AfficherMeteo(), permet l'affichage de la météo de la ville correspondant à l'hôtel, via le web Service globalweather, si les données sont présentes. Appel de l'image correspondant au temps en fonction de la chaîne de caractère renvoyée pour le paramètre SkyConditions.

- AfficherGuestbook(), permet l'affichage du livre d'or correspondant à l'hôtel, après validation avec le xsd L'affichage se fait par le parser Sax en php.

- Affichage des images des prestations avec la fonction ecrirePrestas()

Réservation:

- Classe reservation ('classes/reservation.php'):

- On a rempli et envoyé le formulaire de réservation de la page 'hotel.php'. Le contrôle du formulaire s'est bien passé, la disponibilité est ok.

- fonction constructeur qui récupère la session xml avec les données de l'hôtel, les valeurs POST du formulaire final de résa si elles existent, les valeurs de la session contenant la demande de disponibilité (nbre de chambres, nbre de nuits et date d'arrivée)

- Affichage de la demande avec afficherDemande().

- Soumission du formulaire où le client renseigne ses coordonnées et ses coordonnées bancaires:

- Vérification du formulaire avec le dom Javascript, fichier 'script_js/form_reserver_verif.js'.

Vérification de la présence des champs obligatoires et de la validité du mail et du numéro de téléphone (10 chiffres).

- Simulation de l'accord de la banque avec la case à cocher.

- Si ok, on revérifie une dernière fois la disponibilité de l'hôtel, puis on insère dans la base de données avec la fonction AjouterBdd(). Insertion dans la table client, reservation, et décrémentation du nombre de chambres disponibles dans la table planning.

- Insertion dans la session xml d'un nouveau nœud resa et des données de la réservation. Puis, on va sur la page 'recapitulatif.php'.

- Recup de la session xml avec la fonction constructeur de l'objet reservation.

- Envoi du mail de confirmation avec la fonction EnvoyerMail(). Utilisation de 3 fonctions qui serviront également pour l'écriture de la page:
 - recap_client(), recap_resa(), recap_hotel()
- Génération du pdf avec ecrirePdf(). On enregistre dans un fichier xml (du nom de l'id de la réservation dans le dossier 'session_xml') la session xml, utilisation du fichier contenant la feuille de style: 'xml/recap_fo.xsl'
- Affichage de la page avec les fonctions recap_client(), recap_resa(), recap_hotel() ecrirePrestas().

Gestion de la Session:

J'ai utilisé en plus de la session xml décrite plus haut, une session 'normale', pour garder en mémoire la recherche de l'internaute. J'ai choisi de distinguer la recherche simple de la recherche avec les disponibilités, en faisant 2 sessions distinctes, une pour la ville et le département, ou la région (recherche) et une pour la recherche de disponibilité, ce qui permet de faire le choix du type de recherche en testant la présence ou nom de cette session rech_dispo.