Projet de gestion des vols d'une compagnie aérienne

À intégrer après, si on a le temps :

promos

escales

interface côté compagnie

envoi d’un billet par mail

numéro de tél => autoriser les espaces et les enlever lors de la récup (penser à protéger les requêtes)

hôtels

prix qui évolue selon la date

(espace membres)

Sera géré en Java :

Création d’un vol ?

historique des vols pour un client + gestion carte de fidélité

1 seule compagnie : DEV-FLY

vols internationaux

vols directs

126 avions voyagent tous les jours (destinations desservies une fois par semaine)

allers simples + allers-retours (moins chers)

Pour créer un vol : gestion des droits (admin avec login / mot de passe) sur la base de données directement... (pr le moment)

Nombre de places constant dans les avions : 170

Pas de gestion des plateaux repas

-----------------------

Un client peut réserver pour plusieurs passagers (dont il fait partie) [voir si client hérite de passager ? => non. Un client qui réserve n’est pas forcément passager (cf réservation pour 1 enfant)]

Tarifs (pas une classe) dépendent de : vol, classe (éco / affaire), age (enfants de moins de 3 ans ne payent que l'assurance), date de réservation (réservation ouverte 2 mois avant ! Le prix s'incrémente progressivement, et décrémentera à la fin). [on retrouve le tarif ds la table reservation]

Lors d'une réservation : on propose les dates à partir de la date du jour et jusqu'à 2 mois après

LES TABLES :

TABLE vol -> numvol, lieu\_depart, lieu\_arrivee [sera récupéré sur la table destination], dateheuredep, dateheurearrivee

TABLE reservation -> numreserv, date, #numclient

TABLE passager -> numpassager, civilite, nom, prenom, date\_naissance

TABLE client -> numclient, civilite, nom, prenom, adresse, cp, ville, pays, mail, tel\_fixe, tel\_portable, login, password

TABLE destination -> codeaeroport, pays, ville [à récupérer pour les entrées dans le formulaire]

TABLE place -> numplace, prix, #numpassager, #numvol, #numreservation

# pour compter le nombre de places restantes : COUNT le nb de places pour tel numvol - le nb de place achetées

Rq : on enlève « etat (booléen : acheté ou non) », cf un paiement = une réservation

TABLE employe -> numemploye, civilite, nom, prenom, adresse, cp, ville, pays, fonction

TABLE travailler -> #numvol , #numpilote, #numcopilote, #numhotesse1, #numhotesse2, #numhotesse3, date

TABLE user -> id, statut, login, password, droits [lire, écrire]

LA STRUCTURE DU PROJET :

entity : Vol, Passager, Client, Reservation, Billet

dao : MysqlDao

view : ATTENTION 2 vues à chaque fois (FRANÇAIS et ANGLAIS). index.php, rechercheVol.php, propositionVol.php, reservation.php, paiement.php, confirmation.php, aurevoir.php

controller : 1 contrôleur lié à chaque vue (displayControllerRecherche.php, displayControllerProposition.php, etc)

conf : configuration.ini

setup.php

DETAIL DES PAGES :

\* confirmation.php

"Voulez-vous faire une autre réservation ?"

Oui => retour rechercheVol.php (avec certains champs préremplis ?)

Non => quitter, page aurevoir.php

\* aurevoir.php

"Bon voyage"

\* index.php

Carousel

Choix de la langue anglais et français

3 (grandes) fonctionnalités :

PHP

* Recherche d’un vol + proposition
* Espace client : consulter, annuler, modifier, régler en ligne
* Back office : gestion de passagers (visualiser pour un vol les différents passagers + leurs places) + gestion du personnel (pilotes, copilotes, hôtesses, stewards -> qui travaille sur quel vol)

DANS NOTRE RAPPORT

* "si on avait eu plus de temps, on aurait inséré la possibilité de modifier le nb de passagers sur un vol"
* On aurait pu prévoir le complément d’adresse + « Ville Cedex 2 »

AMELIORATIONS POSSIBLES

* Rendre le site responsive
* Choix des destinations depuis un select
* ajouter du JavaScript
* regrouper des méthodes au sein d'un même contrôleur plus général
* ne pas avoir de vues dans le .htaccess : passer exclusivement par des contrôleurs qui incluraient des vues (au lieu de rediriger dessus)
* enlever les return 1, 2, 3... (cf contexte objet)
* Sur l'appli :
  + système de fidélité
  + système de récupération de mot de passe
  + gestion des allers-retours

JAVA

* 1) Création + modification d’un vol
* 2) Modification des données dans l’interface back office (modifier l'adresse du client, etc)
* 3) Création d’un employé

CDG différents → Services distincts : service « vol » / service « client » / RH ?

**Idées** en vrac (penser à être cohérents si une modif doit se répercuter d'une table à une autre) !!

1) Partie liée à la « réservation d'un vol » côté PHP – gestion des vols

Création + modification d'un vol en le reliant à un code aéroport (reprendre la table Destinations) → modifiable que s'il n'y a pas de résa dessus (sauf pour le prix).

Pour qu'il soit effectif : lui affecter des employés (qui ne travaillent pas à cette date)

Ajout + modification d'une destination.

(Amélioration possible : chger le nombre de places réservables sur un vol).

2) Partie liée à « l'espace client » côté PHP – gestion des clients

Modification des coordonnées d'un client

Modification des coordonnées d'un passager (si on change la date de naissance → pourrait changer le prix du billet déjà réglé : problématique...?)

Modifier une réservation : supprimer un passager → le montant serait remboursé au client directement (on peut imaginer un remboursement de 50 % seulement selon la date ou autre)

3) Partie liée au « back office » côté PHP

Création d'un employé en base

Modification des coordonnées d'un employé existant

Suppression de l'employé : il faut impérativement vérifier qu'il n'est pas affecté sur un vol qui va partir

+ gestion de la date à laquelle l'employé démissionnera

Changement de poste (promotion d'un copilote en pilote...) : implique de vérifier qu'il n'est pas affecté à un vol futur + changer son ID → voir comment le gérer

Autre idée :

Consulter les vols de dernière minute (à définir) avec des places restantes : pour un affichage sur un écran dans l'aéroport pour donner des idées de voyage à ceux qui partent au hasard à la dernière minute