The CERICar WEB Application

Objectifs : Dans le cadre de l'UCE AMS – Concevoir et développer une application WEB, vous allez développer une application web dont le contexte applicatif sera le "covoiturage". Cette activité de mise en situation (AMS) a pour objectif pédagogique de vous amener à revoir et maîtriser les différentes technologies vues tout au long de votre cursus de licence et des cours « UCE Architectures Web » et « UCE Modélisation Objet et UML » de cette année : html, CSS, javascript, ajax, Bases de données et Web, ORM, modélisation objet, PHP objet, UML et patron de conception MVC…

*Le framework Yii sera (**obligatoirement**) votre environnement de travail pour le développement de cette application.*

Contexte Applicatif – The CERICar

Vous allez développer une application WEB de covoiturage (vous ne connaissez pas *BlaBlaCar ?* Visitez le site pour vous donner une petite idée des attendus - https://www.blablacar.fr ?).

Une personne voyageant pourra consulter l'application WEB pour rechercher un voyage qui lui correspond. Le « moteur de recherche » de votre application lui permettra de préciser une ville de départ, une ville d'arrivée et un nombre de personnes voyageant ensemble. Lors de la recherche, cette personne peut indiquer qu'il ou elle accepte un voyage avec des correspondances. Dans ce cas, le « moteur de recherche » se chargera de composer plusieurs voyages proposés par des conducteurs ou conductrices différentes en respectant les contraintes temporelles (l'heure d'arrivée du premier voyage est inférieure à l'heure du départ du second). Le résultat de la recherche sera donc, si disponible l'ensemble des trajets avec et sans correspondance avec les informations s'y référant - information sur le voyage lui-même, sur le ou la conductrice et le coût global du voyage que la personne devra s'acquitter en cas de réservation.

La personne réalisant la recherche d'un voyage ne fournira pas d'heure de départ et d'arrivée désirées. C'est l'application qui, en fonction des propositions de voyages suite à l'enchaînement des trajets (avec respect de la contrainte : fin du trajet N inférieure au départ du trajet N+1), fixera ces deux variables.
Dans l'application que vous allez développer (pour des raisons de simplification), les voyages proposés sont quotidiens (et permanents) et les recherches/réservations se font toujours pour le lendemain. Pour faciliter le développement, nous considérons que la durée moyenne d'un trajet (en minutes) est équivalente à sa distance (en kilomètres). Ainsi, un trajet de 60 kilomètres est donc parcouru en 60mn, soit une heure.

Si un voyage correspond à la demande (avec suffisamment de places disponibles), la personne pourra réserver sa ou ses places sur le ou l'ensemble des trajets composant le voyage demandé. Pour cela, il ou elle devra créer un profil ou se connecter à l'application, si un compte a déjà été créé.
La création d'un compte nécessite de la part d'un internaute de fournir les informations suivantes : son nom et prénom (ex : « *Dupont Alexis* »), un pseudo unique (ex. « *AlexD-84* ») et un mot de passe permettant

l'authentification, son adresse mail (ex : « *alex84.dupont@gmail.com* »), son numéro de permis si il ou elle en possède un (code de 1 à 12 chiffres. Ex : «*12345678910* »), une photo de profil (url simple vers une photo. Ex. « *https://unsplash.com/fr/photos/volkswagen-coccinelle-bleue-sur-un-terrain-en-herbe-7HNftpNvqho* »).

Cette application permettra également aux conducteurs et conductrices qui le souhaitent de proposer des voyages sur des trajets définis en indiquant un certain nombre d'informations : le type de véhicule (liste fermée parmi les choix suivants : « Citadine, Berline compacte, Break, Routière, Monospace, SUV »), sa marque (Ex. « Peugeot »), le tarif du trajet (en **euros par kilomètre et par voyageur**. Ex : « *3.5* »), l'heure de départ du véhicule, le nombre de places disponibles en terme de personnes voyageant (Ex : « *4* »), le nombre de bagages autorisés par perosnne (Ex : « *1* »), d'autres contraintes (champs texte libre. Ex. « *animaux autorisés. Voiture sans cigarette ni vaporette* »). Les conducteurs et conductrices doivent se connecter à l'application pour pouvoir proposer des voyages.
Les trajets possibles sont pré-définis par l'application que ce soit pour la recherche ou pour la proposition d'un voyage. Un trajet correspond à une ville de départ, une ville d'arrivée et à une distance en kilomètres (ex : « *Toulouse* », « *Marseille* », « *412* »).

Un conducteur ou conductrice est une personne ayant indiqué un numéro de permis lors de la création du compte. Il ou elle ne peut pas rajouter un trajet non prévu initialement dans l'application mais il peut proposer des voyages sur les trajets existant.
Attention : les correspondances se font uniquement au niveau de la recherche du voyage et non pas au niveau de la proposition d'un voyage par un conducteur ou conductrice. Cette dernière peut proposer un voyage uniquement sur un trajet direct. S'il ou elle désire proposer un voyage avec correspondance, il ou elle devra proposer autant de voyages que de trajets composant le voyage global.

Pour une personne connectée, il sera possible de visualiser l'ensemble de ses réservations de voyages. Si la personne est un conducteur ou une conductrice, il sera possible de visualiser l'ensemble des voyages qu'il ou elle propose et les réservations associées.

Évidemment, l'application doit être accessible depuis n'importe quel support, aussi bien sur un smartphone, une tablette ou un écran d'ordinateur. Pensez donc bien à traiter la mobilité.

Réalisation fonctionnelle et technique

L'ensemble du développement de l'application CERICar reposera sur le framework Yii basé sur le patron de conception MVC, sur son ORM ActiveRecord et sur l'ensemble des classes à disposition pour développer rapidement une application.
Vous devrez commenter votre code au fur et à mesure de son écriture.

1. Etape 1 : Appropriation du sujet – Modélisation UML

Sur la base du contexte applicatif décrit ci-dessus et de l'application que vous allez devoir développer, vous devez élaborer l'ensemble des diagrammes nécessaires à la phase de conception : diagramme de cas d'utilisation, diagramme de classes, diagramme d'états-transitions, diagramme de séquences, …
Ces diagrammes vous serviront évidemment dans les étapes de développement qui vont suivre.

2. Etape 2 : Yii - MVC, Modèle de données et ActiveRecord

Il s'agit dans cette étape d'implémenter le modèle de données relatif à l'application visée en fonction de l'analyse réalisée dans l'étape 1.

Les données seront physiquement stockées dans des tables disponibles dans une base de données publique PostgreSQL (disponible sur le schéma « *fredouil* »). Vous utiliserez, par conséquent, TOUS et TOUTES, les mêmes tables, en consultation mais également en écriture quand nécessaire (insert, update, delete)[1].

Pour l'accès à ces données stockées en base de données, vous devez **associer votre modèle de données aux différentes tables en utilisant l'ORM de Yii – ActiveRecord et la classe associée portant le même nom**.

A cette étape, vous devez implémenter :
a. l'ensemble des classes nécessaires à votre application, et, notamment, celles en lien avec les tables de la base de données (*internaute*, *trajet*, *voyage*, *réservation, marquevehicule, typevehicule*). Pour ce faire, ces classes devront hériter de la classe mère ActiveRecord et implémenter quand vous le jugerez nécessaires les relations entre entités.
b. la ou les actions (méthodes) dans le contrôleur ainsi qu'une vue basique vous permettant de tester votre modèle et d'afficher (sans mise en page, uniquement du texte brut) :

- les informations d'un ou une internaute (enregistrement de la table *internaute*) ayant un *pseudo* donné (Ex. « *Fourmi »)*
- si cette personne a fourni un numéro de permis et a des propositions de voyages, les informations de ces derniers
- si cette personne a des réservations de voyages, les informations de ces dernières.

Lors de l'affichage de ces informations, tout *id* correspondant à une relation (clé étrangère dans une table) devra être explicité par les données de l'enregistrement correspondant. Ex : l'affichage d'un *voyage* devra montrer les informations d'un *trajet* (ville de départ, ville d'arrivée, distance) et non l'*id* du *trajet.*

Comme exemples de fonctionnalités (et méthodes de classe) dont vous pourriez avoir besoin au cours du développement de votre application et que **vous devez implémenter dans votre modèle de données** :
- *getTrajet* permettant de récupérer un objet de type *trajet* à partir de deux chaines de caractères : ville de départ, ville d'arrivée.
- *getVoyagesByTrajetId* permettant de collecter l'ensemble des objets de type *voyage* correspondant à un *id* de

1 Toutes les commandes sur la base de données sont loguées dans un journal de logs. Aussi, ne faites pas svp n'importe quoi, encore moins, de manière intentionnelle !

trajet.
- *getReservationsByVoyageId* permettant de collecter l'ensemble des objets de type *reservation* correspondant à un *id* de voyage.
- *getUserByIdentifiant* destiné à récupérer les informations d'un ou une internaute selon son *pseudo* (et pas son *serial id*).

3. Etape 3 : Yii - MVC, vue et layout

A ce stade, nous vous demandons de **développer la vue et le contrôleur** permettant à un ou une internaute de **rechercher un voyage** parmi les voyages proposés dans l'application (et dans la base de données). Nous chercherons uniquement les voyages directs sans contrainte particulière et sans correspondance (Ex : un voyage « *Toulouse - Marseille* » pour *3* voyageurs).

Votre application permettra à l'internaute de choisir sa ville de départ, sa ville d'arrivée et d'indiquer le nombre de personnes voyageant. Une fois, ces informations saisies, le serveur devra retourner tous les voyages correspondant au trajet (ville de départ, ville d'arrivée) en précisant le nombre de places disponibles ainsi que les voyages affichant complet, et le coût total du voyage. *Nous vous rappelons que l'internaute recherche un voyage de la veille pour le lendemain sans se soucier des horaires. Par ailleurs, un voyage est dit « complet » si la table reservation comprend autant de places réservées que de places disponibles dans le voyage considéré.*

L'objectif de cette étape est de réfléchir au design de votre application et à sa charte graphique. Vous pouvez récupérer un template existant (du moment que vous citez vos sources et conservez les informations de son propriétaire le cas échéant).
Ce design sera répliqué dans les étapes suivantes quand vous en viendrez à développer les autres fonctionnalités attendues de votre application.

Nous vous rappelons qu'il s'agit ici de faire fonctionner le modèle MVC avec une vue et le layout associé renvoyés par une action présente dans le contrôleur pour répondre à cette fonctionnalité (« *Rechercher un voyage* »). Cette vue permettra d'afficher les *input* nécessaires à la saisie des données par l'internaute et, une fois ces informations saisies et renvoyées au serveur, d'afficher les voyages correspondant (une seule action, une seule vue).

Il est rappelé que :

- l'utilisation du PHP au niveau de la couche Vue doit **se limiter au strict minimum** touchant à la récupération d'informations à afficher.
- Yii vous fournit la **classe assistante Html** (cf. cours) pour simplifier le code côté vue et accélérer le développement.
- Ici toute la page est rechargée côté client une fois l'action du contrôleur exécutée et la vue+layout retournés.

4. Etape 4 : Yii - MVC, intégration d'Ajax

L'appel à la fonctionnalité développée dans l'étape 3 conduit à une action exécutée dans le contrôleur (méthode associée), à une interaction avec le modèle de données et au renvoi du layout et d'une vue (associée à l'action) par le contrôleur.

Il s'agit maintenant de ne recharger, côté client, que la vue nécessaire à l'action « *Rechercher un voyage* » et de ne plus recharger le layout complet. Ce changement passera par l'appel d'**une requête Ajax** comme suit :

- Le code javascript exécuté côté client pour l'appel à la requête Ajax se fera par le biais de la librairie jQuery et l'appel à la méthode $.ajax(). *Note : Yii renvoie la librairie JQuery par défaut, vous n'avez*

pas à vous en charger.

- Les actions de votre contrôleur associées à une requête Ajax renverront soit des données brutes (au format JSON) que votre code JavaScript côté client traitera et intégrera au DOM de votre application, soit directement une vue que votre code JavaScript injectera au DOM.

Dans cette étape, vous devrez également implémenter la gestion d'un bandeau de notification présent dans le *layout*. Toute action côté serveur devra faire l'objet d'un retour au client, sous la forme d'un message de notification dédié tel que « *Recherche terminée* », « *Voyage trouvé* », « *Internaute inconnu* », ... Ce message devra s'afficher dans un bandeau de notification commun à toutes les actions et disponible dans le *layout (*et pas dans la vue*)*. Le code JavaScript gérant le retour de la requête Ajax devra gérer la mise à jour de ce bandeau de notification.

Toutes les fonctionnalités liées à votre application et interagissant avec le serveur devront, **dans les étapes suivantes**, être associées, sauf exception, à du code JavaScript et à une (ou plusieurs) requête Ajax.

5. Étape 5 : Réservation et proposition d'un voyage

Maintenant que l'internaute est capable de chercher un voyage, l'application doit lui permettre de réserver une ou plusieurs places sur un voyage sélectionné. L'internaute doit être connecté sur l'application pour effectuer sa réservation. Plusieurs fonctionnalités sont donc à développer pour cette étape :

- un module d'inscription, permettant à un ou une nouvelle internaute de s'inscrire sur le site. Il ou elle doit pour cela fournir toutes les informations nécessaires pour l'enregistrer dans la base de données.
- un module de connexion, permettant aux internautes inscrits de s'identifier
- un module de réservation, permettant à un ou une internaute inscrite de réserver un voyage, après recherche. Le voyage devra ensuite apparaître dans son profil.

Dans la base de données, un voyage proposé par un ou une conductrice comprend le nombre de places **maximum** disponibles dans le véhicule mais pas le nombre de places réservées. **Ce nombre de places maximum ne doit pas être modifié en fonction des réservations enregistrées.** Pour connaître le nombre de places disponibles, il faudra interroger la table des réservations pour le voyage ciblé.

Pour finir, votre application devra permettre à un internaute (conducteur/conductrice) connecté de proposer un voyage. Pour cela il ou elle devra spécifier le trajet, l'heure de départ, le nombre de places maximum disponibles, le tarif en euros par voyageur et par kilomètre et le nombre de bagages autorisés par voyageur. Il devra aussi préciser dans un champ de commentaires libres s'il ou elle a des contraintes particulières (*animaux acceptés, non-fumeur,* …). Ces contraintes ne seront pas prises en compte dans le module de recherche mais apparaîtront comme informations complémentaires concernant un voyage.

Un bonus sera donné à celles et ceux qui auront pris en compte les voyages avec correspondance possible dans la recherche de voyages et la réservation.

Timeline et Notation

Vous avez 9 séances pour réaliser ce projet. Il s'agit d'un travail individuel.

Il est attendu que vous fournissiez également du travail personnel pour atteindre les objectifs et passer les différentes étapes.

Vous pouvez avancer à votre rythme, mais nous vous recommandons de suivre le planning suivant :
- Séance 1,2 : étape 1
- Séance 2,3 : étape 2
- Séance 4,5 : étape 3
- Séance 5,6 : étape 4
- Séance 7,8,9 : étape 5

La note totale de l'application est composée de cinq notes :
- fin de l'étape 1 => si validée après dépôt 3 points
- fin de l'étape 2 => si validée en séance 4 points
- fin de l'étape 4 => si validée en séance 4 points
- fin de l'étape 5 => si validée 4 points, 2 points supplémentaires pour la fonctionnalité bonus
- mobilité de l'application, commentaires et propreté du code : 2 points

Dès que vous avez fini une étape correspondant à une notation, vous faites appel à l'enseignant présent pour vous évaluer (pour l'étape 1, uniquement le dépôt). Cette évaluation comprendra la présentation de votre application et des fonctionnalités en lien avec l'étape concernée, ainsi que la réponse à des questions sur le code écrit, voire la modification en direct de ce dernier pour atteindre les objectifs de l'étape. Les étapes doivent être validées dans l'ordre. Attention, l'étape 2 doit être validée au plus tard à la fin de la séance 5 et l'étape 4 au plus tard à la fin de la séance 8 (sinon les points correspondant seront perdus). Les points qui correspondent à la mobilité et à la qualité du code (commentaires et propreté du code) ne pourront être attribués que si l'étape 5 est terminée ET validée.