Projets informatiques 2019

ENSG - ING1

Présentation générale

Ce projet informatique intervient pour vous permettre de faire une synthèse entre les différents cours suivis cette année en informatique (algorithmie, analyse informatique, base de données et programmation orientée objet). Etant donné qu'il s'agit d'un premier projet, les contraintes techniques sont assez fortes. Il vous sera ainsi demandé :

- de formaliser une analyse du sujet avec les outils UML;
- de mettre en oeuvre les concepts de l'orienté objet;
- de produire une documentation de votre code.

La livraison d'une interface graphique aboutie n'est pas attendue (des graphiques Java2d ou le dessins de formes géométriques simples en svg suffit dans la plupart des cas). Les groupes pour lesquels les fonctionnalités de base de l'application auront été développées pourront s'attaquer à cette problématique en utilisant javax.swing pour réaliser une application bureautique.

Enfin, pour les sujets de simulation, la gestion du “temps réel” n’est pas demandé : une simulation au “tour par tour” suffit.

Pour tous les projets nécessitant d'enregistrer des données, vous pourrez utiliser le SGBD de votre choix. Pour info, une base de données SQLite présente l'avantage d'être portable et vous permettrait donc de travailler de n'importe où…. **Cependant, l’application devra fonctionner sans SGBD dans un premier temps (pas d'interaction avec la BDD dans vos classes, méthodes, objets) ! L’important étant de mettre l’accent sur les concepts orientés objet.**
D’une manière générale, séparez bien ce qui relève du modèle (classes et objets) et ce qui relève de l’interface graphique ou de la persistance des informations (fichiers ou base de données).

Organisation

Ces projets s'étendent sur 13 séances de 3h, étalées sur un peu moins de deux mois, auxquelles s'ajouteront deux jours de soutenance. Certaines séances seront encadrées, tandis que d'autres s'effectueront en autonomie. Votre présence est bien entendu obligatoire durant l'ensemble des créneaux réservés.

Vous travaillerez en trinômes. Un groupe sera composé de deux étudiants, soit **20 groupes** en tout (19 trinômes et un binôme). La composition des groupes est laissée libre mais svp essayez de faire en sorte qu’ils ne soient pas trop déséquilibrés.

10 sujets sont proposés. **Un sujet sera donc choisi par exactement deux groupes**.
Vos délégués nous feront remonter pour le **mercredi 27/03 à 11h au plus tard** la liste des groupes ainsi que les sujets choisis, dans un fichier excel ou libreoffice.

La séance du 09 mai matin sera réservée aux derniers préparatifs en vue de la soutenance (préparation d'une démonstration, tests en conditions réelles...).

La planning complet des séances de projet est le suivant. Quelques points d'étape importants devant guider votre travail y sont indiqués :

- 27/03 pm : démarrage du projet, début de l'analyse
- 01/04 journée : validation de l'analyse l'après-midi
- 03/04 matin : début des développements
- 03/04 pm **autonomie**
- 08/04 journée
- 12/04 matin- **autonomie**
- 12/04 pm
- 19/04 matin **autonomie**
- 19//04 pm
- 07/05 matin
- 09/05 matin : préparation de la soutenance
- 13/05 journée et 14/05 journée : soutenances

Rendus

Tous les rendus devront se faire sur la plateforme Gitlab ou Github. Chaque groupe devra y créer un projet et nous y donner accès.

- Un **rapport d'analyse** sera à rendre pour le **jeudi 11/04 à 23h59** (un point de pénalité par minute de retard). Le rapport d'analyse devra avoir été validé par l'équipe enseignante pour avoir le droit de soutenir. Il s'agira de rappeler les objectifs du projet, de procéder à une reformulation du sujet et d'en faire une modélisation UML avec a minima un diagramme des cas d'utilisation et de classes, puis en fonction du sujet, d'activités, de séquence, d'états-transitions, etc. Il ne s'agira pas de lister de manière exhaustive toutes les fonctionnalités de l'application mais de se concentrer sur les plus importantes / complexes.
  Le rapport devra également présenter un planning prévisionnel du projet (GANT)

- Le **code** source de l'application, la **documentation** utilisateur (installation des outils, de l'application, tutorial, readme, quickstart, vidéo ...) et développeur (javadoc). A rendre pour le **09 mai avant 23h59.**

- un **document de synthèse** à rendre pour le **09 mai avant 23h59** (même pénalité de retard que précédemment). Le document de synthèse (5 pages maximum) doit vous permettre de faire le bilan de votre projet (comment vous vous êtes organisés? ce que vous avez réussi à faire? pourquoi vous n'avez pas fait d'autres choses? ce que vous avez appris?). Il doit également comporter un partie plus technique expliquant comment sont organisés vos développements et comment exécuter votre programme.

Soutenances

Les soutenances auront lieu le **lundi 13 mai toute la journée et le 14 mai toute la journée**. Chaque soutenance durera 20 minutes incluant 5 minutes de démonstration du fonctionnement de votre application. Elle sera suivie de 10 minutes de questions du jury. Tout naturellement, **la présence à l'ensemble des soutenances est obligatoire**.
Vous devrez appuyer votre discours d'une présentation (power-point ou autre support).

Evaluation

La livraison d'une application fonctionnelle n'est pas la seule fin du projet. L'analyse du sujet, la modélisation UML, la qualité du code produit (clarté, ré-utilisabilité...) et la documentation comptent pour une part importante de la note finale.

La grille d'évaluation indicative se présente de la manière suivante :

Critères d'évaluation		Barème indicatif
Rapport d'analyse	Analyse du sujet Modélisation UML	4
Fonctionnalités	Nombre et complexité des fonctionnalités réalisées en regard de l'ambition du sujet	4
Code	Mise en oeuvre des concepts de POO Clarté des développement, respect ds conventions	3
Documentation	Exhaustivité de la documentation	3
Présentation	Qualité de l'expression orale, du support Respect des contraintes de temps Réponses aux questions	4
Démonstration	Dynamisme, clarté	2

Quelques conseils

- En matière d'analyse
  - L'analyse du sujet doit aboutir aux spécifications fonctionnelles de l'outil à développer: quels sont les différentes fonctionnalités à implémenter ? Vous pouvez essayer d'évaluer chaque développement (diagramme de GANTT par exemple) et de planifier votre projet.
  - Vous avez le droit, et c'est même bien, de faire évoluer votre modélisation après avoir commencé à développer. Cela étant, si vous vous retrouvez à modifier un diagramme de classe lors de la dernière séance de projet, c'est que vous avez raté quelque chose…
  - Prenez toujours les hypothèses qui vous arrange !
  - Énoncez dans votre rapport d'analyse les hypothèses retenues.
- En matière de gestion de projet
  - Fixez vous des objectifs à court terme et évaluez les : "ce matin, je développe telle fonctionnalité" / "j'ai perdu du temps à cause de problèmes d'imports et finalement, je n'ai pas encore complètement implémenté cette fonctionnalité". Gardez une trace de cette organisation -> cela s'intègre bien dans le diagramme de GANTT mais ça peut aussi être un document en plus de votre diagramme.
  - Organisez des points d'avancement rapides et réguliers (10min max) pour répondre aux questions suivantes : qu'est ce que j'ai terminé depuis la dernière réunion ? Qu'est ce que j'aurai terminé d'ici la prochaine réunion ? Quels obstacles me retardent ? Vous pouvez inviter l'encadrant à ces réunions en tant que "modérateur".
  - Mettez à l'écrit le bilan de de vos points d'avancement.
- En matière de développement
  - Vous allez probablement écrire beaucoup de code. Respectez des règles simples d'organisation : une classe = un fichier, 20 lignes de code max par méthodes, structure de packages, sous-packages cohérente, etc. Un code bien structuré sera plus facile à comprendre et à débugger.
  - Contrairement à un TP, au cours d'un projet, vous travaillez à plusieurs sur les mêmes fichiers, parfois dans différentes salles de l'école. Pour vous échanger et sauvegarder votre travail, vous vous rendrez compte que les clés USB, les mails ou le lecteur réseau de l'école ne sont pas toujours les plus adaptés. Utilisez une plateforme comme GitHub. Petit guide GIT http://rogerdudler.github.io/git-guide/
- Pour vos rapports :
  - Si vous abandonnez une solution en cours de route (par exemple parce que vous aurez trouvé une meilleure méthode), ne l'abandonnez pas complètement. Expliquer votre cheminement et montrer pourquoi vous avez privilégié une solution
  - Illustrez vos résultats et expliquez vos choix : une phrase comme "on choisit finalement la première solution, mais avec les bons paramètres" n'explique pas quels sont les paramètres choisis, pourquoi ils le sont, etc.
  - Citez vos sources : vous avez parfaitement le droit de vous inspirer de solutions existantes, voir de les utiliser intégralement (ne pas réinventer la roue), mais ayez l'honnêteté intellectuelle de citer les origines des contenus utilisés.
- Pour la présentation :
  - Préparez un scénario pour votre démonstration, ou encore mieux : faites une vidéo. Ouvrir l'application pour montrer qu'il se passe quelque chose lorsque l'on clique sur un bouton n'apporte rien.
  - Ne projetez pas l'intégralité de votre code lors des soutenances. Nous sommes assez grands pour aller regarder par nous même ce qui nous intéresse.

Les sujets

La difficulté indicative d'un sujet est indiquée par des *.

1. Loueur de voitures *

Il s'agit pour ce projet de réaliser une application destinée à une entreprise de location de voitures.

Les vendeurs en agence attribuent en fonction de leurs disponibilités des véhicules aux clients qui se présentent. Le montant de la location dépendra de la classe du véhicule emprunté, de la durée de la location et du kilométrage effectué (forfait à la location, avec un mécanisme de hors forfait en cas de dépassement). Le client peut également emprunter un véhicule dans une agence et le déposer à une autre mais un coût supplémentaire est appliqué dans ce cas.

Les responsables des agences indiquent dans l'application les entretiens à prévoir sur chacune des voitures qu'ils hébergent.

L'application doit permettre aux employé du siège de visualiser le stock de véhicule dans chaque agence et d'opérer à des transferts en cas de déséquilibre constaté. Des statistiques sur les agences doivent également pouvoir être éditées.

2. Réservation de billets d'avion *

Ce projet vise à proposer une application de réservation de billets d'avion destinée à être utilisée par une compagnie aérienne pour gérer les réservations des clients et optimiser ses coûts.

La compagnie aérienne dispose d'une flotte d'avions qui desservent différents aéroports. Aucun avion n'est affecté à une liaison particulière et tous peuvent effectuer presque tous les trajets. Chaque avion possède une capacité maximale de passagers et présente une autonomie maximale (exprimée en km par exemple). Nous attribuerons également des coûts à chaque vol : faire voler un gros avion peu rempli reviendra par exemple plus cher à la compagnie qu'utiliser un petit avion plein.

L'application doit ainsi permettre à la compagnie aérienne :

* de trouver un trajet reliant deux aéroports;
* d'ajouter/supprimer des voyageurs sur un vol;
* de fournir un récapitulatif d'un voyage;
* de choisir l'avion le plus économique pour un vol donné.

Vous proposez enfin des visualisations graphiques des destinations desservies (par jour, semaine ou mois, éventuellement en ajoutant la référence de l'avion effectuant la liaison).

3. Gestion de bien immobiliers */**

Il s'agit pour ce projet de réaliser une application bureautique destinée à une entreprise de gestion de biens immobiliers.

Différents types de biens immobiliers seront envisagés : appartements, maisons, chateaux, parkings ….
Différents types de transactions devront également être possibles : location, vente, vente en viager ….

L'accès à l'application se fera suivant 4 profils : public (particulier non authentifié cherchant un bien à louer ou acheter), client de l'agence (particulier cherchant à louer ou vendre un bien), agent immobilier (employé de l'agence) et responsable d'agence (agent immobilier possédant des responsabilités managériales).

Un particulier non authentifié doit pouvoir rechercher un bien selon différents critères (surface, nombre de pièces, de chambres, emplacement, distance aux commodités, prix, etc.) et demander à l'agence un rdv pour une visite. Si il est intéressé par le bien, il peut envoyer un dossier de candidature à l'agence.

Un client de l'agence immobilière doit pouvoir demander la création d'un compte (donc devenir client de l'agence), demander une estimation de son bien et enfin rédiger une fiche descriptive (=annonce) du bien et demander à l'agence de la publier. Il doit également classer les dossiers des particuliers intéressés par le bien.

Un agent immobilier publie les annonces rédigées par les clients de l'agence, estime les biens immobiliers, organise les visites et conclut les transactions immobilières.

Le responsable d'agence est un agent immobilier possédant d'autres responsabilités : celle de valider les comptes des clients, et d'attribuer aux agents immobiliers une prime d'intéressement fonction de leur performance de vendeur. Ils peuvent également éditer des statistiques sur les différents biens immobiliers publiés, les transactions effectuées, etc.

Vous êtes libres de vous réapproprier le sujet et de modifier / ajouter certaines contraintes ….

4. Application pour une salle d'escalade **

L'objet de ce projet est de réaliser une application pour une salle d'escalade.
Différents utilisateurs : gestionnaires de la salle, ouvreurs et grimpeurs.

Pour les grimpeurs l'application permettra :

- De créer un profil (pseudo, âge, type de grimpe(à vue, après travail, en tête, salle, dévers ...), chaussons préférés, niveau …..)
- De suivre et gérer sa progression :
    - De connaître le nombre de voies effectuées par session et leur difficulté
    - Avoir un historique des voies parcourues selon l'essai effectué (à vue, après travail, flashé..)
    - Faire des analyses croisées des performances en fonction des essais. il sera également intéressant de permettre les agrégations par semaine et/ou activité.
    - Participer à la qualité d'ouverture en donnant son avis sur la justesse de la cotation et la qualité des voies et en y associant des commentaires

- Poster des petites annonces pour vendre ou acheter du matos
- Rechercher un ou des partenaires de grimpe
- Créer des alertes personnalisées lorsque qu'une voie dans son niveau est ouverte ou fermée
- Accéder aux autres commentaires des grimpeurs sur les voies
- Participer à un événement organisé par la salle.

Pour les ouvreurs, qui sont aussi des grimpeurs, l'application permettra de simplifier la planification et de s'adapter aux exigences des grimpeurs

- Modifier la difficulté d'une voie en fonction des remarques des grimpeurs
- Indiquer l'ouverture d'une nouvelle voie (difficulté, couleur des prises, secteur, type (dynamique, dalle, devers, toit, à doigts, souplesse……)
- Indiquer la fermeture d'une voie
- Consulter des statistiques sur les réussites des voies (pourcentage, pourcentage de réussite à vue, flash, après travail, etc.)

Pour les gestionnaires de la salle l'application permettra de gérer le planning des ouvertures de voies, d'affecter les ouvreurs sur les différents secteurs afin de garantir une rotation des ouvreurs, organiser des événements (challenges, soirées …) et de notifier les grimpeurs.

5. Service de livraison à domicile **

Ce sujet traite de la gestion d'un service de livraison à domicile : validation des commandes clients et planification des livraisons. L'ensemble des biens vendus sont stockés dans un unique entrepôt.

Les clients passent des commandes via un site web (hors du sujet du projet). Elles doivent alors être validées, via l'application à développer, par le service d'administration des ventes qui édite également une facture.

Le service des livraisons peut alors livrer la commande. Pour cela, un livreur présent à l'entrepôt sélectionne via l'application les commandes dont il compte s'occuper. En fonction des lieux de livraison, l'application édite alors l'itinéraire optimal pour une tournée.

Les livreurs se déplacent à vélo et disposent d'une capacité de transport limitée (poids et volume). Le développement d'un module destiné à affecter de manière optimale les commandes aux livreurs pourra être envisagé.

6. Pokemon **

Pour ce projet, il s'agit de réaliser un jeu inspiré de la série de jeux vidéos Pokemon (https://fr.wikipedia.org/wiki/Pok%C3%A9mon_(s%C3%A9rie_de_jeux_vid%C3%A9o)).
Pour une première partie du travail, il s'agira de réaliser un jeu sans interface graphique où les événements surviennent de manière plus ou moins aléatoire.
Le périmètre exact du jeu développé sera à définir précisément en début de projet.

7. Risk **

Pour ce projet, il s'agit de réaliser un jeu de Risk (https://fr.wikipedia.org/wiki/Risk).

Dans un premier temps, vous vous contenterez de réaliser un jeu avec bureautique n'intégrant que les parties humain contre humain.
Lorsque le jeu sera opérationnel, vous pourrez ajouter une intelligence artificielle permettant les parties humain contre ordinateur.

8. Super Planning **/***

Ce projet consiste à créer une application de saisie d'emploi du temps et de notes dans une école.
Le personnel administratif de l'école doit pouvoir saisir l'emploi du temps (ajout/suppression de cours avec affectation des salles, des professeurs et des promotions). Le professeur intervient pour saisir les notes des cours dont il a la charge. Enfin l'étudiant utilise l'application pour visualiser son emploi du temps et les notes qu'il a obtenu.
Il serait également intéressant qu'à la fin d'une année scolaire, le logiciel permette d'éditer automatiquement les bulletins pour l'année.
Pour rendre l'application plus fonctionnelle, on pourra aussi penser à gérer la capacités maximales de chacune des salles et les équipements dont elles disposent (salle classique ou informatique, logiciels installés, présence d'un vidéo-projecteur. . .).

9. Vie d'un écosystème ***

Ce projet s'intéresse à l'évolution d'un écosystème simplifié. La carte de la simulation sera couverte d'herbe, mais pourra aussi comporter des forêts, des rivières, etc.

Le système étudié est constitué de trois types d'animaux : les herbivores qui mangent de l'herbe, les carnivores qui mangent des animaux et les charognards qui mangent les restes d'animaux morts.

Pour chacun des types d'animaux vous proposerez des règles simples modélisant leurs comportements. Celui-ci pourra dépendre de l'environnement proche et de la présence d'autres animaux. Naturellement, les animaux se déplacent et se nourrissent. Vous pourrez également envisager d'implémenter un modèle d'espérance de vie et des modes de reproduction pour les différents espèces.

La simulation se jouera au tour par tour. L'état de la simulation à chacun des tours sera enregistré dans une base de données.

10. Simulation de trafic routier ***

Ce projet consiste à réaliser une application destinée à simuler le trafic routier en un noeud donné du réseau (typiquement un carrefour avec des feux tricolores). Le programme doit permettre après avoir configuré le rythme des feux et les flux entrant pour chacune des voies (nombre de voitures et piétons arrivant sur le carrefour par unité de temps), de visualiser un simulation et d'éditer des statistiques de congestions (combien de temps met une voiture / un piéton à traverser le carrefour). Ainsi, en faisant varier les rythmes des feux tricolores, le programme doit permettre de mettre en évidence un rythme idéal pour diminuer les temps de trajet des utilisateurs.

Pour que la simulation soit réaliste, il conviendra de définir quelques règles :

* distance minimale de sécurité entre les voitures / piétons;

* modèle d'accélération et de freinage;

* vitesse maximale de déplacement.