Projet intégrateur 420-DA3-AS

Phase 3: Réalisation de l’application

Rappel

Le projet est divisé en 4 phases:

- Formation de l’équipe (semaines 1 à 3)
- Analyse: analyse du cahier de charges, liste de cas d’utilisations (semaines 4-5)
- Conception: Interactions-séquences des cas d’utilisations (semaines 6 à 8)
- Réalisation: production du code, revue et tests de qualité (semaines 9 à 15/fin)

Table des matières

Consignes

Vous avez jusqu'au **21 décembre à Midi** du cours pour compléter la phase de réalisation (production de l'application). Lisez attentivement la section Consignes et suggestions du présent document; à partir du cahier de charges de la phase d'analyse, de votre analyse des cas d'utilisation, ainsi que des documents de conception, vous devrez réaliser l'application requise. Vous recevrez dans les prochains jours un dossier de conception étoffé avec des diagrammes suggérant une approche pour la réalisation. ***Vous n'êtes pas obligés de suivre ce document de conception; cependant je vous le conseille fortement vu qu'il est fait en sens inverse de la méthode normale, c'est à dire à partir d'une application fonctionnelle, et donc, il est garanti de représenter un système fonctionnel aussi.***

Consignes et suggestions

Éléments requis absolument

- Vous devez **impérativement** réaliser le projet sous la forme d'une application native (pas web) dans le langage C# (version 10 ou inférieure; la version par défaut avec *.NET 6.0*), pour la plateforme *.NET 6.0* et utiliser les technologies *Windows Forms*, *Entity Framework Core* ainsi qu'un serveur Sql Server 2022.
- Vous devez **impérativement** utiliser le système de gestion de versions Git pour travailler en commun sur un seul projet. Poussez vous changements régulièrement et tirez (pull) avant de travailler sur le projet. ***Contactez-moi en cas de problème***.

Détails des fonctionnalités requises de l'application

NOTE : les points de sous-fonctionnalités impliquées ne sont pas exhaustives et ne sont là qu'à titre informatif pour vous aider à la planification. À vous de voir selon la logique de votre application ce qui est nécessaire pour atteindre les fonctionnalités principales.

Tous les utilisateurs doivent pouvoir :

- s'authentifier et se connecter à l'application.

Un Administrateur (utilisateur possédant rôle administrateur) doit pouvoir :

- Faire certaines des 4 opérations de base (create, read, update et delete, CRUD) pour TOUTES les entités (DTOs) :
  - Address (Adresse enregistrée) : CRUD; implique :
    - pour RUD : sélection préalable ***par recherche filtrée*** d'Address existante
  - Client (compagnie cliente) : CRUD; implique :
    - pour RUD : sélection préalable ***par recherche filtrée*** de Client existant
    - pour CU : sélection de Warehouse et d'Address
    - création intégrée ***optionnelle*** (pas obligé) de Warehouse et d'Address
  - Warehouse (Entrepôt) : CRUD
    - pour RUD : sélection préalable de Warehouse existante
    - pour CU : sélection d'Address existante
    - création intégrée ***optionnelle*** (pas obligé) d'Address
  - Product (Produit) : CRUD; implique :
    - pour RUD : sélection préalable ***par recherche filtrée*** de Product existant
    - pour D : le produit doit avoir une quantité en stock de 0
    - sélection de Client et de Supplier
    - création intégrée ***optionnelle*** (pas obligé) de Client et de Supplier
  - Supplier (Fournisseur) : CRUD; implique :
    - pour RUD : sélection préalable ***par recherche filtrée*** de Supplier existant
    - sélection d'Address existante
    - création intégrée ***optionnelle*** (pas obligé) d'Address
  - ShippingOrder (Ordre d'expédition) : CRD + U si statut = NEW; implique :
    - pour RUD : sélection préalable ***par recherche filtrée*** de ShippingOrder existante
    - pour CU : sélection ***par recherche filtrée*** de Product existants à ajouter/retirer à la ShippingOrder, incluant une quantité à ajouter/retirer
    - sélection de Warehouse, de Client, et de User (où rôle = employé d'entrepôt)
    - création intégrée ***optionnelle*** (pas obligé) de Client
    - la création d'une ShippingOrder change la quantité en stock des Product lui étant associée
  - Shipment (Livraison) : CR (pas UD); implique :
    - pour R : sélection ***par recherche filtrée*** de Shipment existant

        - sélection de ShippingOrder (où le ShippingOrder n'a pas déjà de Shipment associée)
    - PurchaseOrder (Ordre de restockage) : CR (pas UD); implique :
        - pour C : sélection ***par recherche filtrée*** de Product existant
        - pour R : sélection ***par recherche filtrée*** de PurchaseOrder existate
    - User (Utilisateur) : CRUD, implique :
        - pour RD : sélection d'un User existant
        - pour CU : sélection de Role a ajouter au User (où le User ne possède pas déjà le Role)
        - pour CU : sélection de Role (où le User possède déjà le Role) à retirer du User
    - Role : CRUD, implique :
        - pour RD : sélection d'une User existant
        - pour CU : sélection de Role a ajouter au User (ou le User ne possède pas déjà le Role)
        - pour CU : sélection de Role (où le Role est déjà associé au User) à retirer

Un Employé de bureau (utilisateur possédant le rôle correspondant) doit pouvoir :

- Procéder à la gestion des ShippingOrder et des Client, et tout ce qui va avec :
    - Création de ShippingOrder; implique :
        - sélection ***par recherche filtrée*** de Product existants à ajouter/retirer à la ShippingOrder
        - sélection de Warehouse et de Client
        - création intégrée ***optionnelle*** (pas obligé) de Client
        - la création d'une ShippingOrder change la quantité en stock des Product lui étant associée
    - Gestion de Client : CRU implique :
        - Création de nouveaux Client et tout ce qui va avec
            - sélection de Warehouse existant
            - création d'Address
        - Modification/Visualisation de Client
            - sélection préalable ***par recherche filtrée*** de Client existant
            - gestion des Product pour un Client - CRUD
                - pour C de nouveau Product avec sélection ou création de Supplier et ce qui va avec.
                - pour RUD de Product du Client
                    - sélection ***par recherche filtrée*** de Product existant
                    - suppression seulement si quantité en stock du Product est 0

Un Employé d'entrepôt (utilisateur possédant le rôle correspondant) doit pouvoir :

- Visualiser sous forme de liste les ShippingOrder dont le statut est 'créée' et non-assignées à un autre User employé d'entrepôt de Client assignés au Warehouse où il travaille.
- Visualiser sous forme de liste les ShippingOrder assignées à lui-même et non-complétées.

- Visualiser sous forme de liste les PurchaseOrder non-complétées de Product assigné au Warehouse où il travaille.
- S'assigner une ShippingOrder nouvellement créée et non-assignée à un autre User employé d'entrepôt (change aussi automatiquement le statut de la ShippingOrder à 'en cours').
- Changer le statut d'une ShippingOrder lui étant assignée dont le statut est 'en cours' à 'ramassée' (crée automatiquement un Shipment pour la PurchaseOrder).
  - sélectionner le fournisseur pour la création du Shipment.
- Changer le statut d'une ShippingOrder lui étant assignée dont le statut est 'ramassée' à 'complétée'.
- Changer le statut d'une PurchaseOrder non-complétée d'un Product assigné au Warehouse où il travaille de 'en cours' à 'complétée'; i.e. recevoir une PurchaseOrder (change la quantité en stock du Product de la PurchaseOrder).

Le Système (actions automatiques) doit pouvoir :

- Lors de la création d'une ShippingOrder (mais pas lors de la modification manuelle d'un Product), pour chaque Product dont la quantité en stock tombe en bas de 50% de sa quantité en stock visée, création automatique d'une PurchaseOrder pour le Product.

Données initiales minimales à faire insérer par migration

- 3 Rôles, un pour chaque type d’emploi
- 3 Utilisateurs, un par rôle
- 1 Entrepôt
- 2 Clients
- 3 Produits (au moins un par client)
- 2 Fournisseurs (au moins un par client)
- Autant d’adresses que nécessaires

Horaire/Calendrier (fortement) suggéré

N'oubliez pas de pousser vos changements vers GitHub régulièrement!

Semaine 1 : Bases

- Création de toutes les classes prévues (vides)
- Création des classes-entités

Semaine 2 : Opérations de base avec la BdD

- Configuration des classes-entités (classe de contexte)
- Création des classes DAO (fonctionnement initial)
- Création de classes-service (fonctionnement initial)

Semaine 3 : Début implémentation des fonctionnalités

Suggestion - commencez par la section des fonctionnalités de l'Administrateur)

- Système de Login/Facades
- Continuation des classes DAO et services avec ajouts suivant les fonctionnalités requises.
- Début des interfaces graphiques

Semaine 4+ : Suite et fin des fonctionnalités

- Continuation des interfaces graphiques, classes-service et des classes DAO jusqu'à complétion de l'application

Éléments de la correction

Critères de base

Éléments critiques de l'application (échec si non-respectés)

- Application native Windows Forms pour *.NET 6.0*. (fourni)
- Codé en utilisant le langage C# version 10 ou inférieure (fourni)
- Supporte une base de données *SQL Server 2022*
- Utilise le framework *Entity Framework Core* (fourni)
- Implémentation du modèle à trois tiers minimale (fourni)
- Tout cas catastrophique de fonctionnalités manquantes/non-fonctionnelles

Application et code fonctionnel (45%)

- Le code est libre d'erreurs majeures, s'exécute correctement et les fonctionnalités requises sont fonctionnelles. (45%)
  - Fonctionnalités implémentées tel que requis (5%)
  - Entités, configuration des entités, vérifications anti-concurrence et données de test inclues (15%)
  - DAOs et instructions EF Core (13%)
  - Services et façades (12%)
- Toute fonctionnalité supplémentaire non complétée ou non fonctionnelle doit être gérée avec grâce (correction négative).

Qualité du code (40%)

- Validation des données entrées (5%)
- Visibilité des objets/membres (sécurité interne) (2.5%)
- Membres et variables fortement typées (5%)
- Méthodes fonctionnellement atomiques (5%)
- Gestion des exceptions (5%)
- Respect des conventions de nommage (5%)
- Séparation correcte des éléments fonctionnels entre les trois tiers (5%)
- Documentation du code (7.5%)

Design de l'application (15%)

- Choix de contrôles appropriés, UX claire, facile et intuitive (12%)
- Interface dynamique : s'adapte en taille (3%)

Grille de correction

Élément de compétence : Analyser le projet de développement de l'application. (00SS.1)	
Critères de performance	**Poids**
1.2 Détermination correcte des tâches à effectuer. (app fonctionnelle – fonctionnalités)	/5
Élément de compétence : Préparer l'environnement de développement informatique. (00SS.2)	
Critères de performance	**Poids**
2.1 Installation correcte des logiciels et des bibliothèques sur la plateforme hôte.	**absolu**
Élément de compétence : Préparer la ou les bases de données. (00SS.3)	
Critères de performance	**Poids**
3.1 Création ou adaptation correctes de la base de données locale ou de la base de données distante. (app fonctionnelle – Entités configuration fonctionnelle pour migrations)	/3
3.2 Insertion correcte des données initiales ou des données de tests. (app fonctionnelle – Entités données initiales)	/5
3.3 Respect du modèle de données. (app fonctionnelle – Entités configuration)	/5
Élément de compétence : Générer ou programmer l'interface graphique. (00SS.4)	
Critères de performance	**Poids**
4.1 Choix et utilisation appropriés des éléments graphiques pour l'affichage et la saisie. (UI-UX)	/12
4.3 Adaptation de l'interface en fonction du format d'affichage et de la résolution. (UI-UX)	/3
Élément de compétence : Programmer la logique applicative. (00SS.5)	
Critères de performance	**Poids**
5.1 Programmation ou intégration correctes de mécanismes d'authentification et d'autorisation. (app fonctionnelle – Services services limités par façade)	/2
5.2 Programmation correcte des interactions entre l'interface graphique et l'utilisatrice ou l'utilisateur. (app fonctionnelle – Services fonctionnalités fonctionelles)	/10
5.3 Choix approprié des clauses, des opérateurs, des commandes ou des paramètres dans les requêtes à la base de données. (app fonctionnelle – DAOs requetes)	/10
5.4 Manipulation correcte des données de la base de données. (app fonctionnelle – DAOs actions)	/3
5.5 Programmation correcte de la synchronisation des données. (app fonctionnelle – Entités anti-concurrence)	/2
5.8 Application rigoureuse des techniques de programmation sécurisée. (qualité du code – validation, visibilité, typage)	/12.5
Élément de compétence : Contrôler la qualité de l'application. (00SS.6)	
Critères de performance	**Poids**
6.2 Revues de code et de sécurité rigoureuses. (fonctions atomiques, exceptions)	/10

6.4 Respect des procédures de suivi des problèmes et de gestion des versions.	**absolu**
6.5 Respect des documents de conception. (qualité du code – conventions et 3 tiers)	/10
Élément de compétence : Rédiger la documentation. (00SS.8)	
Critères de performance	**Poids**
8.1 Détermination correcte de l'information à rédiger. (qualité du code - doc)	/2.5
8.2 Notation claire du travail effectué. (qualité du code - doc)	/5