PROJET À COMPLÉTER

Testez vos développements Java

70 heures

Mis à jour le lundi 29 juin 2020

Pré-requis

- Développer une application JEE
- Configurer le frawmework Spring avec XML
- Mettre en œuvre et utiliser une base de données

Contexte

Votre équipe est en train de réaliser un système de facturation et de comptabilité pour un client. Le développement a débuté depuis quelques temps et vous devez commencer à vérifier que l'application fonctionne correctement, qu'elle répond bien aux règles de gestion et les respecte.

Extrait du dossier de spécifications fonctionnelles

Voici un extrait du dossier de spécifications fonctionnelles sur lequel vous vous appuyez pour écrire vos tests.

Comptabilité

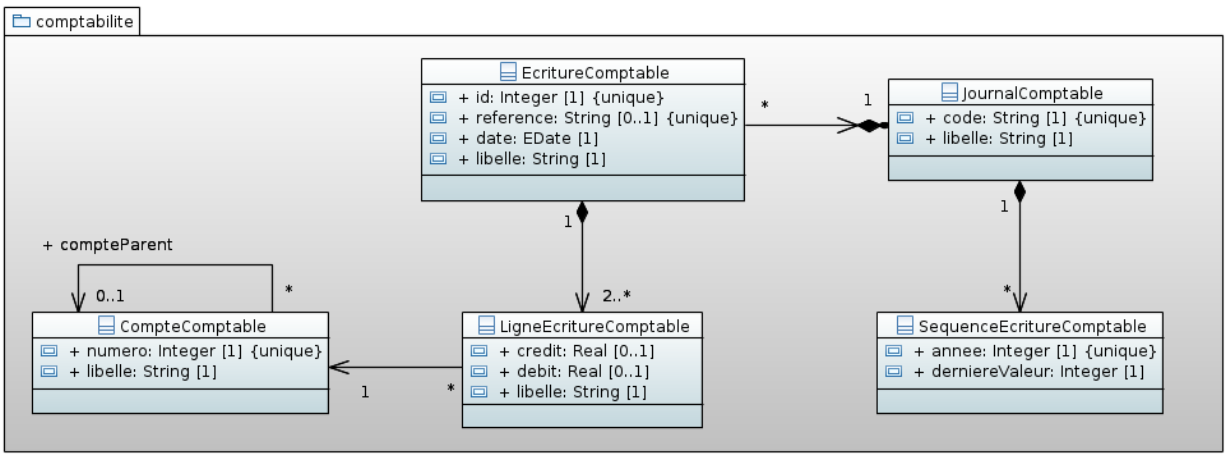

Diagramme de classes - Package Comptabilité

En ce qui concerne la gestion de la comptabilité, l'application s'appuie sur les principes de la **comptabilité en partie double**.

Règles de gestion de la partie comptabilité :

ID	Description
RG_Compta_1	Le solde d'un compte comptable est égal à la somme des montants au débit des lignes d'écriture diminuées de la somme des montants au crédit. Si le résultat est positif, le solde est dit "débiteur", si le résultat est négatif le solde est dit "créditeur".
RG_Compta_2	Pour qu'une écriture comptable soit valide, elle doit être équilibrée : la somme des montants au crédit des lignes d'écriture doit être égale à la somme des montants au débit.
RG_Compta_3	Une écriture comptable doit contenir au moins deux lignes d'écriture : une au débit et une au crédit.
RG_Compta_4	Les montants des lignes d'écriture sont signés et peuvent prendre des valeurs négatives (même si cela est peu fréquent).
RG_Compta_5	La référence d'une écriture comptable est composée du code du journal dans lequel figure l'écriture suivi de l'année et d'un numéro de séquence (propre à chaque journal) sur 5 chiffres incrémenté automatiquement à chaque écriture. Le formatage de la référence est : XX-AAAA/#####. Ex : Journal de banque (BQ), écriture au 31/12/2016 --> BQ-2016/00001
RG_Compta_6	La référence d'une écriture comptable doit être unique, il n'est pas possible de créer plusieurs écritures ayant la même référence.
RG_Compta_7	Les montants des lignes d'écritures peuvent comporter 2 chiffres maximum après la virgule.

Ressources

Les ressources de base pour débuter ce projet sont disponibles ici : fichier ZIP de ressources

ressources.

> La partie IHM n'est pas intégrée dans le code source disponible ci-dessus. Ceci est normal, il ne vous est pas demandé de réaliser des tests sur cette partie.
> Concentrez-vous sur les tests du cœur de l'application (couches *model*, *business* et *consumer*).

Travail demandé

Les tests

Votre travail consiste à réaliser 2 types de tests :

- des **tests unitaires** : leurs objectifs sont de valider les règles de gestion unitaires de chaque "composant" de l'application
- des **tests d'intégration** : leurs objectifs sont de valider la bonne interaction entre les différents composants de l'application

À vous de définir vos tests et la stratégie que vous allez mettre en place pour les tests d'intégration.

Vous implémenterez et automatiserez ces tests à l'aide de JUnit, Mockito, Maven et Travis CI / GitLab CI / Jenkins.

Les tests seront lancés via le build Maven.

Les tests d'intégration font l'objet de profils Maven spécifiques (cf. le fichier `pom.xml` du projet parent fourni).

Cet environnement sera construit (à partir des éléments disponibles dans le dépôt Git du projet) et monté à la volée par votre système d'intégration continue.

> Quelques erreurs ont été volontairement disséminées dans le code de l'application. Il vous faudra toutes les corriger.
> Si les tests que vous allez créer sont pertinents, aucun soucis, vous devriez toutes les relever !
> Vos tests devront donc présenter une très bonne couverture de votre code.

> N'hésitez pas à améliorer le code ou faire du refactoring pendant l'élaboration des tests. Vous augmenterez ainsi la maintenabilité et la lisibilité du code et faciliterez également le débogage de l'application.
> Vous pouvez par exemple :
>
> - tracer les requêtes SQL exécutées
> - compléter les messages des exceptions (détail des violations de

- compléter les messages des exceptions (détail des violations de contraintes...)

- tracer les informations de règles de gestion violées avec leur identifiants associés dans le dossier de spécifications

Complément d'implémentation

Vous compléterez également l'implémentation de l'application en vous appuyant sur les commentaires `TODO` insérés dans le code source. Vous devez tous les réaliser, il ne devra rester aucun `TODO` dans votre livrable.

> La plupart des EDI peuvent afficher une liste des `TODO`, sinon une simple recherche dans les fichiers du projet les relèvera.

Bien évidemment, tout le code supplémentaire que vous aller écrire pour implémenter les `TODO` sera testé par vos tests unitaires/d'intégration !

Livrables attendus

Le code doit être géré avec Git.

Vous livrerez, sur GitHub ou GitLab , dans un seul dépôt **privé** et **dédié** :

- Le code source de l'application avec vos ajouts, modifications, corrections
- Les éléments d'automatisation des tests unitaires
- Les éléments d'automatisation des tests d'intégration
- La configuration du serveur d'intégration

> Vous transmettrez le(s) lien(s) vers le(s) dépôt(s) dans un fichier texte que vous uploaderez sur la page du projet **avant la soutenance**.
> Vous donnerez un accès en lecture à votre/vos dépôt(s) Git à votre mentor et à l'évaluateur qui vous fera passer la soutenance

Soutenance

Avant la soutenance

Quelques jours avant la date de la soutenance (en général 3-4 jours avant), vous transmettrez tous vos livrables au mentor qui vous fera passer celle-ci, afin qu'il puisse en prendre connaissance en amont et éventuellement tester votre application.

Déroulement de la soutenance

Il vous est demandé de vous mettre en situation réelle : en effet, il s'agit d'une **réunion professionnelle**.
Vous vous adresserez à votre chef de projet qui doit valider votre travail.

La soutenance, d'une durée d'environ 25-30 minutes, se déroulera comme ceci :

- Partie 1 – 20 minutes : **Simulation d'une réunion professionnelle**
  L'évaluateur jouera le rôle du chef de projet
  - [~ 10 minutes] En vous appuyant sur un support adapté que vous aurez préparé, vous lui présenterez
    1. les éléments de tests unitaires et tests d'intégration que vous avez mis en place
    2. votre stratégie de test d'intégration.
  - [~ 10 minutes] Vous ferez une démonstration de l'exécution de ces tests et montrerez le niveau de couverture du code par ceux-ci.
- Partie 2 – 5-10 minutes : **Retour sur la soutenance**
  - L'évaluateur pourra vous demander d'approfondir certains aspects ou vous questionner sur vos livrables.
  - L'évaluateur vous fera un debrief sur votre prestation en soutenance.

Référentiel d'évaluation

La validation du projet implique l'acquisition de l'ensemble des compétences qui y sont associées. Chaque compétence est validée si *tous* les critères d'évaluation définis dans ce référentiel sont validés.

Réaliser l'audit d'un système

- Les 4 erreurs de développements dans le projet fourni sont identifiées et résolues.
- Le développement a été complété en suivant les TODO.

Mettre en place une démarche qualité et sa méthodologie

- Les tests unitaires ont été réalisés à l'aide de JUnit.
- Les tests d'intégration ont été réalisés à l'aide des "profiles" Maven.
- L'ensemble des modules a un code coverage de 75% minimum.

Gérer l'évolutivité et l'adaptabilité d'un système

Gérer l'évolutivité et l'adaptabilité d'un système

- Un serveur d'intégration est installé et configuré (Jenkins / Travis CI / GitLab CI au choix).
- Un rapport d'exécution des tests est automatiquement généré à chaque commit.
- Un logiciel de versionning a été correctement utilisé.

Compétences évaluées


Réaliser l'audit d'un système


Gérer l'évolutivité et l'adaptabilité d'un système


Mettre en place une démarche qualité et sa méthodologie