

UNIVERSITÉ DU QUÉBEC À MONTRÉAL

MGL7460-90 HIVER 2020

REALISATION ET MAINTENANCE DE LOGICIELS

PROJET TECHNIQUE / FINAL

PRÉSENTÉ PAR

MAJED ABIDI

JEAN-PIERRE PASCAL SUDRE

26 AVRIL 2020

Équipe Enseignante :

Professeur : Sébastien Mosser (UQAM)

Laboratoires : Jean-Philippe Gélinas (SFL, Lévis), Jonatan Cloutier (SFL, Montréal)

## Table des matières

1. Introduction.....	3
2. Objectifs.....	3
3. Réalisation de l'application et mise en place du pipeline continu .....	4
4. Les forces et faiblesses de l'application développée et du pipeline. ....	7
5. Prise de recul sur le travail effectué .....	8

## 1. Introduction

Dans ce projet technique l'hypothèse de travail repose sur le fait que nous nous intéressons à la définition d'un logiciel de gestion de produits bancaires.

Un premier module est à destination des employés de la banque, pour créer des comptes clients et gérer ceux-ci. Un second module est destiné aux clients, qui peuvent consulter les produits qu'ils ont dans la banque, et y apporter des modifications. Ces deux modules partagent une dépendance vers le système de la banque.

Les modules des employés et des clients fonctionnent sous la forme d'application en ligne de commande, non interactive. La communication entre les modules est basée sur des communications directes entre les classes. Nous ne prenons pas en considération la sécurité, ni la persistance.

Nous avons choisi de développer notre application avec Eclipse IDE car **n'ayant pas une expertise avancée en développement ni des rôles de développeurs dans notre compagnie**, nos connaissances de Maven et/ou Gradle étaient extrêmement limitées et nous avons dû pallier cet handicap par plus de recherches personnelles accompagnées de compromis dans le développement de notre application.

Les complications rencontrées au vu des limitations de notre environnement de travail (Laptop Desjardins, (pas Administrateur sur le poste pour un des membres de l'équipe, certaines applications bloquées pour l'installation car nous n'avons pas des rôles de développeurs) avec notre implication dans des cellules de crise covid19 pour le déploiement des employés chez Desjardins en télétravail durant 3 semaines (horaire de jour et de nuit) sont venus affecter notre capacité à livrer l'application dans sa totalité. De ce fait les user stories listés ci-dessous n'ont pas été implantées dans notre version finale de l'application du Projet technique, elles sont en statut en cours dans notre Projet GitHub.

- User story #1 / Employé peut ajouter un nouveau client
- User story #3 / Employé doit pouvoir accepter l'ajout du produit pour le client
- User story #4 / Employé doit pouvoir rejeter le produit demandé par le client
- User story #5 / Employé doit être capable de lister les clients ayant des produits en attente de validation
- User story #6 / Employé doit être en mesure d'élever le statut du client qui accède à de nouveaux produits
- User story #7 / Employé doit être en mesure de diminuer le statut du client et restreindre les produits auxquels il accède

## 2. Objectifs

L'objectif est la réalisation de l'application et la mise en place du pipeline de déploiement continu. L'application reste volontairement simpliste pour les besoins du projet technique. Les

fonctionnalités sont données pour les cas nominaux. Nous avons mis en place la gestion d'erreur ainsi que la gestion des tests.

### 3. Réalisation de l'application et mise en place du pipeline continu

Dans le cadre de la phase finale nous avons révisé le projet dans son ensemble et y avons apporté les modifications nécessaires pour donner suite aux recommandations qui nous ont été faites lors de la livraison 1. Nous avons apporté les modifications suivantes :

#### 3.1. Dépôt Git public créé pour le projet

<https://github.com/Majeddesj/ProjetTechnique.git>

La user story, restructuration projet et User stories suite aux recommandations après Livraison1 #13 reflète la restructuration du projet au niveau du regroupement dans le tableau de projet GitHub :

- les fichiers d'IDE (.classpath, .project, .settings, launchers) ne sont plus présent dans le dépôt et donc ne seront plus commités sur le dépôt.
- Le fichier gitignore a été créé et commité.
- Le readme a été mis à jour et déplacé sur le dépôt avec un merge de la branch Feature vers le dépôt Master.

#### 3.2. Mise en place d'un système de suivi d'exigences

##### 3.2.1. Définition des user stories associée au projet

<https://github.com/Majeddesj/ProjetTechnique/projects/1>

et

<https://github.com/Majeddesj/ProjetTechnique/issues>

Le tableau GitHub a été regroupé en stade de développement vs la séparation initiale en module pour mieux refléter un projet géré en mode Agile.



##### 3.2.2. Estimation des stories en termes de risque technique et de valeur métier

Dans la version finale, nous avons estimés tous les récits (user stories). Dans GitHub il s'agit de Issues qui sont attachés et apparaissent sous le tableau Projet, regroupés en stade de développement pour mieux refléter les avancements du développement de notre application.



##### 3.2.2.1. Parties prenantes

- Employé
- Client

### 3.2.2.2. Liste des exigences

Priorités requises / Parties prenantes	Employé 	Client 
Doit	Créer nom client	Lister tous les produits du client
Devrait	Lister les produits du compte client	Lister tous les produits auquel le client a accès
Devrait	Élever le statut du client qui accède à de nouveaux produits	Se dissocier d'un produit avec approbation éventuelle employé
Devrait	Diminuer le statut du client et restreindre les produits auxquels il accède	
Pourrait	Accepter l'ajout du produit pour le client	S'associer à un produit avec approbation éventuelle employé
Voudrait	Rejeter le produit demandé par le client	
Voudrait	Lister les clients avec des produits en attente de validation	

### 3.2.2.3. Évaluation du risque

Priorités requises / Parties prenantes	Employé 	Client 
Doit	Créer nom client	Lister tous les produits du client
Devrait	Lister les produits du compte client	Lister tous les produits auquel le client a accès
Devrait	Élever le statut du client qui accède à de nouveaux produits	Se dissocier d'un produit avec approbation éventuelle employé
Devrait	Diminuer le statut du client et restreindre les produits auxquels il accède	
Pourrait	Accepter l'ajout du produit pour le client	S'associer à un produit avec approbation éventuelle employé
Voudrait	Rejeter le produit demandé par le client	
Voudrait	Lister les clients avec des produits en attente de validation	

### 3.2.3. Stories tracée au code via les commits

Pour refléter la recommandation de la Livraison 1 des liens du code aux exigences qui étaient manquants, les user stories principales ont été tracées au code.

### 3.3. Développement des 3 modules

#### 3.3.1. Mise en place d'un système de build tenant compte des dépendances

Les dépendances ont été créés avec Maven qui aide à définir, créer et maintenir des builds reproductibles avec des chemins de classe et des versions de bibliothèque bien définis.

Ajout dans la version finale du module : Banque.

Amélioration du main en réduisant son utilisation afin de respecter les concepts orientée objet et qu'idéalement le main ne fasse qu'initialiser l'application.

### 3.4. Tests du système

#### 3.4.1. Tests unitaires pour chaque module

Les tests unitaires n'ont été créés que sur la Base de Données et toutefois de manière « hardcoded ». Les test unitaires testent les lignes de codes hardcoded car notre classe était hardcoded. Nous comprenons que le hardcode s'applique pour la BD dans le cadre du cours et que nous ne pourrions utiliser ce mode de fonctionnement pour une application livrée en prod.

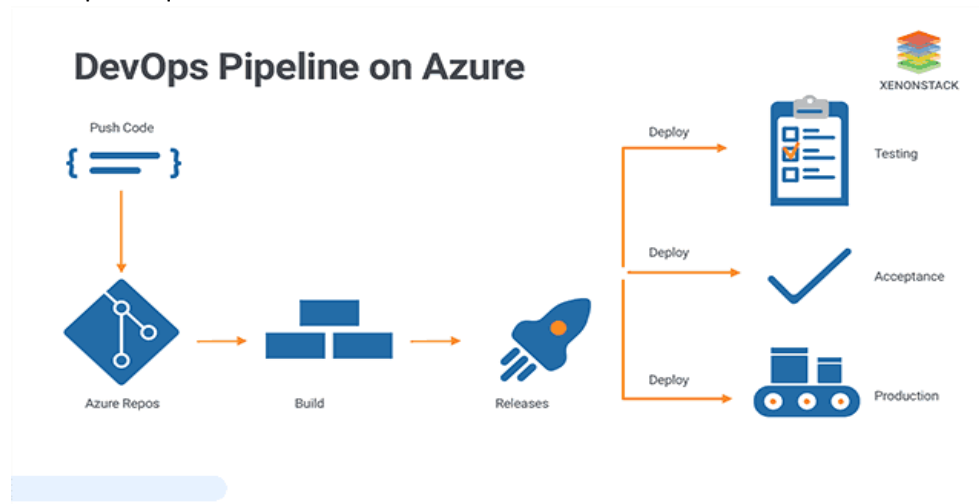
Au vu de cette définition de nos tests unitaires et des raisons d'utilisation, nous avons choisi d'utiliser JUnit5 pour effectuer nos tests unitaires.

### 3.5. Pipeline de déploiement continu

#### 3.5.1. L'outil d'intégration/de déploiement continu que nous avons utilisé, est Azure Pipelines.

Description générale de Azure Pipelines : service cloud qui peut être utilisé pour créer et tester automatiquement un projet et le mettre à la disposition d'autres utilisateurs. Il fonctionne avec à peu près n'importe quel langage ou type de projet.

Azure Pipelines combine l'intégration continue (CI) et la livraison continue (CD) pour tester et créer le code en permanence et de manière cohérente et l'expédier à n'importe quelle cible.



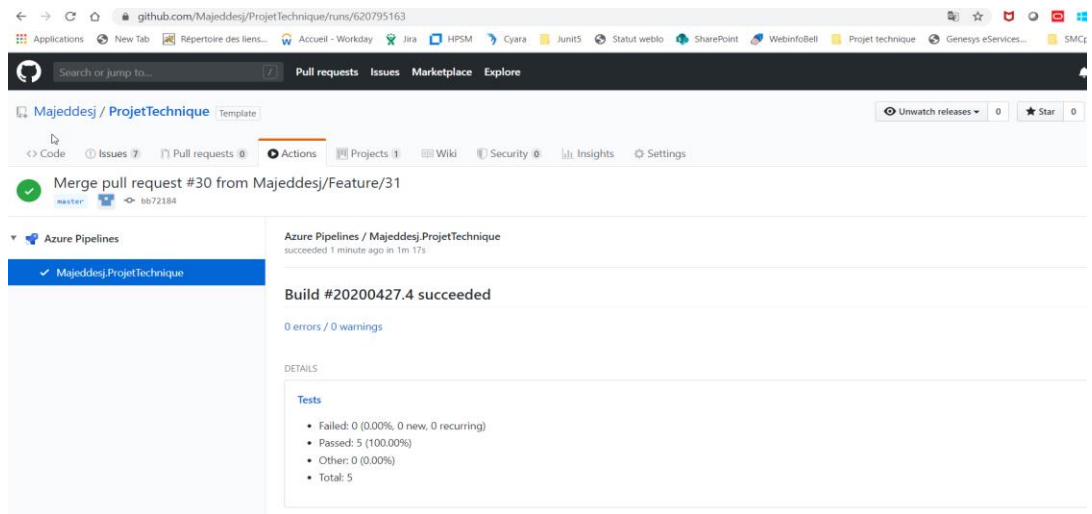
### 3.6. Mise en place de scénarios de démonstration

#### 3.6.1. Démonstration de la valeur ajoutée du pipeline de déploiement continu pour le développement du produit.

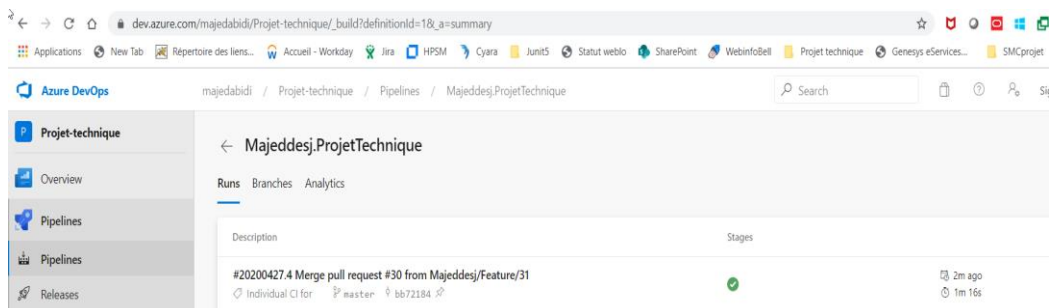
Dans le cadre de notre projet technique, le déploiement en continu est fonctionnel à 100%. Dès qu'un merge est fait sur le master, un nouveau build/test est automatiquement créé sur notre pipeline et met notre package à jour sur le cloud.

Voir résultats ci-dessous:

#### Exécution de Azure Pipeline via Github lors d'un merge vers Master



#### Exécution du pipeline automatiquement lors d'un merge vers Master sur Dev Azur



## 4. Les forces et faiblesses de l'application développée et du pipeline.

### 4.1. Les forces de l'application développée

L'application dans son état actuel est fonctionnelle mais incomplète. Les dépendances sont gérées efficacement par Maven, ce qui nous permettra d'ajouter des

bibliothèques tierces à notre application. Les tests unitaires partiels ont été effectués avec succès. Le code développé est de bonne qualité.

#### 4.2. Les faiblesses de l'application développée

Les user stories n'ont pas toutes été tracées au code, la base de données n'est pas totalement indépendante (il ne s'agit pas d'un backend) ce qui ne nous a pas permis de finaliser toutes les user stories.

#### 4.3. Les forces du pipeline

L'utilisation de Azure pipeline, nous permet d'automatiser et simplifier le déploiement de l'application. Avec cette approche d'intégration continue et de déploiement continu généré automatiquement et entièrement intégré, l'application peut être mise à jour chaque fois que le code source est modifié. Nous pourrions créer et suivre les backlogs, gérer des dépôts de code et améliorer la collaboration entre les membres de l'équipe.

#### 4.4. Les faiblesses du pipeline

Le fait que la BD n'est pas totalement indépendante ne nous permet pas de tirer profit du plein potentiel du pipeline d'intégration.

## 5. Prise de recul sur le travail effectué

#### 5.1. Prise de recul par membre équipe projet / Jean-Pierre

Le travail effectué dans le cadre de ce projet technique m'a permis seulement d'effleurer et d'entrevoir la complexité de la réalisation et la maintenance de logiciels. J'ai pu explorer les différents outils de développement et d'intégration (Maven, Gradle, Cucumber, Github, Azure...) qui m'étaient jusque là inconnus.

Venant d'un environnement TI plus infrastructure (réseautique et téléphonie) avec des connaissances en développement très superficielles, ce cours avec ce type de projet technique qui mêle développement et analyse de code, ne cadre pas du tout avec mon champ de compétences ni mes intérêts professionnels.

Les difficultés rencontrées dès le début du cours au niveau de l'environnement technologique UQAM/Desjardins suivi de la pandémie du covid19 ont ajoutés aux manques de connaissances en développement et accroître la difficulté de livrer des projets complets et de qualité (aussi bien projet personnel que technique en équipe). Toutefois, j'aurai pu mettre en pratique certains concepts du cours précédent Principes et applications de la conception de logiciels et j'ai investi beaucoup d'heures d'efforts même si les résultats ne le reflètent pas.

#### 5.2. Prise de recul par membre équipe projet / Majed

Le travail effectué pendant ce projet technique m'a permis d'apprendre et de pratiquer les principes de la maintenance et l'évolution du logiciel.

Je liste ci-dessous les aspects techniques et méthodologiques que j'ai appris durant le projet :

- L'intégration continue



- Le développement des tests Unitaire en utilisant Junit, même si ce n'était que la première fois que j'utilisais ce type de test (Livre utilisé : Junit in action)
- Les différents outils de développement et d'intégration (Maven, Gradle, Cumcomber, Github, Azure...)
- Tous les changements que nous avons effectué du L0 du projet au L1 cela m'a permis d'apprendre et de progresser
- Le réusinage (refactoring) du logiciel et l'analyse des impacts des changements
- Méthodologie Devops ainsi Azure pipeline.