

Sohbi Elias

Génie Logiciel

Dani Oumayma

Wolfart Remi

GitHub : <https://github.com/RemiWolfa/projet-gl>

Trello : <https://trello.com/b/FvvCO1LE>

Contexte et outils :

Dans le cadre de la réalisation du projet de Génie Logiciel de M1 MIAGE, nous avons développé une application permettant de gérer un restaurant, en fonction de notre statut, de nos fonctions. Sans passer par une véritable interface, nous avons su simuler le confort qu'elle représente en affichant les différentes commandes possibles sous forme de chaîne de caractère. Le but de ce projet était avant tout d'être optimisé au possible, et a fait l'objet de différentes modifications.

Actuellement terminé, nous avons fait les sprints 1, 2 et 3. Les exigences concernant l'environnement de développement en vue de son intégration se sont portées principalement sur l'utilisation de l'IDE Eclipse et du logiciel Git pour partager nos fichiers. De plus, pour travailler avec une base de données, le choix s'est porté sur mongoDB qui est une base de données orientée document.

Utilisation :

Pour utiliser le logiciel, il faut le lancer depuis eclipse en le compilant comme étant un projet java, ou le lancer depuis le répertoire bindist à l'aide du fichier run adapté à votre OS (.bat ou .sh).

Une base de données est configurée par défaut pour visualiser les données de test, pour la consulter vous pouvez vous connectez sur le lien suivant :

<https://cloud.mongodb.com/v2/60a8f5d9b55e811cba625d2c#clusters>

Vous pouvez vous y connecter avec le compte google suivant :

projet.gl.miage@gmail.com

aa99zz++

Il nous sera alors demandé de nous connecter, en saisissant notre identifiant utilisateur. En fonction de cette valeur, notre statut (serveur, maître d'hôtel, directeur, assistant de service ou cuisinier) sera renvoyé puis les différentes fonctionnalités associées seront proposées.

Vous pouvez utiliser l'identifiant **0007** pour vous connectez en tant que directeur.

Ces fonctionnalités sont divisées dans des classes et associées aux entités correspondantes.

Par exemple, le cuisinier se chargera de récupérer les commandes, de vérifier les stocks ou encore de créer un plat tandis que le directeur va lui plutôt avoir des devoirs de supervisions comme par exemple créer et modifier un utilisateur, manipuler la carte, consulter les plats les plus rentables et voir le temps moyen que restent les clients dans le restaurant. L'assistant de service voudra vérifier les états des tables, le serveur sera concerné par les tables des réservations en cours. Le maître d'hôtel sera le « manager », car il gérera l'affectation des serveurs aux tables, la mise à jour des stocks et sera en charge de prendre les réservations.

Fonctionnalités :

Ces fonctionnalités ont été divisées et réparties dans le Trello, en sous-parties, et associées à un membre de l'équipe. De plus, lorsqu'il était vraiment possible de quantifier l'urgence de cette tâche, nous avons noté la criticité associée, et également le rôle concerné lorsqu'il s'agissait d'une fonctionnalité d'un corps métier pour mieux la situer.

Pour commencer, le sprint 1 :

Sprint1 Criticité 3
Conception de l'application
OD RW

Sprint1 Criticité 3
Initialisation du projet
OD RW

Sprint1 Criticité 3
Lister les éléments à stocker en base de données
RW

Sprint1 Criticité 3
Traitement d'une cmd
ES

Sprint1 Criticité 3
Tests Unitaires
OD

Sprint1 Criticité 3
Tests Unitaires
OD

Sprint1 Criticité 3
Sauvegarde des différents objets dans les collections associées.
RW

Sprint1 Criticité 3
Connexion à la base de données et création des différentes collections.
RW











Sprint1 Criticité 3
Création de la base de données de dev
RW

Pour le sprint 2 :

<div> <div>Sprint2</div> <div>Criticité 2</div> </div> <div>Création d'une réservation</div> <div>RW</div>	
<div> <div>Sprint2</div> <div>Criticité 2</div> </div> <div> <div>Assistant de service</div> </div> <div>trouver plat souhaité</div> <div>RW</div>	
<div> <div>Sprint2</div> <div>Criticité 2</div> <div>Serveur</div> </div> <div>Créer une commande avec des plats</div> <div>RW</div>	
<div> <div>Sprint2</div> <div>Criticité 2</div> <div>Cuisinier</div> </div> <div>Consulter les stocks pour voir quel plat il est possible de faire.</div> <div> <div>👁</div> <div>ES</div> </div>	
<div> <div>Sprint2</div> <div>Criticité 2</div> <div>Serveur</div> </div> <div>Afficher informations de la table</div> <div>OD</div>	
	<div> <div>Sprint2</div> <div>Criticité 2</div> <div>Cuisinier</div> </div> <div>Avertir le personnel service des commandes prêtes</div> <div>RW</div>
	<div> <div>Sprint2</div> <div>Criticité 2</div> <div>Cuisinier</div> </div> <div>Définir des plats à partir des matières premières</div> <div>RW</div>
	<div> <div>Sprint2</div> <div>Criticité 2</div> </div> <div> <div>Maître d'hôtel</div> </div> <div>Affecter une table à un serveur</div> <div>RW</div>
	<div> <div>Sprint2</div> <div>Criticité 2</div> </div> <div>Créer une base pour les tests unitaires (avec des collections de tests)</div> <div> <div>👁</div> <div>ES</div> <div>OD</div> <div>RW</div> </div>

<div> <div>Sprint2</div> <div>Criticité 2</div> <div>Serveur</div> </div> <p>Afficher listes tables dont il est responsable.</p> <div>OD</div>	
<div> <div>Sprint2</div> <div>Criticité 2</div> <div>Serveur</div> </div> <p>Suivre l'état de l'occupation de chaque table (propre, sale, occupée)</p> <div>RW</div>	<div> <div>Sprint2</div> </div> <p>Tests fonctionnelles</p> <div>OD</div>
<div> <div>Sprint2</div> <div>Criticité 2</div> <div>Assistant de service</div> <div>Directeur</div> <div>Cuisinier</div> <div>Maître d'hôtel</div> </div> <p>Sprint fonctionnel : connexion --nomUser --mdp / déconnexion / inscription --nomUser --mdp --mail</p> <div>ES</div>	<div> <div>Sprint2</div> <div>Criticité 2</div> <div>Maître d'hôtel</div> </div> <p>Mettre à jours les stocks</p> <div>ES</div>
<div> <div>Sprint2</div> <div>Criticité 2</div> <div>Cuisinier</div> </div> <p>visualiser les commandes entrantes</p> <div>ES</div>	<div> <div>Sprint2</div> <div>Criticité 2</div> <div>Serveur</div> </div> <p>Afficher l'état d'avancement du repas de chaque table dont il est responsable</p> <div>RW</div>
	<div> <div>Sprint2</div> <div>Criticité 3</div> <div>Directeur</div> </div> <p>Gestion des employés (création, modification, suivi)</p> <div>RW</div>
<div> <div>Sprint2</div> <div>Criticité 2</div> </div> <p>Tests unitaires des fonctionnalités du sprint 2</p> <div>ES OD</div>	
<div> <div>Sprint2</div> <div>Serveur</div> </div> <p>Vérifier stock lors de l'ajout d'un plat à une commande, et lors de la sauvegarde de la commande décrémente les stocks</p> <div>RW</div>	

Pour le sprint 3 :

<div>Sprint3</div> <div>Modifier méthode lecture entier</div> <div> </div> <div>ES</div>	<div>Sprint3 Directeur Criticité 1</div> <div>Consulter les statistiques sur la part de chaque plat dans les bénéfices faits</div> <div> </div> <div>ES RW</div>
<div>Sprint3 Criticité 1</div> <div>Gestion des catégories</div> <div>RW</div>	<div>Sprint3</div> <div>Transformer les textes en dur en constante</div> <div>OD RW</div>
<div>Sprint3</div> <div>Factorisation du code</div> <div>RW</div>	<div>Sprint3</div> <div>Gestion de stock avec quantité (actuellement pour chaque plat, qu'une seule quantité de mp est associée)</div> <div>RW</div>
<div>Sprint3</div> <div>Utiliser des constantes pour les attributs des collections mongodb</div> <div></div> <div>RW</div>	
<div>Sprint3 Criticité 1</div> <div>Placer les menus enfants en tête de liste pour éviter les attentes trop longues</div> <div>RW</div>	
<div>Criticité 2 Sprint3</div> <div>gérer les prix</div> <div> 5  </div> <div>ES RW</div>	
<div>Sprint3 Assistant de service</div> <div>Criticité 1</div> <div>Gérer les tables (desservir une table, dresser une table)</div> <div>OD RW</div>	<div>Sprint3</div> <div>Tests fonctionnelles</div> <div></div> <div>ES OD RW</div>
<div>Sprint3 Criticité 1</div> <div>Savoir le temps que les clients passent dans le restaurant</div> <div> </div> <div>ES</div>	<div>Sprint3 Directeur Criticité 1</div> <div>Gérer la carte du jour</div> <div>RW</div>

Packages :

Le code s'est scindé en cinq packages. Le premier concerne les classes de base des éléments, telle qu'on pourrait les retrouver dans la base par exemple.

Le second liste les différentes fonctionnalités principales à exécuter, implémentant une interface d'action d'utilisateur. Le troisième contient les classes représentant les DashBoard des utilisateurs. Le quatrième concerne lui les appels à la base, les classes de Collection et les appels de méthodes faisant intervenir MongoDB.

Le dernier package contient lui les énumérations que nous avons utilisées.

Optimisation :

Dans un soucis d'optimisation, différentes techniques ont été mises en œuvre pour optimiser le code. Par exemple, aucune méthode ne rejette d'exception contrairement à ce qui était fait au début, car les try/catch sont peu souhaitables. A la place, les exceptions et erreurs sont gérées en interne et la classe Tools permet de simplifier la manipulation des outils de travail.

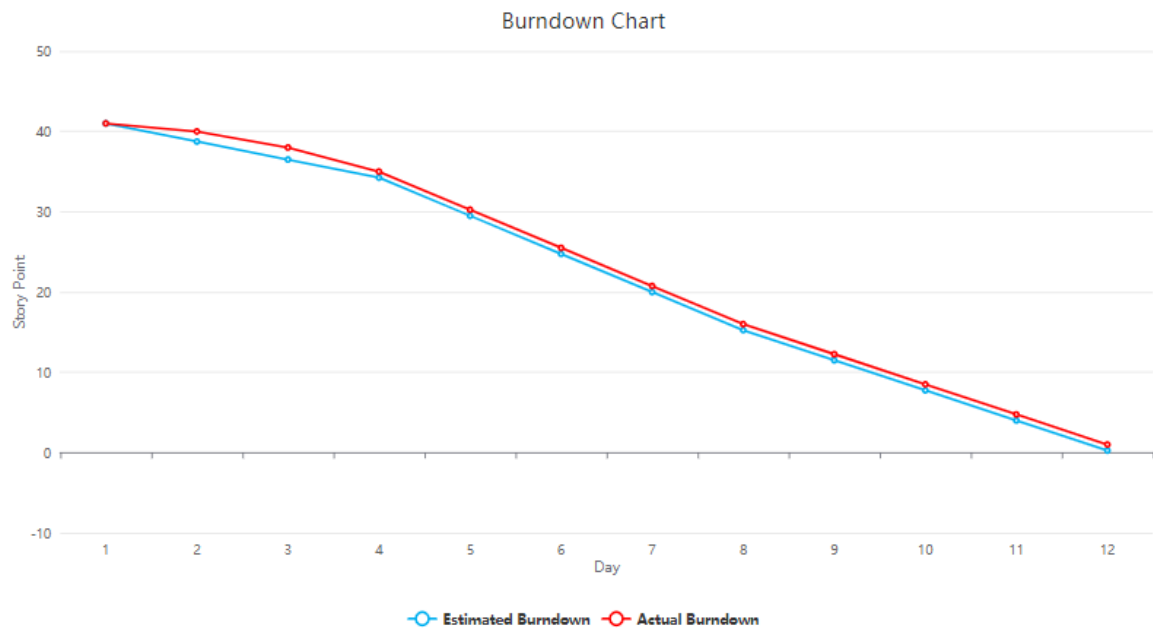
Les tests :

Concernant les tests, seules les méthodes les plus importantes ont été testées. En l'occurrence celles dépendant d'entrées utilisateurs n'ont pas fait partie des tests, mais leur utilisation nous a permis de vérifier la viabilité de nos raisonnements. Nous avons ainsi fait usage des tests unitaires, fonctionnels et d'intégration.

Ces tests nous ont permis de découvrir plusieurs bugs engendrés par l'ajout d'une nouvelle fonctionnalité.

Bien que ce ne soit pas des tests unitaires, nous avons également mis en place des tests sur les méthodes en rapport avec la base de données, et cela nous a aussi permis de découvrir des problèmes.

Burndown chart :



Utilisation de git / github

Nous avons créé une branche pour chaque fonctionnalité, sauf pour les très petites fonctionnalités qui pouvait être regrouper.

lors de la création d'une branche nous vérifions à chaque fois qu'elle était bien à jour avec master, nous récupérons donc régulièrement le contenu de master dans notre branche locale, ainsi nous n'avons pas eu beaucoup de conflits.

Pour mettre à jour master, nous utilisons des pull-requests (la branche master était verrouillée), au moins le code était vérifié par une autre personne.

Quand la branche à ajouter à master était composé de trop de commits, nous regroupions les commits en un seul.

Les messages de commits auraient pu être composé du numéro de la tâche.