Darquest, détails d'implémentation du sprint 1

Date début : semaine du 13 mars

Date remise : Le 04 avril. Date de revue (Démonstration et revue) : le 05 avril 2023

Liste des fonctionnalités : Voici la liste des fonctionnalités attendues pour le sprint 1.

Énoncé des fonctionnalité	Priorité
Rechercher des objets selon un ou plusieurs critères et voir le détail d'un objet	M
Inscription d'un joueur et connexion de celui-ci	M
Ajouter un objet au panier	M
Supprimer un objet du panier	M
Modifier la quantité du panier	M
Consulter son panier	M
Payer son panier • Mise à jour du solde du joueur • Mise à jour de l'inventaire du joueur • Mise à jour de la quantité en inventaire. (Item)	M
Consulter son inventaire	S

- Le prix d'un item est de l'ordre de 1 à 100 pièces d'or.
- Le montant initial d'un joueur est :
    - 1000 pièces d'or
    - 1000 pièces d'argent
    - 1000 pièces de bronze

Une pièce d'or vaut : 10 pièces d'argent. Une pièce d'argent vaut 10 pièces de bronze.

Stories techniques	Détails
Vérifier la connexion au serveur PHP	Chaque membre de l'équipe doit faire le test qu'il peut bien transférer une page sur le serveur
Créer et peupler la base de données	Au moins : 5 joueurs. Au moins 5 items de chaque type. (**voir exemple de procédure stockée pour simplifier l'insertion**)
Configurer GitHub	Chaque membre de l'équipe doit avoir un compte GitHub Un membre de l'équipe sera chargé de créer le dépôt dans son propre compte GitHub et doit inviter les membres de son équipe.
Créer son tableau de tâches verticale avec Trello ou Jira ou autre outil de gestion de projet	Même principe. Un membre vac créer le Trello pour le sprint 1 et invite les co-équipiers.

Les tests : À titre d'indication, voici un ensemble non exhaustif de tests unitaires qui seront réalisés pour Darquest-sprint1.

Test unitaires, le compte du joueur
- Validation de tous les champs. - L'alias est unique. - Le courriel est unique s'il y a lieu - Afficher les cas d'erreur (connexion multiple, alias dupliqué etc..) - Connexion réussie : afficher les infos pertinentes du joueur. - Connexion non réussie : afficher le message erreur

Test unitaires, recherches d'item
- Aucun critère - Tous les critères - Recherche selon un critère : Exemple Potion. Tous les critères seront testés - Recherche selon deux critères. Exemple Potion et Arme. Toutes les combinaisons seront testées - Recherche selon trois critères. Toutes les combinaisons seront testées - Afficher les détails pertinents pour un item - Trier le résultat de la recherche. - Interface conviviale et intuitive. Facilité de navigation

Test unitaires, le panier

- Ajouter un item au panier : Quantité disponible, le montant s'affiche. Le montant total du panier s'affiche
- Modifier la quantité d'un item : Quantité disponible, le montant est mis à jour. Le montant total du panier est mis à jour
- Supprimer un item du panier : le montant total du panier se met à jour
- Ajouter un item au panier, quantité non disponible
- Modifier la quantité du panier, quantité non disponible
- Payer son panier : le joueur a le montant.
  o Le nombre de pièces d'or, d'argent, et de bronze est mis à jour
  o L'inventaire du joueur est mis à jour
  o Le magasin (la boutique) ou l'item est mis à jour.
  o Le panier est vidé.
- Payer son panier : le joueur n'a pas le montant : indiquer que le jouer n'a pas le solde pour payer.
- Acheter sans se connecter. --> erreur
- Acheter un sort si le joueur n'est pas mage. --> erreur

Test unitaires, l'inventaire du joueur
- Aucun critère : on affiche tous les items. - (optionnel) afficher selon un critère : Exemple Potion. Tous les critères seront testés - (optionnel) afficher selon des critères combinés

Les insertions : Exemple de procédure stockée pour ajouter une arme:

```
use dbsaliha;
drop procedure if exists ajouterArme;
delimiter |
create procedure ajouterArme(
    in pNom varchar(50),
    in pQuantite int,
    in pPrix int,
    in pPhoto varchar(100),
    in pDescription varchar(500),
    in pEfficacite varchar(30),
    in pGenreArme varchar(45))

begin
        declare pTypeItem char(1) default 'A';
        declare pidItem int;
                start transaction;
                        insert into Items (nom, quantiteStock, prix, photo,typeItem)
                        values ( pNom, pQuantite, pPrix, pPhoto, ptypeItem);

                        select LAST_INSERT_ID() into pidItem;

                        insert into Armes (idItem, description,efficacite, genre)
                        values (pidItem, pdescription,pEfficacite,  pGenreArme);
                commit;

end |
```

L'appel de la procédure se fait comme suit:

```
CALL ajouterArme('la hâche du daible',21,60,'hache.jpg','la hâche qui coupe ','super efficace','une main');
```

Dans MySQL Workbench :

```
use dbsaliha;
drop procedure if exists ajouterArme;
delimiter |
create procedure ajouterArme(
    in pNom varchar(50),
    in pQuantite int,
    in pPrix int,
    in pPhoto varchar(100),
    in pDescription varchar(500),
    in pEfficacite varchar(30),
    in pGenreArme varchar(45))

begin
    declare pTypeItem char(1) default 'A';
    declare pidItem int;
        start transaction;
            insert into Items (nom, quantiteStock, prix, photo,typeItem)
                values ( pNom, pQuantite, pPrix, pPhoto, ptypeItem);
                select LAST_INSERT_ID() into pidItem;
                insert into Armes (idItem, description,efficacite, genre)
                values (pidItem, pdescription,pEfficacite,  pGenreArme);
        commit;

end |

call ajouterArme('la nouvelle hache',21,60,'hache.jpg','hache hache ','super efficace','deuxmain');
```