Rapport Labo 2 AMT – Laurent Tailhades / Karel Ngueukam

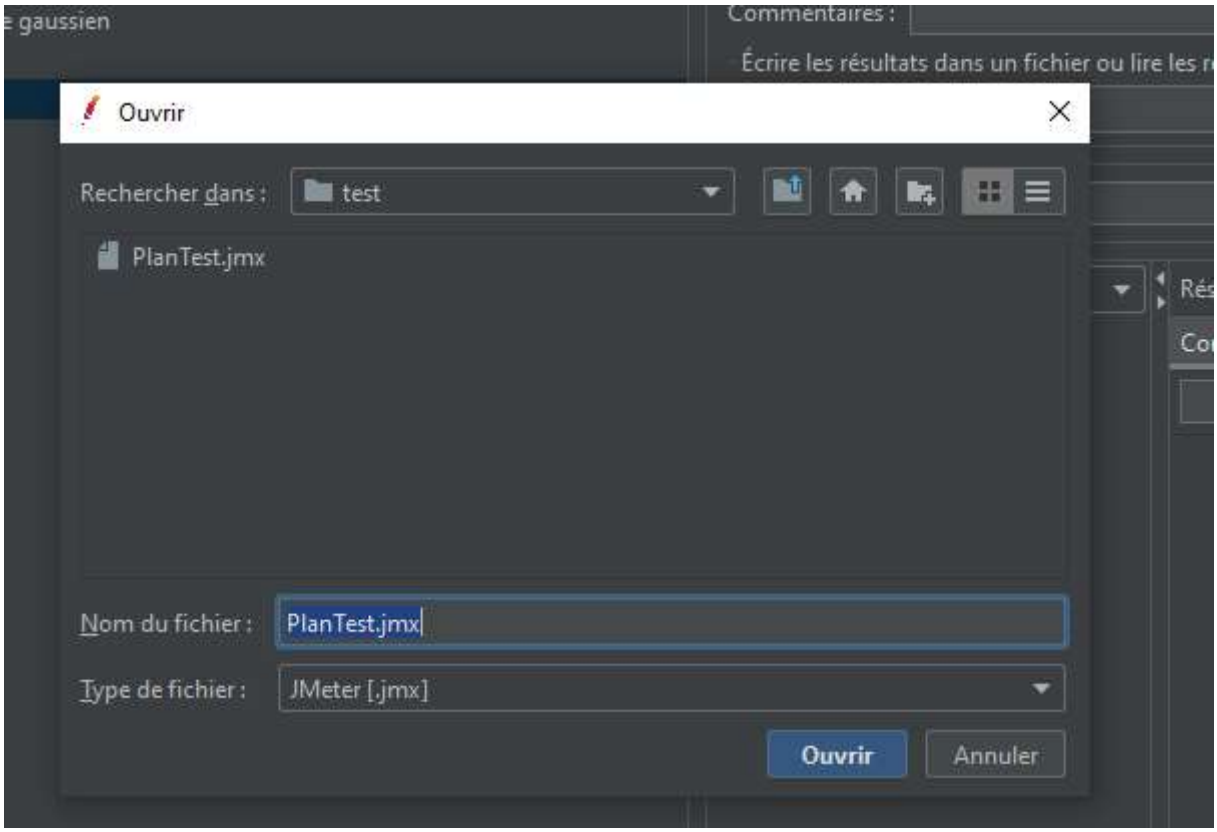
Introduction

Le laboratoire est partagé en 2 parties. L'une dédiée aux tests de performance, aux tests d'acceptation et aux tests de la gestion des transactions, et l'autre dédiée au mapping objet-relationnel.

Partie 1 : Test

Tests de performance avec JMeter

Pour utiliser JMeter, il suffit, dans un environnement Windows, de lancer ApacheJMeter.jar situé dans labo2.1/apache-jmeter-5.4.1/bin pour avoir un GUI de l'application.
Une fois lancé, il faudra ouvrir le plan de test « PlanTest » déjà configuré, dans le dossier /test

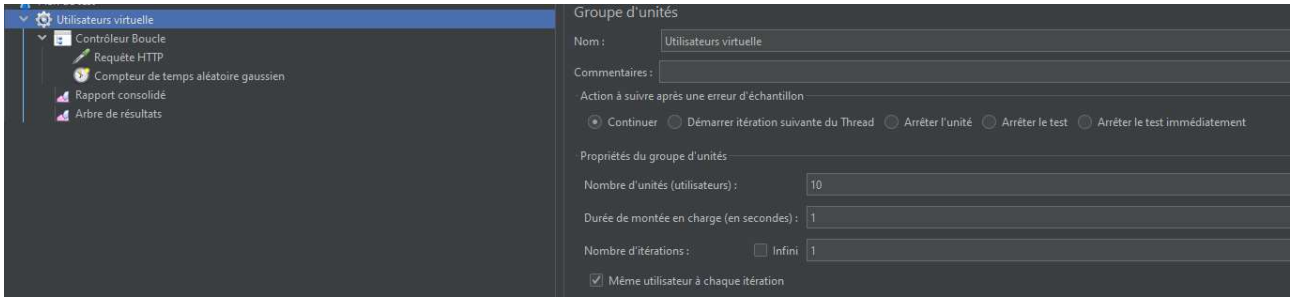


emplacement.jmx

Pour cette partie de test avec JMeter nous allons nous focaliser sur la partie affichage des stations avec la pagination. Nous allons tester avec un nombre différent d'utilisateurs, qui feront chacun une centaine de requêtes sur la même page.

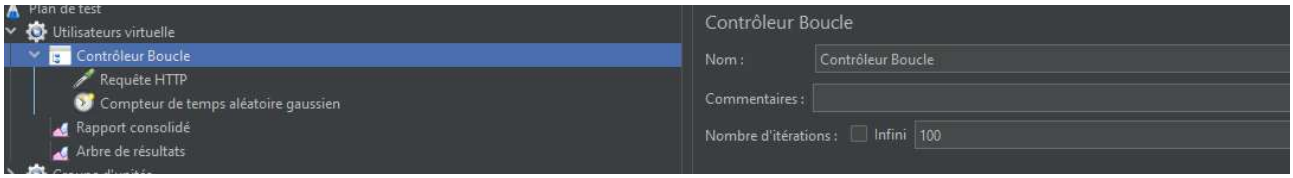
Nous voulions effectuer les tests sur la partie 2 avec les JPA mis en place, mais nous avons rencontré quelques problèmes qui nous ont obligé à effectuer les tests de JMeter sur la version 1 du labo.

Ici nous pouvons changer le nombre d’utilisateurs virtuel qui feront une requête en même temps



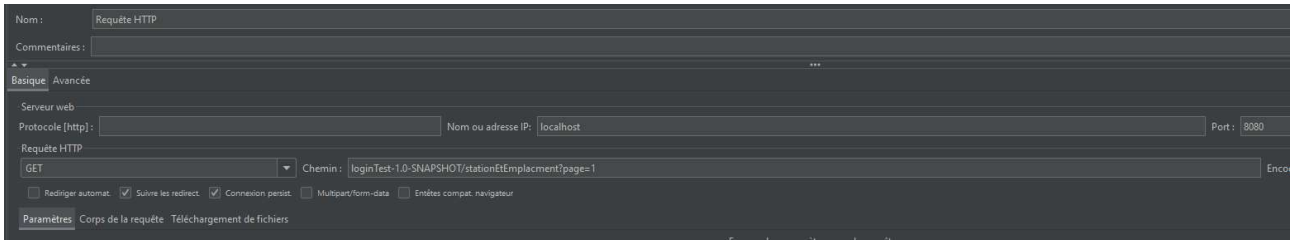
Jmeter utilisateurs virtuel

Nous avons un contrôleur de boucle qui va permettre de définir combien de requête fait chaque utilisateur, ici, il en fait 100.



Jmeter boucle

dans la requête HTTP, il y a le chemin que les utilisateurs virtuel vont essayer d’appeler, ici, la première page de la pagination des stations



Jmeter HTTP

Chaque utilisateur devrait avoir cet affichage :

Home || Station || Reserver une voiture || Rendre une Voiture || Admi

Station

Affichage des stations

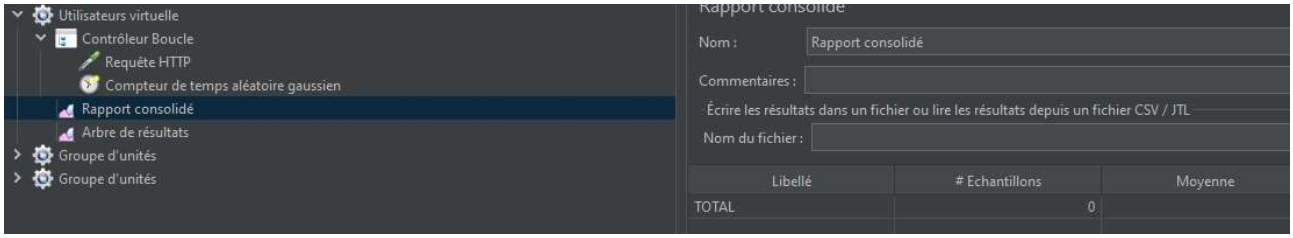
Emp ID	Emp Addr	Emp occupe	Emp reserve
1	Rue du lac 6	MOTO 1	Occupé
2	Rue du lac 6	BERLINE 2	Occupé
3	Rue du lac 6	BERLINE 3	Occupé
4	Rue du lac 6	MOTO 4	Occupé
5	Rue du lac 6	FOURGON 5	Occupé
6	Rue du lac 6		Libre
7	Rue du lac 6		Libre
8	Rue du lac 6		Libre
9	Rue du lac 6		Libre
10	Rue du lac 6		Libre

1 2 3 4 5 6 7 8

Next

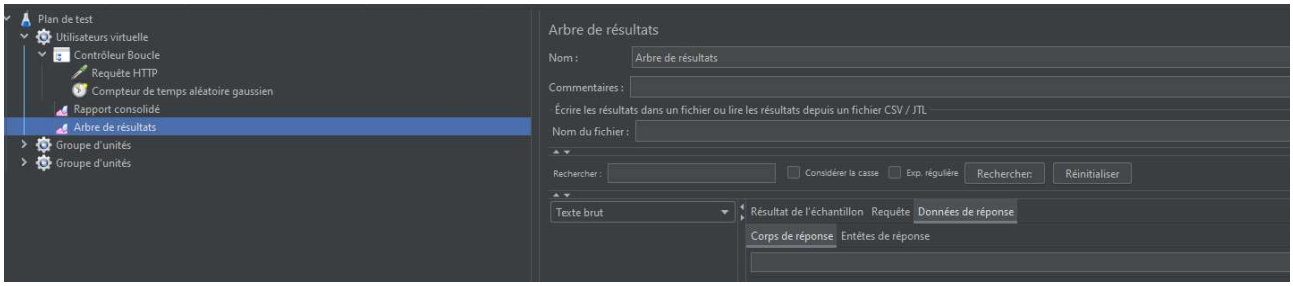
Screenshot 5 – Station pagination

Un listener, appelé un rapport consolidé, va permettre d'observer le nombre de requête effectuer, le taux d'erreur, ainsi que le temps de réponse moyen, minimum et max :



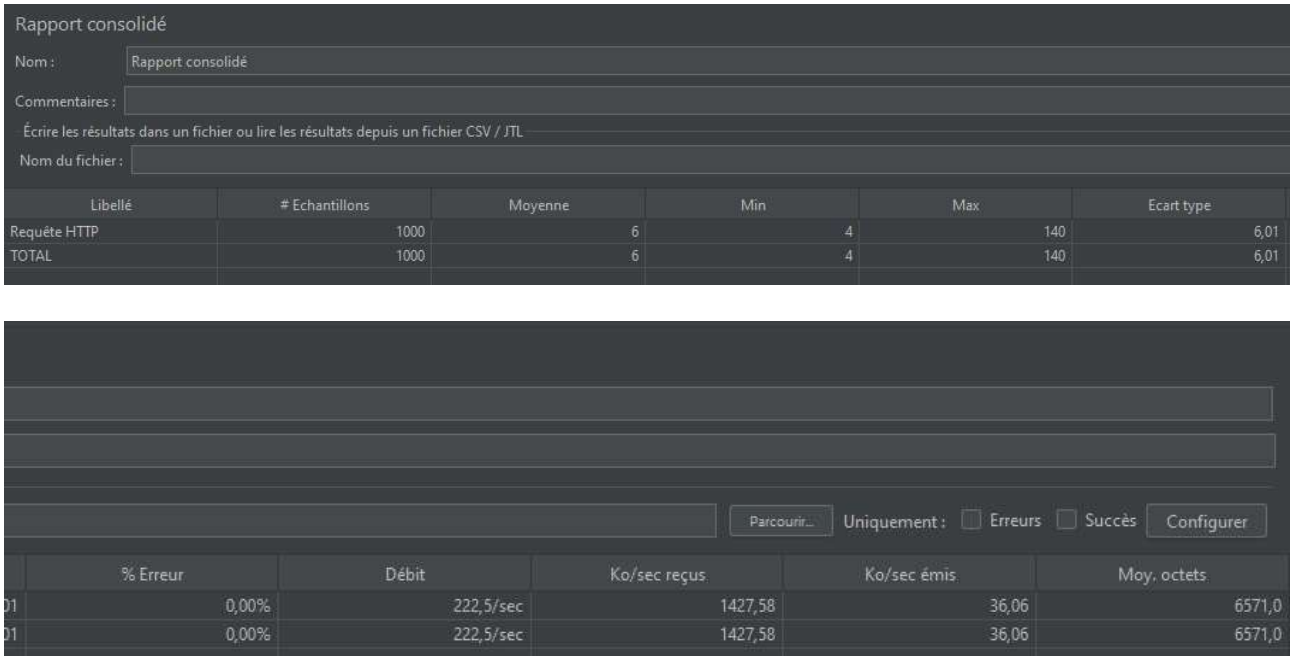
Rapport des résultats

Un deuxième listener, arbre de résultats, va permettre de suivre l'évolution de chaque, sa requête ainsi que sa réponse :



Arbre des résultats

Pour une pagination de 10 résultats par page, nous allons tester avec 10 utilisateurs qui font chacun 100 requêtes :


Rapport consolidé

Nom : Rapport consolidé

Commentaires :

Écrire les résultats dans un fichier ou lire les résultats depuis un fichier CSV / JTL

Nom du fichier :

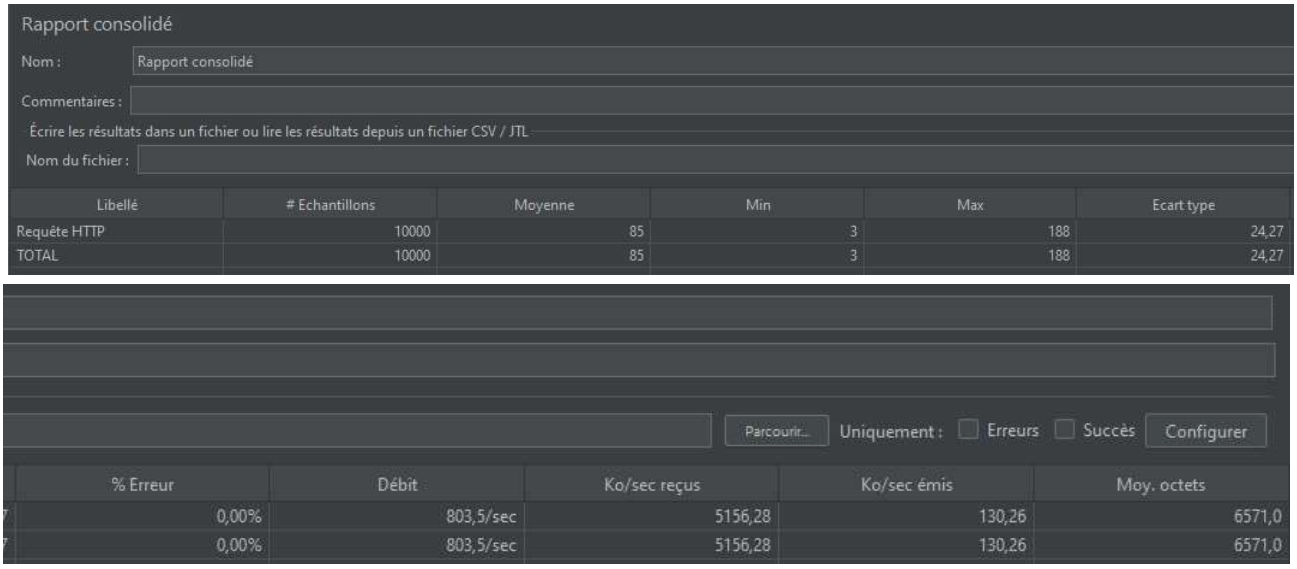
Libellé	# Echantillons	Moyenne	Min	Max	Ecart type
Requête HTTP	1000	6	4	140	6,01
TOTAL	1000	6	4	140	6,01

Parcourir... Uniquement : Erreurs Succès Configurer

	% Erreur	Débit	Ko/sec reçus	Ko/sec émis	Moy. octets
01	0,00%	222,5/sec	1427,58	36,06	6571,0
01	0,00%	222,5/sec	1427,58	36,06	6571,0

resultat 10 utilisateurs

Avec 100 utilisateurs :


Rapport consolidé

Nom : Rapport consolidé

Commentaires :

Écrire les résultats dans un fichier ou lire les résultats depuis un fichier CSV / JTL

Nom du fichier :

Libellé	# Echantillons	Moyenne	Min	Max	Ecart type
Requête HTTP	10000	85	3	188	24,27
TOTAL	10000	85	3	188	24,27

Parcourir... Uniquement : Erreurs Succès Configurer

	% Erreur	Débit	Ko/sec reçus	Ko/sec émis	Moy. octets
7	0,00%	803,5/sec	5156,28	130,26	6571,0
7	0,00%	803,5/sec	5156,28	130,26	6571,0

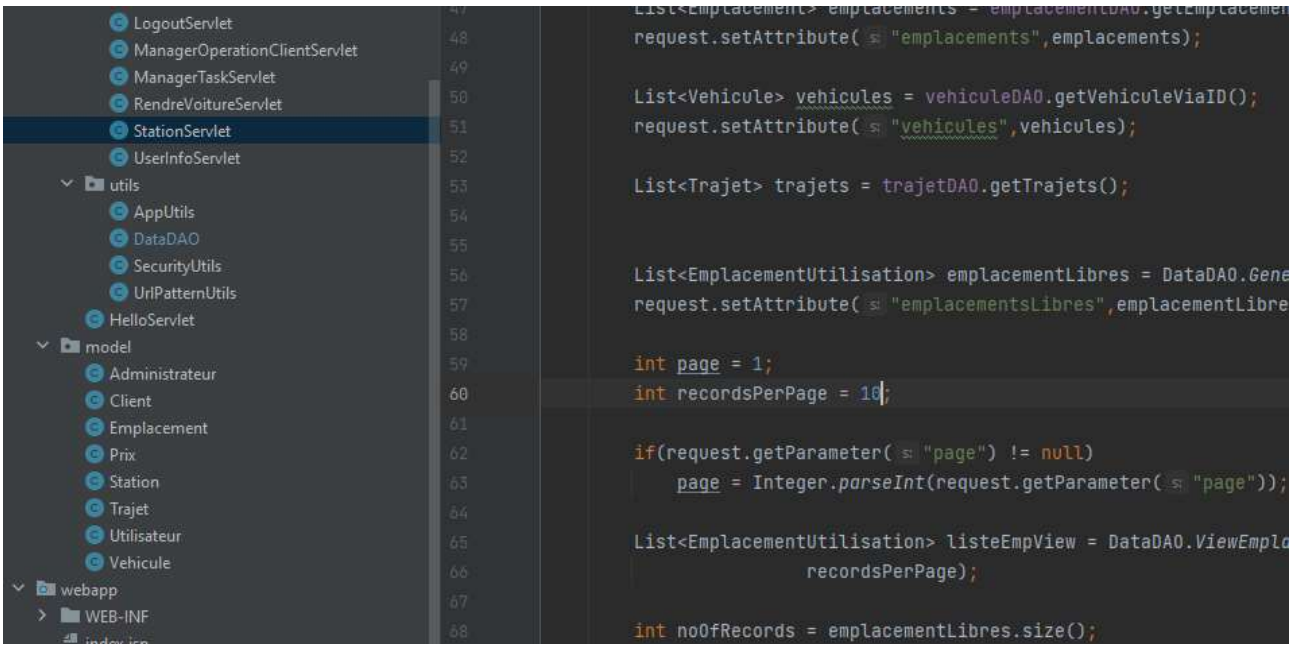
resultat 100 utilisateurs

On voit bien qu'il y a une différence de temps d'attente avec un plus grand nombre d'utilisateurs, nous n'avons pas eu d'erreur mais la performance diminue beaucoup.

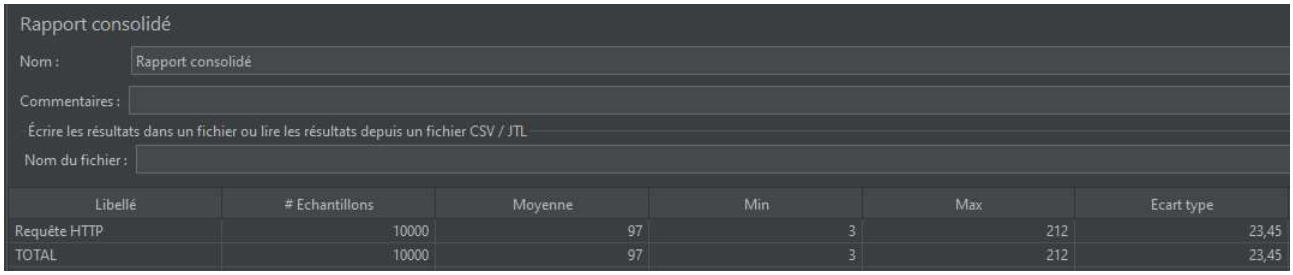
Nous allons maintenant essayer de voir s'il peut y avoir une différence avec le nombre de résultats afficher dans la pagination. De base les stations devraient être affichées sur 8 pages différentes, ici nous allons faire 2 tests différents, un avec 1 résultat par page, et un autre avec tous les résultats sur une page.

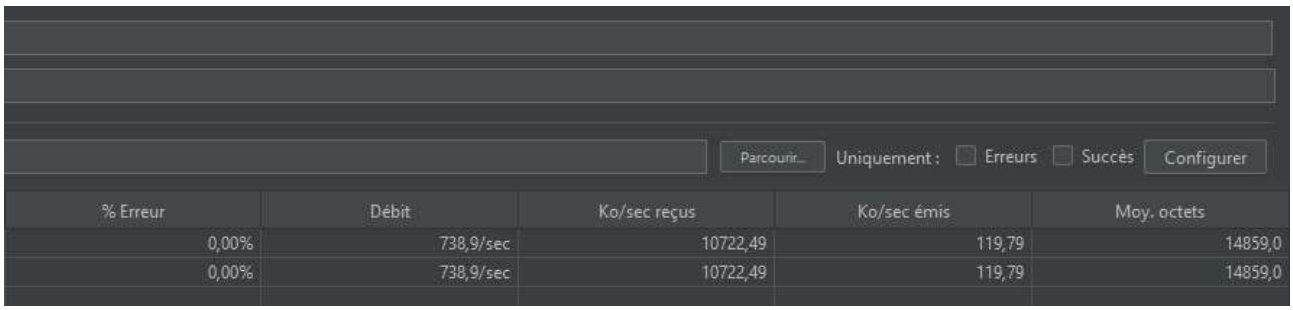
Nous ferons les tests avec uniquement 100 utilisateurs cette fois.

Pour changer les paramètres de la pagination, il faut changer l'attribut recordPerPage dans la classe StationServlet du package servlet et redéployer le serveur.

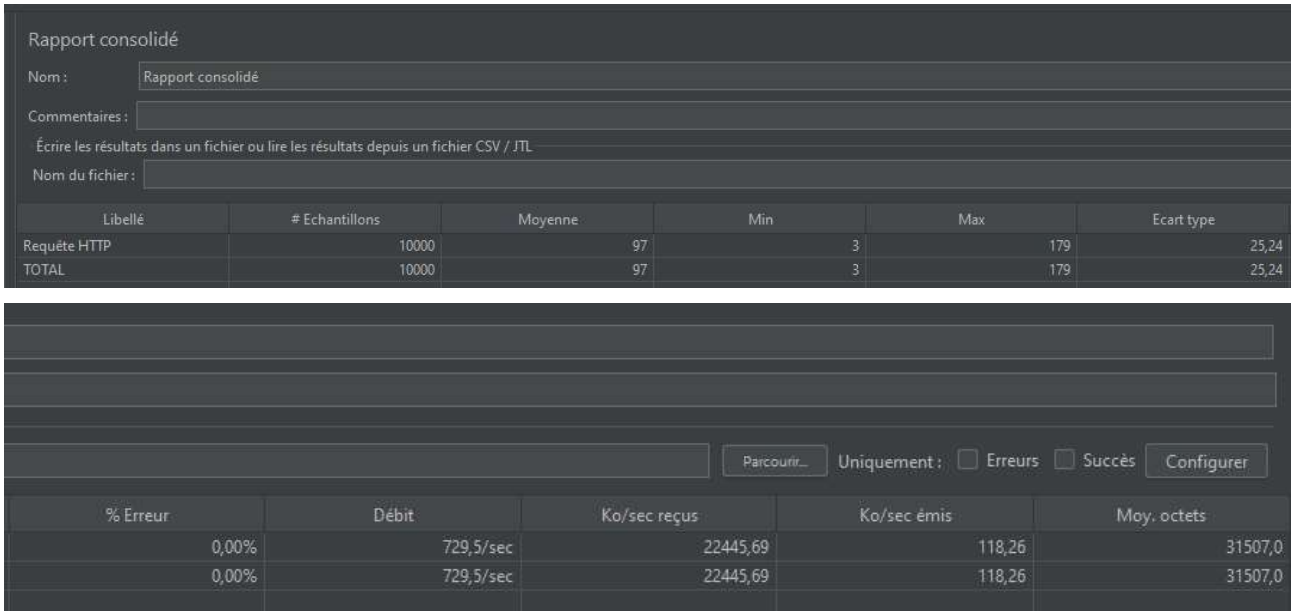


Avec 1 résultat par page :





Avec 100 résultat par page (soit le total)



C’est intéressant de voir que l’attente moyenne est équivalant pour les 2 cas, cependant, si il y a beaucoup plus de données à télécharger pour le cas avec 100 résultats, il y en a tout de même plus pour le cas d’un seul résultat à afficher par rapport à l’affichage de base, c’est sûrement dû au nombre important d’index à afficher pour la pagination, ici 76.

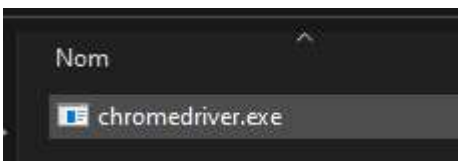
Il est donc important de faire des tests, et d’avoir un juste milieu pour contenter l’utilisateur et les ressource du serveur.

Test d’acceptation avec Selenium

Pour la partie test d’acceptation du projet nous devons utiliser selenium qui est un outil qui va travailler ici avec chrome pour naviguer dans les pages de notre projet et ainsi pouvoir vérifier le bon fonctionnement de celui-ci.

Utilisation et fonctionnement.

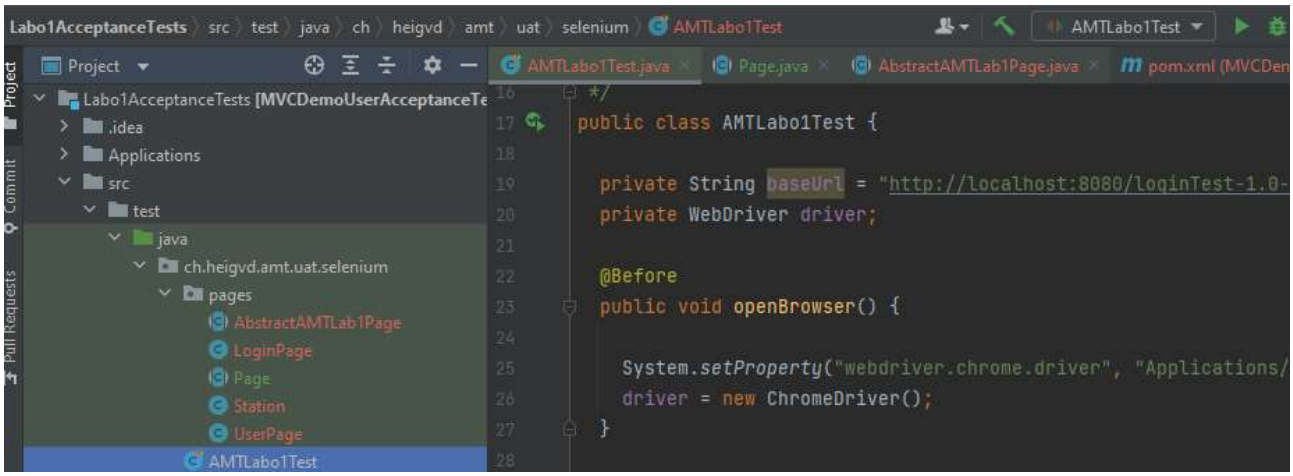
Pour son utilisation il faut utiliser un driver lié à notre environnement, ici Windows. Nous l’avons placé dans le dossier de test java dans le dossier Application



Il est appelé ici dans le code, dans la classe AMTLabo1Test.java

```
@Before
public void openBrowser() {

  System.setProperty("webdriver.chrome.driver", "Applications/chromedriver.exe");
  driver = new ChromeDriver();
}
```

Pour lancer les tests, il faut ouvrir le dossier LAbo1AcceptanceTests dans intelij, et avoir notre site web du labo1 qui tourne dans payara. Il suffit ensuite de lancer la classe AMTLabo1Test pour voir les tests s'effectuer dans notre navigateur chrome qu'il faut avoir installé au préalable.



Point 6 : la « page de login » n'est pas proposée si client connecté.

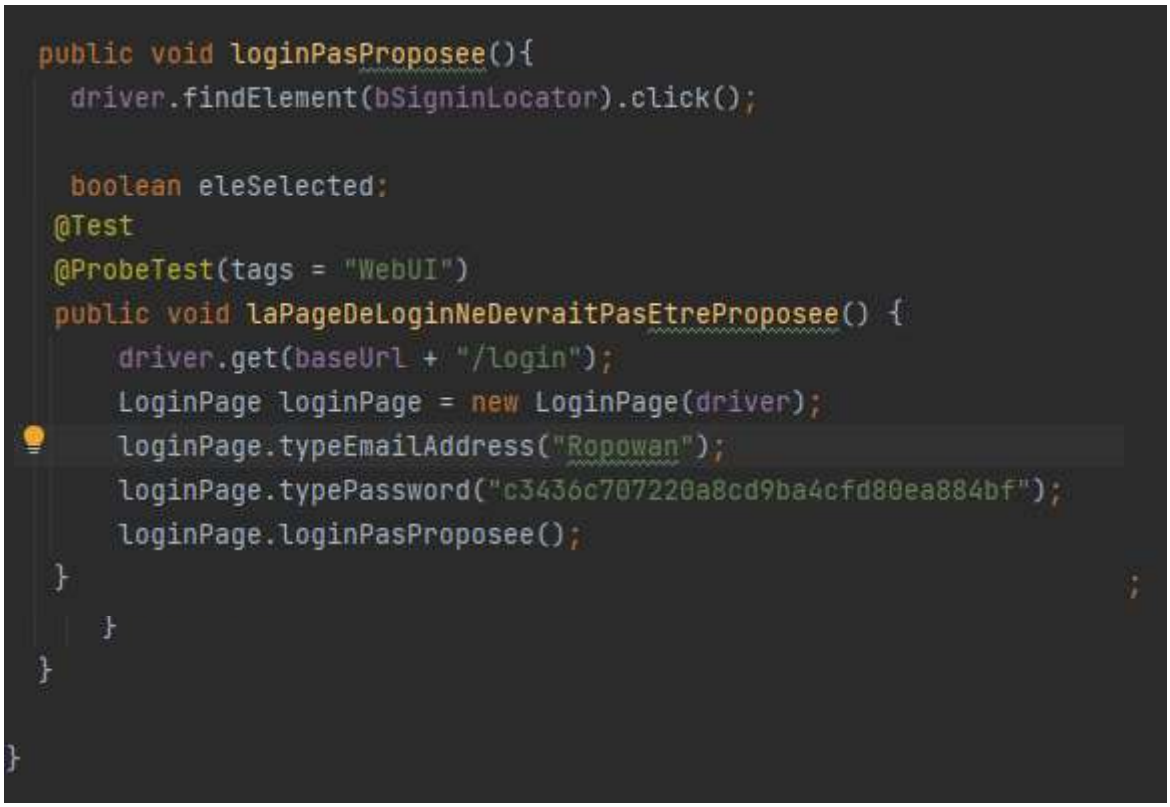
Comme demandé, nous avons testé la partie Login, où, lorsque l'on vient de se connecter, plus aucune page de login ne nous est proposé.

Pas encoré loggé, « Login » nous est encore proposé

« Login » doit disparaître

Le test renvoie une erreur si le bouton login est encore présent.

Les tests



```
public void loginPasProposee(){
  driver.findElement(bSigninLocator).click();

  boolean eleSelected;
@Test
@ProbeTest(tags = "WebUI")
public void laPageDeLoginNeDevraitPasEtreProposee() {
    driver.get(baseUrl + "/login");
    LoginPage loginPage = new LoginPage(driver);
    loginPage.typeEmailAddress("Ropowan");
    loginPage.typePassword("c3436c707220a8cd9ba4cfd80ea884bf");
    loginPage.loginPasProposee();
}                                                              ;
   }
}

}
```

passent bien avec ces identifiants, mais on peut imaginer qu'on change des données dans la table pour que les tests ne passent plus.
Aussi, si on n'a plus la page de login proposé, on peut toujours y accéder dans la barre du navitateurn avec «/login », à nous de voir si c'est voulu ou non.

Point 4 : « Affichage pour toutes les stations de tous les emplacements » en mettant en œuvre la pagination.

Pas besoin d'être logué pour ce point, il faut juste trouver un moyen de tester si l'affichage de la pagination marche correctement

Home || Station || Reserver une voiture || Rendre une Voiture || Administrateur || User Info || Login [ ]

Station

Affichage des stations

Emp ID	Emp Addr	Emp occupe	Emp reserve
1	Rue du lac 6	MOTO 1	Occupé
2	Rue du lac 6	BERLINE 2	Occupé
3	Rue du lac 6	BERLINE 3	Occupé
4	Rue du lac 6	MOTO 4	Occupé
5	Rue du lac 6	FOURGON 5	Occupé
6	Rue du lac 6		Libre
7	Rue du lac 6		Libre
8	Rue du lac 6		Libre
9	Rue du lac 6		Libre
10	Rue du lac 6		Libre

1 2 3 4 5 6 7 8

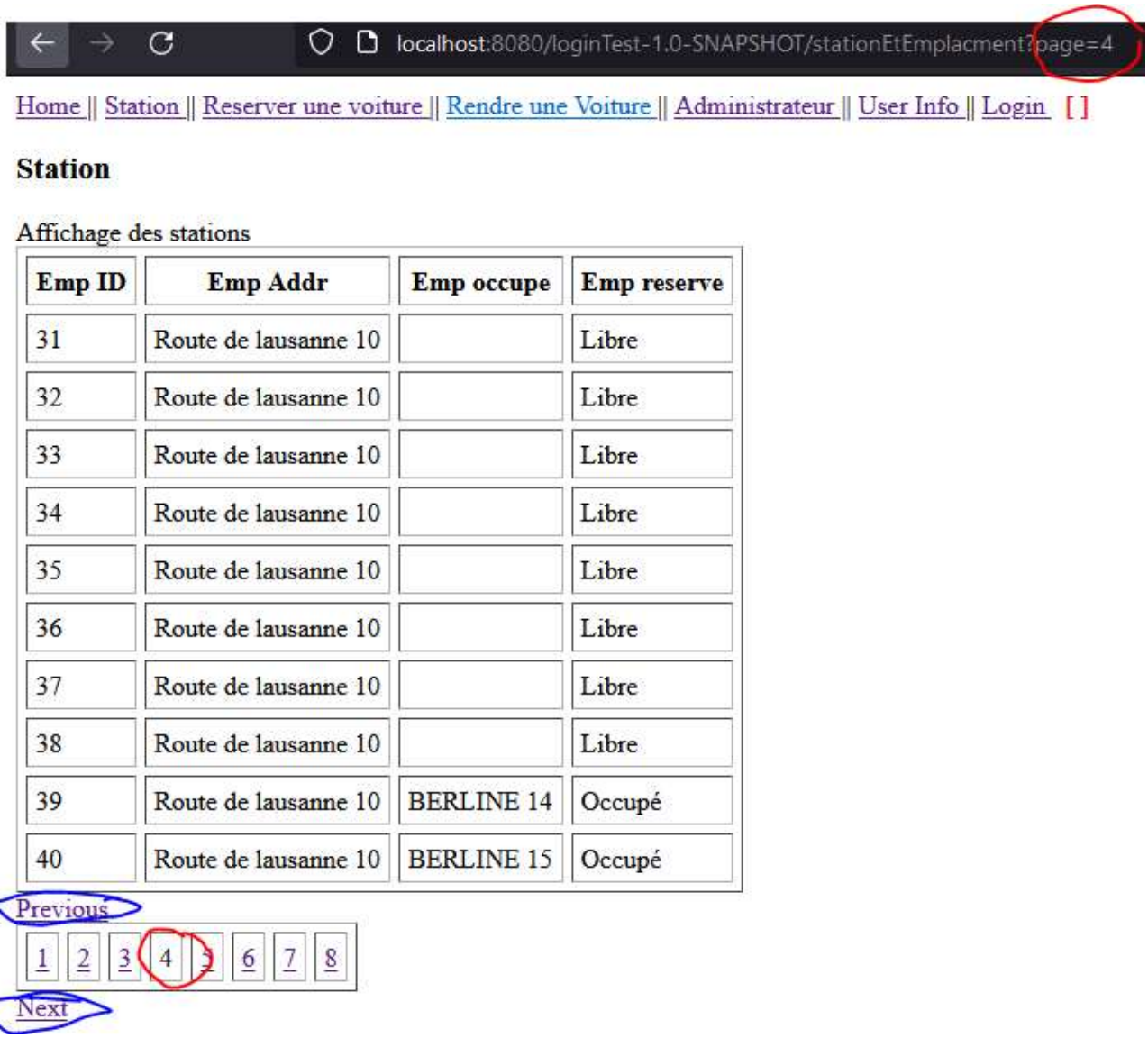
Next

Pour les tests, nous avons pensé à vérifier que le bouton next puisse permettre de naviguer jusqu'au bout, et donc d'afficher tous les emplacements, puis de tester dans le sens inverse et qu'on aille pas plus loin que les limites de la table. Un test permet de vérifier la barre d'adresse pour que le numéro de page corresponde au numéro de la pagination.

On vérifie aussi que les boutons de pages de la pagination existent, peut importent le nombre d'éléments affichés.

```java
public void testPagination(){

    //on vérifie qu'il y ait au moint un numéro de page
    if (driver.findElements(bNextLocator).isEmpty())  {
        throw new IllegalStateException("Pagination pas en marche");
    }

    int pagePaginationNumber = 1;

    //on defile les pagination vers la fin
    while(!driver.findElements(bNextLocator).isEmpty()){
        driver.findElement(bNextLocator).click();
        pagePaginationNumber++;

        String URL = driver.getCurrentUrl();
        // on recupere le numero de page de la pagination de l'url pour le comparer
        String page = URL.substring(URL.lastIndexOf( str: "=") + 1);

        if(Integer.parseInt(page) != pagePaginationNumber) {
            throw new IllegalStateException("Pas la bonne adresse de pagination");
        }
    }

    //on defile la pagination vers le début
    while(!driver.findElements(bPreviousLocator).isEmpty()){
        driver.findElement(bPreviousLocator).click();
        pagePaginationNumber--;

        String URL = driver.getCurrentUrl();
        // on recupere le numero de page de la pagination de l'url pour le comparer
        String page = URL.substring(URL.lastIndexOf( str: "=") + 1);

        if(Integer.parseInt(page) != pagePaginationNumber) {
            throw new IllegalStateException("Pas la bonne adresse de pagination");
        }
    }
}
```

Les tests passent, mais encore une fois, si la table est vide, ce qui pourrait être normal dans certains cas, les tests ne passeront plus



le numéro de page doit correspondre, les boutons Previous et Next sont utilisés jusqu'à ce qu'ils disparaissent respectivement.

Test de la gestion de transaction

Comme demandé, il n'y a pas eu de test à mettre en œuvre pour cette partie. Je me suis juste inspiré des exemples des webcast pour avoir un exemple similaire.

J'ai travaillé sur le labo 1 avec les JDBC. Plutôt que de travailler sur l'ajout de véhicule, j'ai choisi la partie administrateur qui ajouter un client. Cela affecte 2 tables, la table client et la table utilisateurs.

Au moment de l'ajout, un utilisateur est ajouté, puis un client. On va donc introduire un problème lors de l'ajout d'un utilisateur.

Pour commencer il a juste fallu faire quelques changements pour travailler avec un conteneur EJB

```java
@WebServlet("/managerTask")
public class ManagerTaskServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;


    @EJB
    private UtilisateurDAOLocal utilisateurDAO;
    @EJB
    private ClientDAOLocal clientDAO;
    @Inject
    private TrajetDAOLocal trajetDAO;
    @Inject
    private VehiculeDAOLocal vehiculeDAO;
    @Inject
    private AdministateurDAOLocal administrateurDAO;
```

EJB et non Stateless pour les 2 tables qui nous interessent

```java
@Stateless
@TransactionAttribute(TransactionAttributeType.REQUIRES_NEW)
public class UtilisateurDAO implements UtilisateurDAOLocal{
```

Ajout du

```java
@Stateless
@TransactionAttribute(TransactionAttributeType.REQUIRES_NEW)
public class ClientDAO implements ClientDAOLocal{
```

TransactionAttribute

```java
public void add(Utilisateur utilisateur) {
    try (
            Connection connection = dataSource.getConnection();
            PreparedStatement pstmt = connection.prepareStatement
                    ( sql: "insert into utilisateur (login, password) values (?,?)");) {

        pstmt.setString( parameterIndex: 1, utilisateur.getLogin());
        pstmt.setString( parameterIndex: 2, utilisateur.getPassword());
        pstmt.executeUpdate();
        //throw new RuntimeException("probleme client");
    } catch (SQLException e) {
        Logger.getLogger(UtilisateurDAO.class.getName()).log(Level.SEVERE,  msg: null, e);
    }
}
```

Ajout d’une erreur (ClientDAO.java)

```
try {
    utilisateurDAO.add(utilisateur);
    utilisateur = utilisateurDAO.getUtilisateur(newUserName);

} catch (Exception e) {
    errors.add("probleme avec l'ajout d'utilisateur");
    request.setAttribute( s: "errors", errors);
}
```

Lorsqu'on essaye d'ajouter un nouveau client, il y a donc un problème avec l'ajout d'utilisateur, et la gestion de transaction détecte maintenant le problème, ce qui ne va pas lui faire changer la table des clients

affichage de l'erreur

Sans cette gestion, la table d'utilisateur n'était pas modifiée, mais la table des clients si, à présent ce n'est plus le cas et la table des clients n'est plus affecté.