I) Préambule sur le déroulement du Quest AlloDocteur.

Le Quest AlloDocteur comporte **10 étapes** (avec 2 étapes **bonus 8 et 9** et une **étape 10** de projet final).
Il est indispensable d'avoir comme pré-requis les design patterns Singleton, Factory et DAO avant de commencer le Quest AlloDocteur sinon vous risquez de ne pas bien comprendre les differentes architectures du Quest.
Le but de ce Quest est de vous préparer à la mise en œuvre de l'application **AlloDocteur**.
L'application **AlloDocteur** sera utile pour les médecins qui travaillent seuls et qui sont dérangés fréquemment au cours d'une journée (car ils doivent prendre des rendez-vous téléphoniques) ce qui diminue leur disponibilité pour leurs patients. L'application AlloDocteur qui vous allez développer va permettre aux patients de prendre un rendez-vous chez un médecin donné. Le patient aura le droit de choisir entre plusieurs créneaux de 15 minutes entre 8h00 à 12h00, puis 14h00 à 19h00. Le patient devra toutefois lors de sa première connexion créer son compte avec son numéro de sécurité sociale et son email.

L'application AlloDocteur sera divisée en deux parties :

- Une partie **Front Office** qui va permettre aux patients de créer des comptes utilisateur, modifier ces comptes, prendre ou annuler des rendez-vous, gérer ses futurs rendez-vous, exporter ses futurs rendez-vous sous différents formats. L'application pourrait ressembler à http://elhadji-gaye.fr/alloDocteur/frontOffice (représente une maquette HTML possible de l'application. Vous n'êtes pas obligé de l'utiliser si elle ne vous plait pas vous créez votre propre maquette).

- Une partie **Back Office** qui va permettre aux médecins d'administrer l'application AlloDocteur (modifier les informations personnelles du médecin, parrainer d'autres médecins, vérifier la présence des patients à leur rendez-vous, voir les rendez-vous de la journée, rechercher des rendez-vous à une date particulière, confirmer la présence des patients à leur rendez-vous passés etc..). L'application pourrait ressembler à http://elhadji-gaye.fr/alloDocteur/backOffice (représente une maquette HTML possible de l'application. Vous n'êtes pas obliger de l'utiliser si elle ne vous plait pas vous créez votre propre maquette).

Nous vous précisons que le projet **Front Office** est obligatoire et que le projet **Back Office** est un projet bonus qui peut vous rapporter beaucoup de points donc nous vous conseillons fortement de le faire pour éviter toute surprise sur votre note finale. Il est tout à fait possible que dans un binôme l'un fasse le **Front Office** et l'autre le **Back Office** en utilisant le même projet Dao-Service (ici ce sera **maven-allo-docteur-dao-service**) que le binôme aura développé au préalable ensemble. Cette organisation serait peut être idéale pour une repartions des taches équilibrée entre binôme.

Le développement de l'application va necessiter 6 tables (**Utilisateur**, **Adresse**, **Creneau**, **Medecin**, **Patient** et **RendezVous**) en base de données.
Comme il sera peut-être difficile pour certains d'entre vous de développer l'application entière avec une bonne architecture alors on propose ce quest de **10 étapes** dans lequel vous allez développer une petite application avec deux tables (**Utilisateur** et **Adresse**) pour les étapes : 1 à 9 (les étapes 8 et 9 étant des étapes bonus). Nous allons ainsi voir plusieurs architectures possibles et vous serez libre de choisir l'architecture qui vous parrait le plus simple à mettre en œuvre pour AlloDocteur.
Nous développerons un **DAO Manuelle**, puis un **DAO JDBC**, puis un **DAO Hibernate** et enfin un **DAO Spring/Hibernate**.

Nous allons aussi coupler une application Web **Http Servlet** avec le **DAO JDBC**, puis avec un **DAO Spring/Hibernate**.

Il sera plus judicieux d'utiliser un **DAO Spring/Hibernate** pour votre projet à l'étape 9 mais si vous êtes dans l'incapacité de l'implementer correctement il vous sera permis d'utiliser le **DAO Hibernate** ou encore le **DAO JDBC**. Mais vous aurez 2 à 3 points de malus du fait que vous n'avez pas utilisé l'implémentation DAO optimale qui est ici un **DAO Spring/Hibernate**.

Pour l'application Web Java vous étes libre de rester sur du Http Servlet. Vous pouvez aussi choisir un FrameWork Web Java (Struts 1-2, JSF (Java Server Faces), Spring MVC, Spring Boot ect...

Vous pouvez aussi développer un Micro-Service pur Java (exemple un API Rest Json) qui sera consommé par un client de votre choix. Il est donc possible de choisir un client Angular 6-7, un client React Js, un client Node Js, un client C#/Asp.Net, un client PHP mais **je reprécise il faudra que le Micro-Service soit à 100 % Java**.

Pour réaliser le microservice Java les **étapes 8 et 9** vont vous aider de manière décisive donc n'hésiter pas à réaliser ces étapes et les soumettre à validation.
Il sera plus facile pour vous de réaliser les étapes du Quest dans l'ordre.

La logique d'enchainement des étapes est la suivante :

- Pour commencer les DAO il est necessaire d'avoir des bases solides sur les Design Pattern et sur la reflexivité d'où l'**étape 1**.
- Pour manipuler des sources de données il est nécessaire de maitriser les **List**, **Array** et **Map** d'où l'**étape 2**.
- Pour comprendre l'accées à la base de données avec des FrameWork JPA comme Hibernate il est nécessaire de comprendre la base de comunication d'une base de données avec Java d'où l'**étape 3** avec du JDBC (Java DataBase Connectivity).
- Pour créer une application Web il est necessaire de comprendre les notions de base du **Http Servlet** d'où l'**étape 4**.
- Pour comprendre les ORM JPA tel que **Hibernate** il est nécessaire de les utiliser sans Spring dans un premier temps d'où l'**étape 5**.
- Une fois qu'on a compris le fonctionnement d'**Hibernate** tout seul il est facile de l'integrer à Spring d'où l'**étape 6**.
- Une fois qu'on a compris le fonctionnement de **Spring/Hibernate** dans une application console, alors il devient facile de l'integrer à une application Web d'où l'**étape 7**.
- Pour ceux qui veulent créer une application **Micro-Service Java** avec un client donnée (Angular 6-7, React Js, Asp.Net, PHP ect…), les **étapes 8** et **9** vous permettront de le réaliser facilement.
- Pour réaliser l'application AlloDocteur dans les meilleures conditions il est indispensable de valider les étapes : 1, 2, 3, 4, 5, 6 et 7.

La validation des étapes : 1, 2, 3, 4, 5, 6 et 7 va vous permettre :

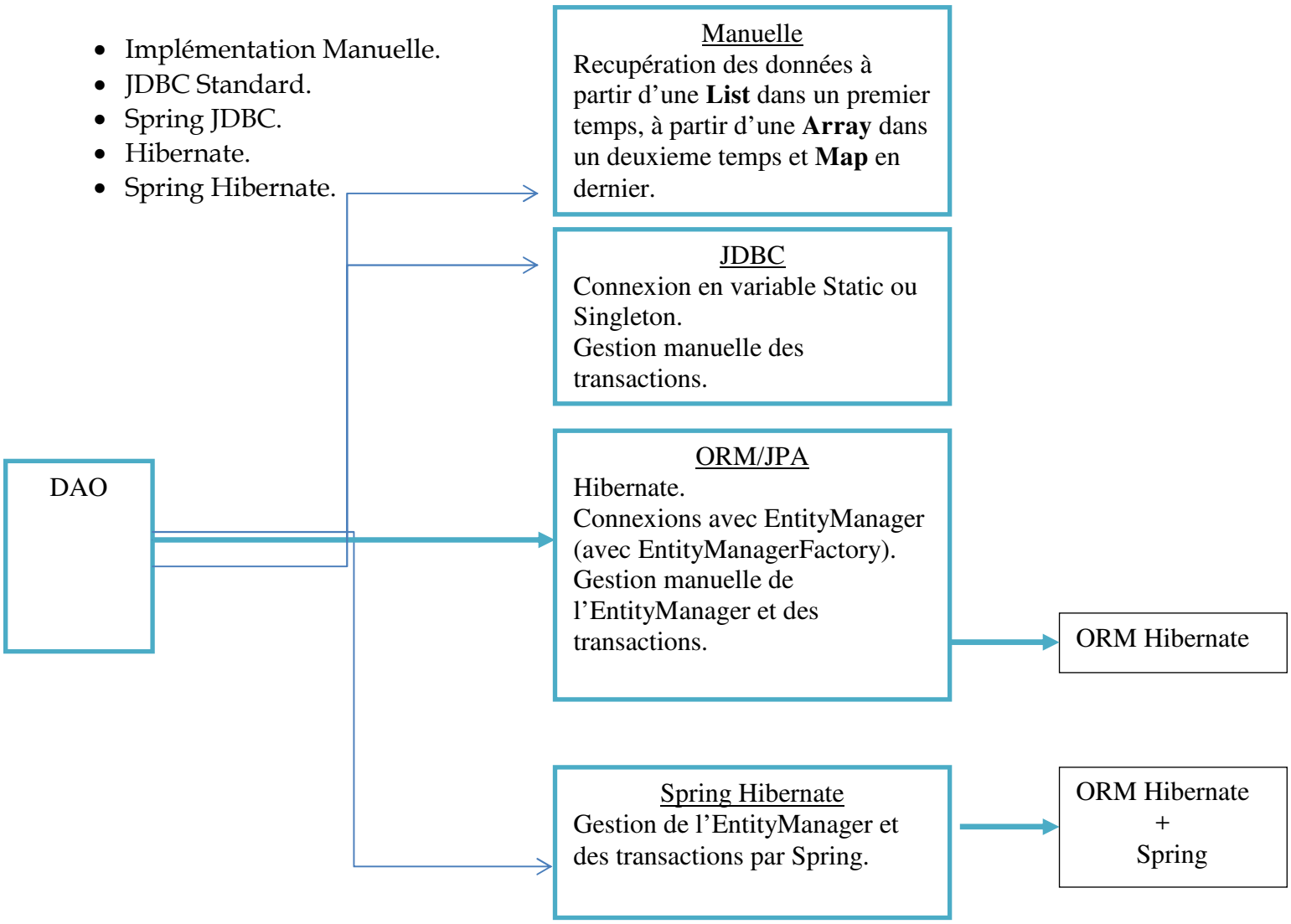
- D'une part de gagner des points précieux.
- D'autre part de faire le projet **AlloDocteur** à l'étape 10 en toute sérenité avec les tables **Utilisateur** et **Adresse** en moins car vous les aurez déjà implémentées au cours Quest.

Vous devrez obligatoirement repondre à chaque étape à **deux à trois questions sur Java**. Vous mettrez à chaque fois les réponses dans un fichier **readMe.txt** à la racine de votre projet.
Je vous rappelle que même si le Quest se fait en binôme il est important que vous le fassiez ensemble pour avoir le même niveau en Java, Ceci va vous permettre d'etre plus éfficace dans le projet finale à l'étape 10. Chaque membre du groupe pourais par exemple faire son Quest individuellement et quand le binôme aura terminé il fera un point pour savoir ce qu'il doit commiter au pas dans le repos SVN. Je vous conseille cette méthode de travail qui a déjà portée ses fruits dans le passé.
De plus c'est uniquement de cette manière que chaque élément d'un binôme pourra répondre correctement aux **questions techniques qui seront individuellement posées lors de la soutenance finale**. La réponse à ces questions sera bien entendu notée.

Je reste à votre disposition pour des questions supplémentaires sur l'organisation du Quest.

Nous allons donc voir plusieurs implémentations de DAO à travers les bases de données MYSQL « **base_quest_allo_docteur** ».

- Implémentation Manuelle.
- JDBC Standard.
- Spring JDBC.
- Hibernate.
- Spring Hibernate.

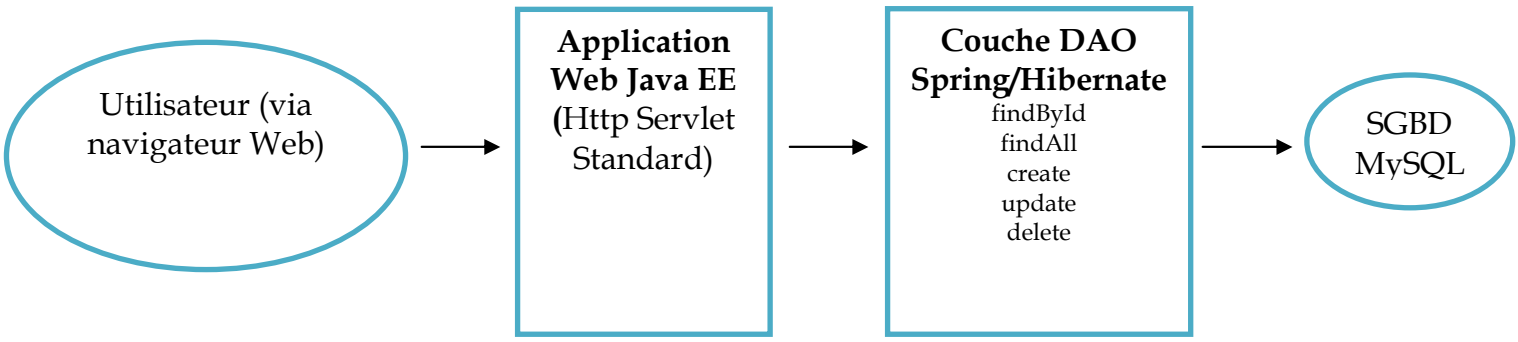


Notre objectif finale est l'implémentation **Spring Hibernate** mais il nécessaire de passer par les étapes intermédiaires afin de bien assimiler tous ces concepts.
Nous allons à chaque implémentation créée les interfaces **com.cours.allo.docteur.dao.IUtilisateurDao** et **com.cours.allo.docteur.dao.IAdresseDao** avec toutes les méthodes ci dessous. Nous créerons aussi leur classe d'implémentations **com.cours.allo.docteur.dao.impl.UtilisateurDao** et **com.cours.allo.docteur.dao.impl.AdresseDao**.
Puis **com.cours.allo.docteur.service.IServiceFacade** et **com.cours.allo.docteur.service.impl.ServiceFacade** qui vont faire appel aux interfaces **IUtilisateurDao** et **IAdresseDao**.
La classe **com.cours.allo.docteur.utils.Constants** nous servira à garder les constantes de l'application.
Vous utiliserez obligatoirement la classe **com.cours.allo.docteur.exception.CustomException** pour traiter toutes les Exceptions des couches **DAO** et **Service**.

public List<Utilisateur> findAllUtilisateurs()
public Utilisateur findUtilisateurById(int idUtilisateur)
public List<Utilisateur> findUtilisateursByIdentifiant(String identifiant)
public List<Utilisateur> findUtilisateursByPrenom(String prenom)
public List<Utilisateur> findUtilisateursByNom(String nom)
public List<Utilisateur> findUtilisateursByCodePostal(String codePostal)
public Utilisateur createUtilisateur(Utilisateur user)
public Utilisateur updateUtilisateur(Utilisateur user)
public boolean deleteUtilisateur(Utilisateur user)

public List<Adresse> findAllAdresses()
public Adresse findAdresseById(int idAdresse)
public List<Adresse> findAdressesByVille(String ville)
public List<Adresse> findAdressesByCodePostal(String codePostal)
public Adresse createAdresse(Adresse adresse)
public Adresse updateAdresse(Adresse adresse)
public boolean deleteAdresse(Adresse adresse)

Vous aurez à chaque étape **2 à 3 questions** à répondre. Les réponses se feront dans un fichier **readMe.txt** qui se trouve dans le package par défaut de votre projet.
Nous allons par la suite coupler notre DAO **Spring Hibernate** à un projet Web **Java EE** de type Http Servlet Standard puis un projet Web Rest Spring MVC puis un projet Web Rest Spring Boot.



Après avoir joué le script SQL « script_base_quest_allo_docteur.sql » Il est impératif que votre base de données s'appelle base_quest_allo_docteur avec l'utilisateur application, le mot de passe passw0rd et le port 3306. Au cas contraire la validation de votre étape sera refusée.

Il faudra aussi respecter la nomenclature des projets et ne pas les modifier.Dans certains étapes vous allez partir d'un fichier de tes JUnit avec un certains nombres d'élements vous pourez faire des ajouts mais en aucun cas le modifier intégralement afin que votre test JUnit passe. C'est à vous de vous integrer à mes tests unitaires JUnit. Je vous ais mis toutes les librairies

necessaire dans vos pom.xml donc normalement vous ne devriez avoir besoins de le modifier.

II) Installation de NetBeans 8.0.2.

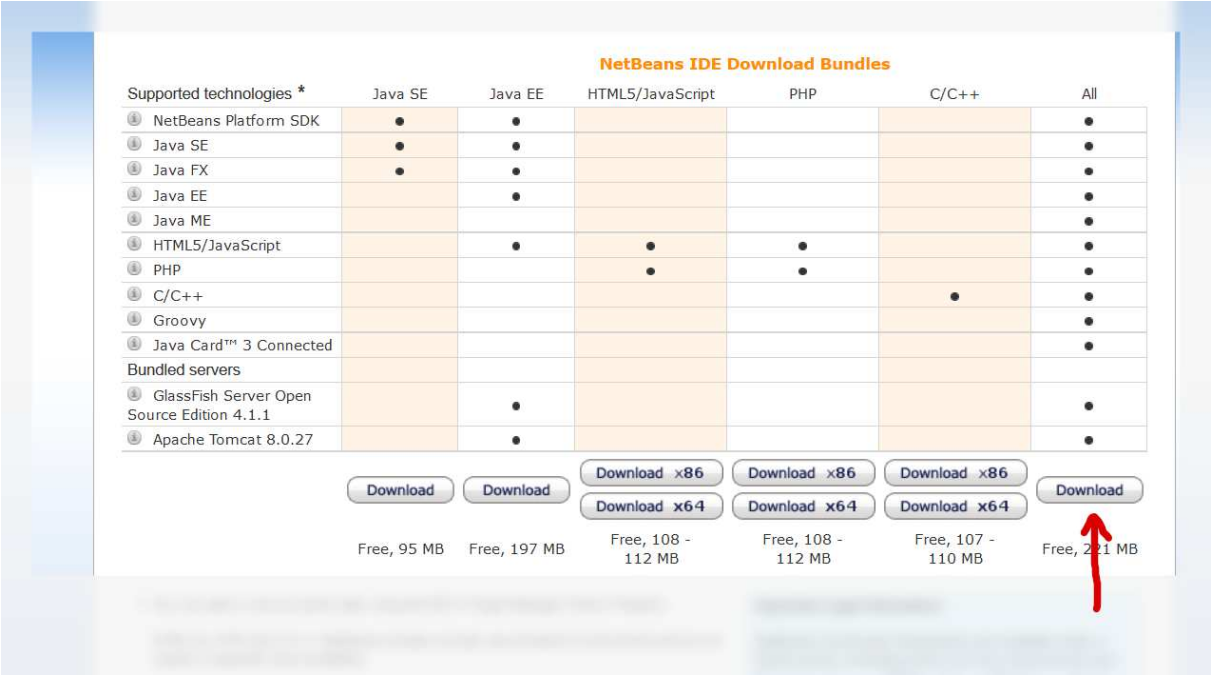
- Télécharger le **Pack JDK 8 si vous ne l'avez pas**

http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

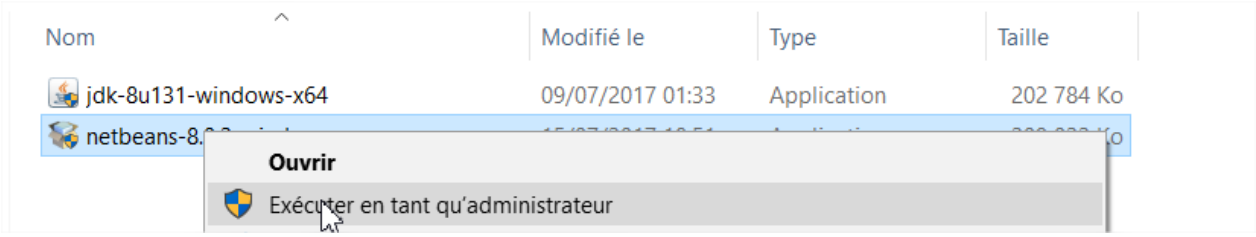
- Télécharger le **NetBeans 8.0.2 (à ne pas confondre avec la version 8.2):**

https://netbeans.org/downloads/8.0.2/

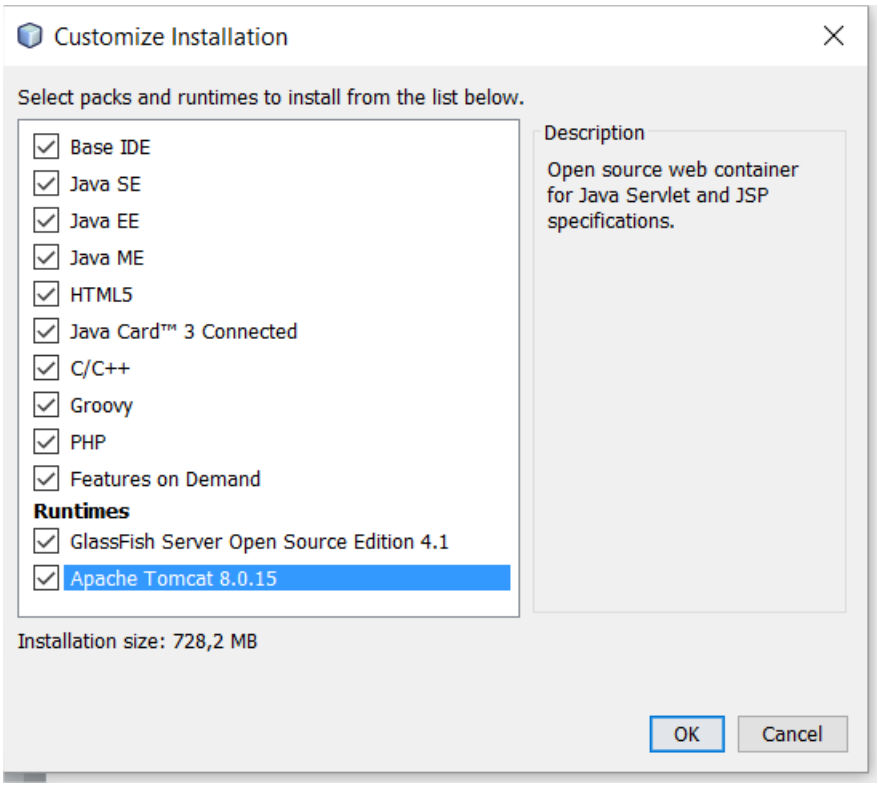
Prendre la version Full avec GlassFish installé dessus.



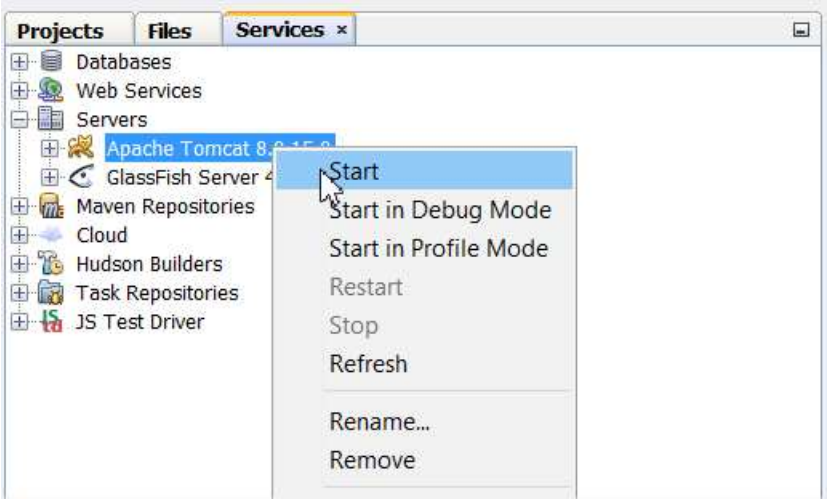
Installer **NetBeans 8.0.2** comme Administrateur.

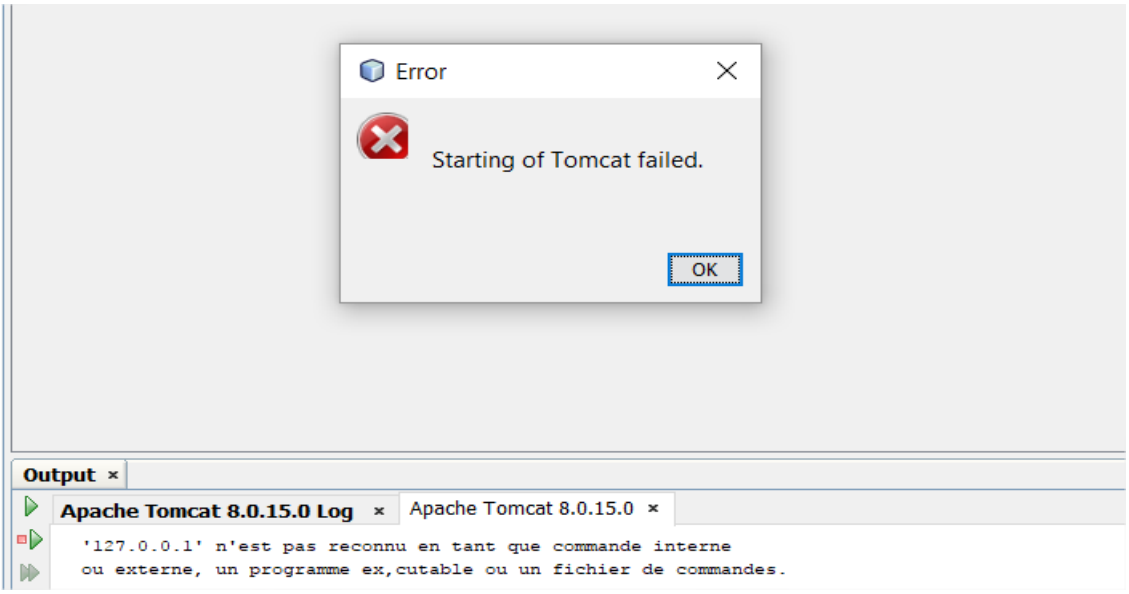


Cliquer **Customize** pour sélectionner **Tomcat 8.0.15** pour l'installation puis OK puis Next.
Si vous voulez utiliser le serveur d'application par défaut de Java c'est-à-dire **GlassFish** alors vous pouvez de ne pas installer Tomcat et dans ce cas là nulle besoin d'éxécuter le reste procedure ci-dessous en ce qui concerne l'installation de **NetBeans 8.0.2**.


Customize Installation
Select packs and runtimes to install from the list below.
Base IDE
Java SE
Java EE
Java ME
HTML5
Java Card™ 3 Connected
C/C++
Groovy
PHP
Features on Demand
Runtimes
GlassFish Server Open Source Edition 4.1
Apache Tomcat 8.0.15
Description
Open source web container for Java Servlet and JSP specifications.
Installation size: 728,2 MB
OK
Cancel


Projects
Files
Services
Databases
Web Services
Servers
Apache Tomcat 8.0.15.0
GlassFish Server 4.1
Maven Repositories
Cloud
Hudson Builders
Task Repositories
JS Test Driver

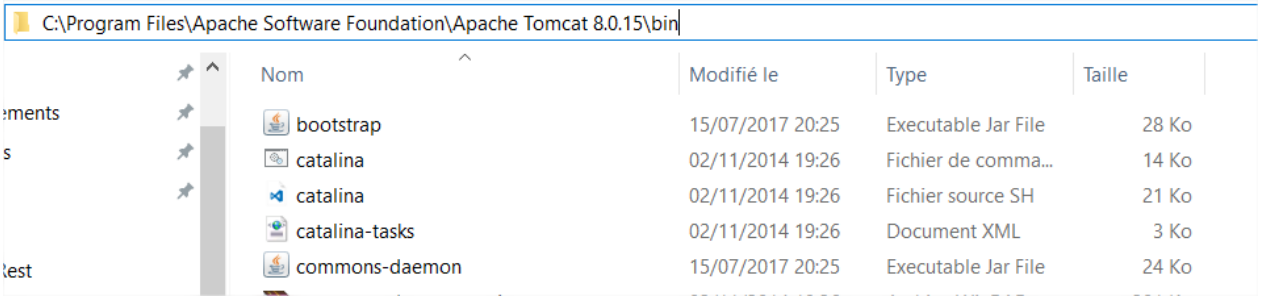




La console de Tomcat affiche l'erreur suivante :

'127.0.0.1' n'est pas reconnu en tant que commande interne

Pour résoudre ce problème il faudra modifier le fichier **catalina.bat** dans **C:\Program Files\Apache Software Foundation\Apache Tomcat 8.0.15\bin**



Rechercher dans ce fichier **JAVA_OPTS**.
Regardons de plus près les lignes **196** et **201** de **catalina.bat**.

```
190
191 if not "%LOGGING_CONFIG%" == "" goto noJuliConfig
192 set LOGGING_CONFIG=-Dnop
193 if not exist "%CATALINA_BASE%\conf\logging.properties" goto noJuliConfig
194 set LOGGING_CONFIG=-Djava.util.logging.config.file="%CATALINA_BASE%\conf\logging.properties"
195 :noJuliConfig
196 set "JAVA_OPTS=%JAVA_OPTS% %LOGGING_CONFIG%"
197
198 if not "%LOGGING_MANAGER%" == "" goto noJuliManager
199 set LOGGING_MANAGER=-Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager
200 :noJuliManager
201 set "JAVA_OPTS=%JAVA_OPTS% %LOGGING_MANAGER%"
```

Remplacer :

set "JAVA_OPTS=%JAVA_OPTS% %LOGGING_CONFIG%"

Par

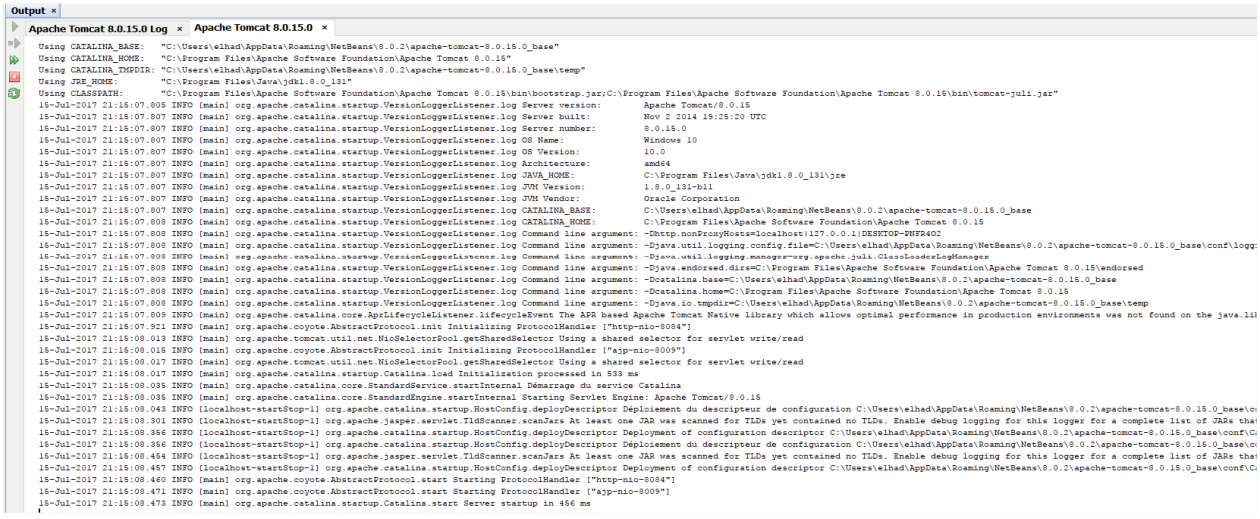
set JAVA_OPTS=%JAVA_OPTS% %LOGGING_CONFIG%

Remplacer :

set "JAVA_OPTS=%JAVA_OPTS% %LOGGING_MANAGER%"

Par

set JAVA_OPTS=%JAVA_OPTS% %LOGGING_MANAGER%



21:15:08.473 INFO [main] org.apache.catalina.startup.Catalina.start Server startup in 456 ms
Après ces modifications le serveur d'application **Tomcat** démarre en quelques millisecondes.

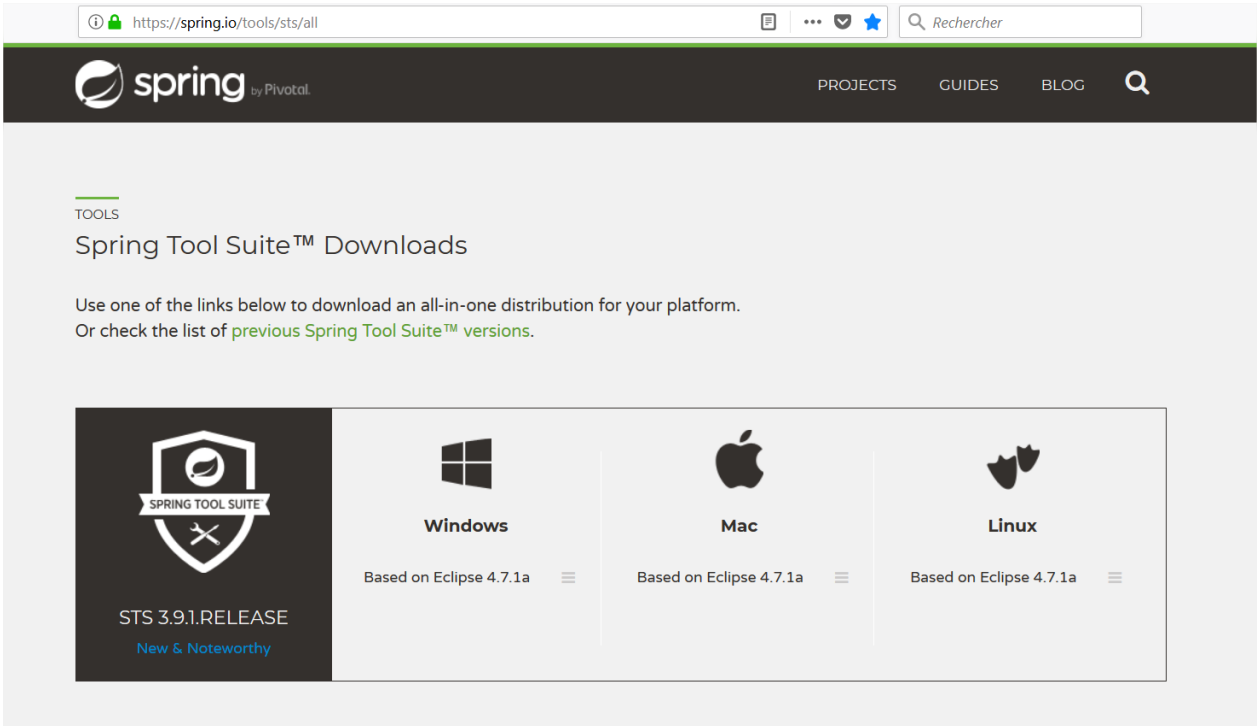
III) Installation de Spring Tool Suite.

- Télécharger le **JDK 8 si vous ne l'avez pas**

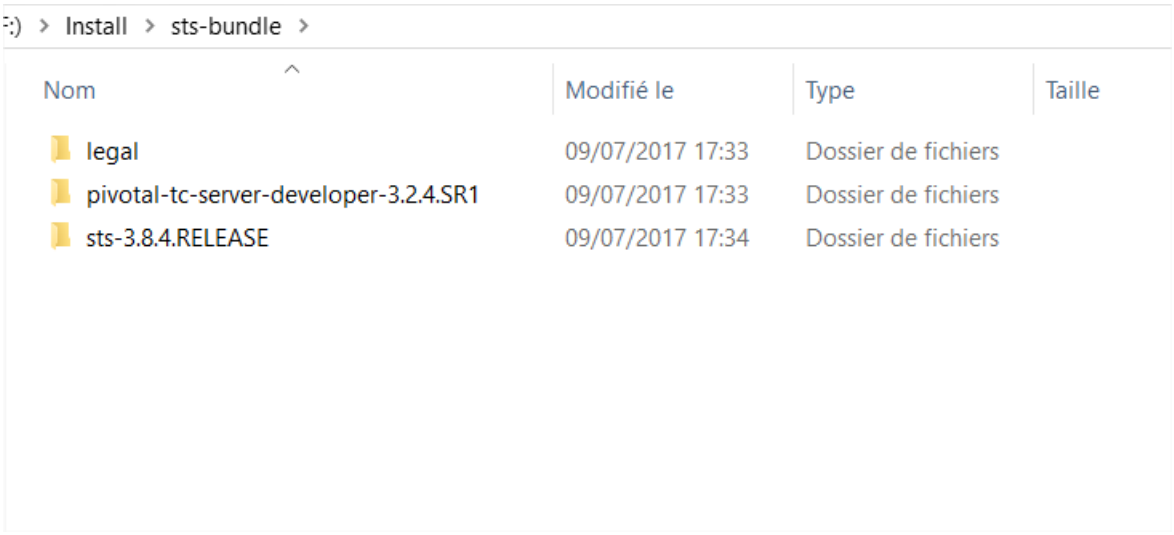
http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

- Télécharger **Spring Tool Suite**

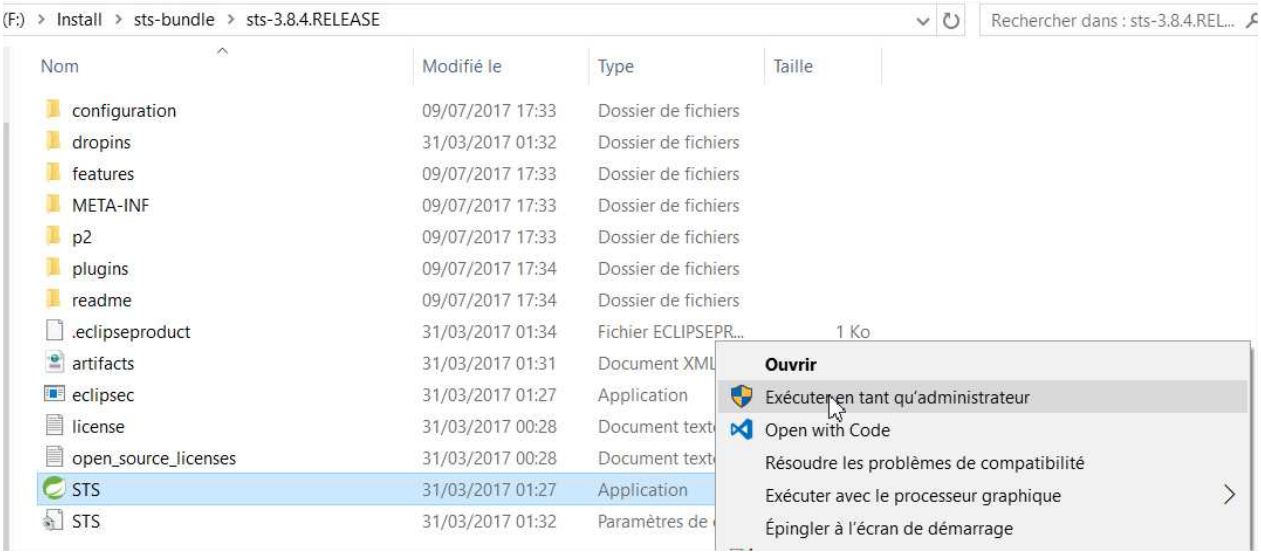
https://spring.io/tools/sts/all



Une fois que vous avez fini de telecharger le fichier, le mettre dans le repertoire que vous voulez.



Allez dans le repertoire **sts-bundle\sts-3.8.4.RELEASE** puis lancer l'éxecutable en tant que Administrateur.



IV) Initialisation de la base de données MYSQL.

1. Telechargement de Wamp ou MYSQL avec la version 5.

Télécharger Wamp Serveur : http://www.wampserver.com/

Ou

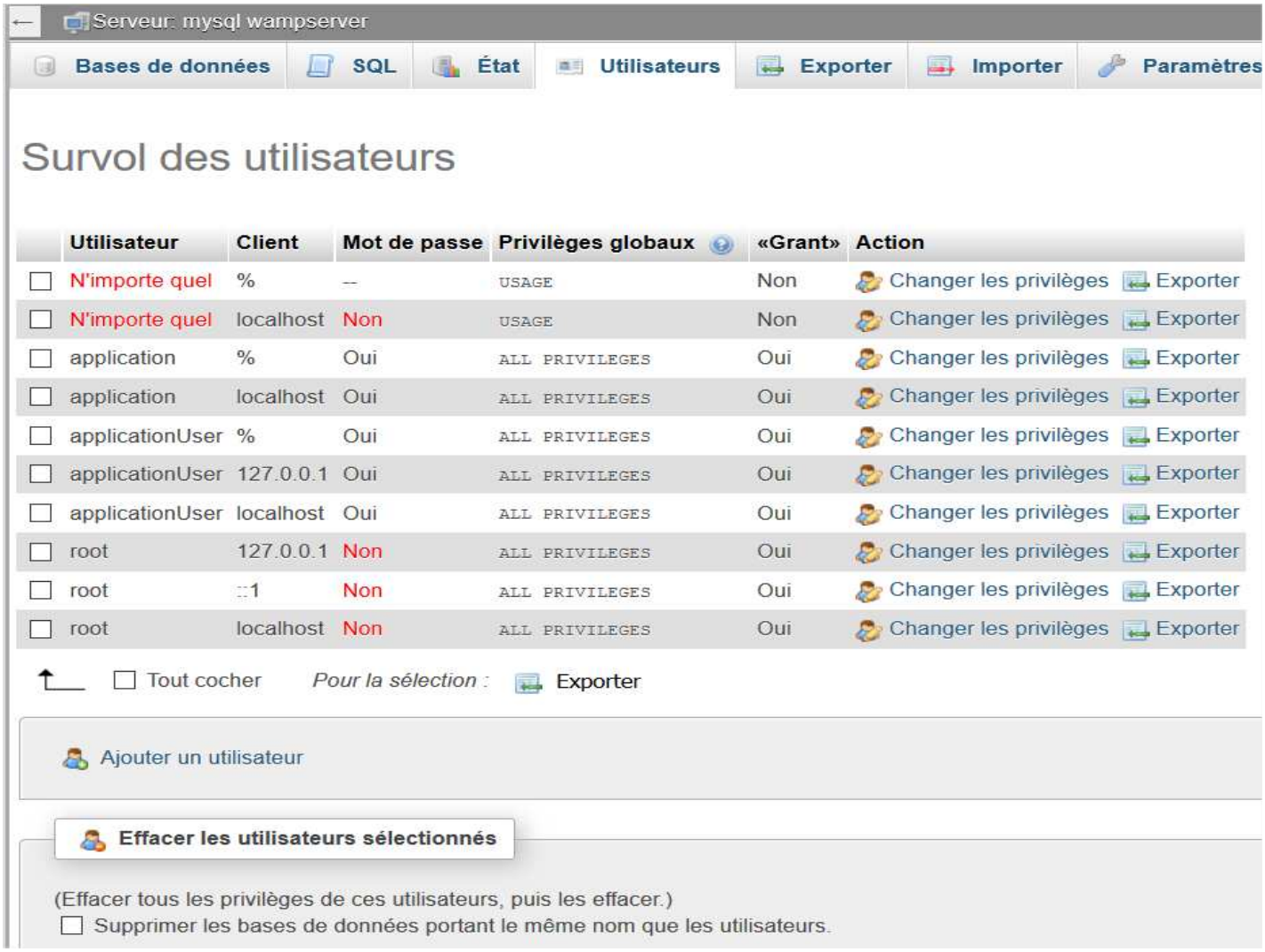
MySQL http://dev.mysql.com/downloads/

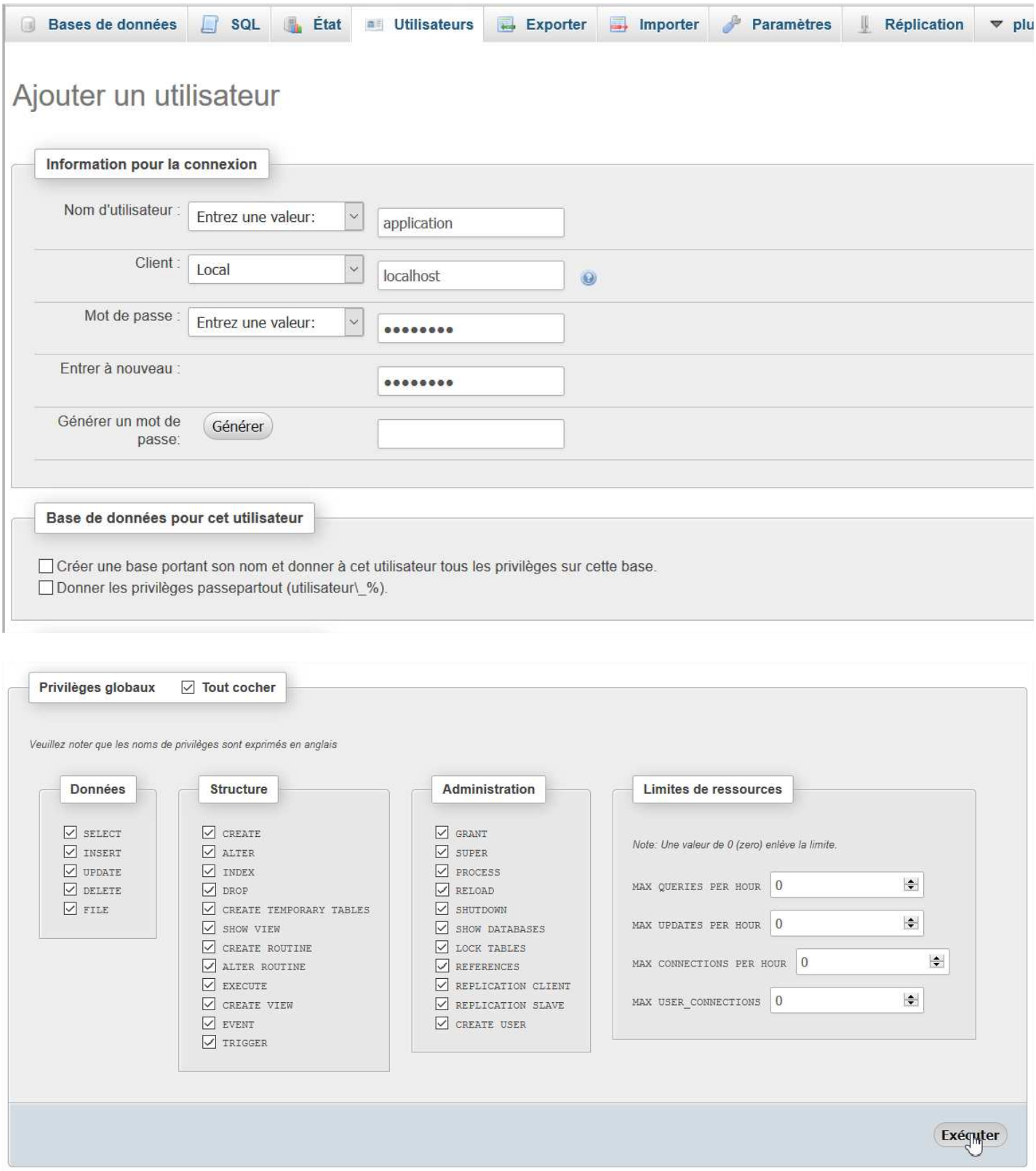
2. **Création de la base de données « base_quest_allo_docteur »** :

Créer la base MySQL [base_quest_allo_docteur] avec l'outil de votre choix soit en executants les instructions ci-dessous ou juste avec le script **script_base_quest_allo_docteur.sql**. La base sera par la suite la propriété de l'utilisateur « **application** » avec le mot de passe « **passw0rd** ».

Pour ajouter un nouveau utilisateur :

- Allez dans la console d'administration de **PhpMyAdmin**, puis cliquer sur « **Utilisateur** », puis sur « **Ajouter un utilisateur** ».
- Mettre « **application** » dans « Nom d'utilisateur », « **passw0rd** » dans « Mot de passe » et « **localhost** » dans « Client », Cocher « **Tout cocher** » dans « Privilèges globaux » et cliquer sur « **Exécuter** ».
- Réitérer l'opération précédente en mettant pour le client « **Tout Client** » c'est-à-dire la valeur **%**, puis la valeur « **127.0.0.1** ».




Bases de données
SQL
État
Utilisateurs
Exporter
Importer
Paramètres
Réplication
plu
Ajouter un utilisateur
Information pour la connexion
Nom d'utilisateur :
Entrez une valeur:
application
Client :
Local
localhost
Mot de passe :
Entrez une valeur:
Entrer à nouveau :
Générer un mot de passe:
Générer
Base de données pour cet utilisateur
Créer une base portant son nom et donner à cet utilisateur tous les privilèges sur cette base.
Donner les privilèges passepartout (utilisateur_%).
Privilèges globaux
Tout cocher
Veuillez noter que les noms de privilèges sont exprimés en anglais
Données
SELECT
INSERT
UPDATE
DELETE
FILE
Structure
CREATE
ALTER
INDEX
DROP
CREATE TEMPORARY TABLES
SHOW VIEW
CREATE ROUTINE
ALTER ROUTINE
EXECUTE
CREATE VIEW
EVENT
TRIGGER
Administration
GRANT
SUPER
PROCESS
RELOAD
SHUTDOWN
SHOW DATABASES
LOCK TABLES
REFERENCES
REPLICATION CLIENT
REPLICATION SLAVE
CREATE USER
Limites de ressources
Note: Une valeur de 0 (zero) enlève la limite.
MAX QUERIES PER HOUR
0
MAX UPDATES PER HOUR
0
MAX CONNECTIONS PER HOUR
0
MAX USER_CONNECTIONS
0
Exécuter

Après avoir créer la base de données **base_quest_allo_docteur** puis cliquer sur SQL et mettre le contenu du fichier **script_base_quest_allo_docteur.sql** à partir de **SET FOREIGN_KEY_CHECKS.**