

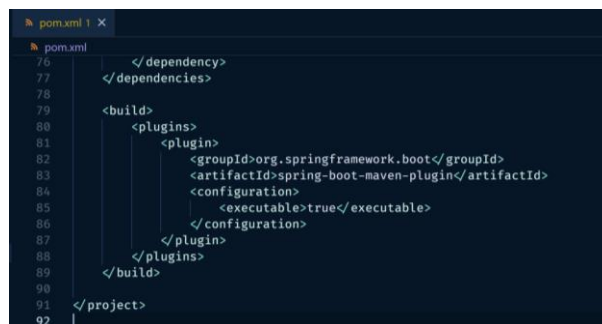
# Cours Inf3522 - Développement d'Applications N tiers

## Lab 15 : Déploiement de l'application

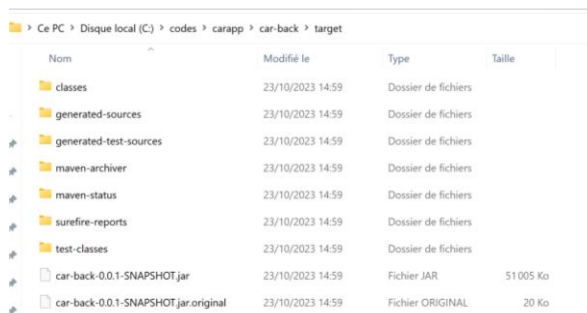
Ce lab expliquera comment déployer votre backend et votre frontend sur un serveur. Le déploiement réussi est une partie essentielle du processus de développement logiciel, et il est important d'apprendre comment fonctionne un processus de déploiement moderne. Il existe divers serveurs cloud ou fournisseurs de PaaS (Platform-as-a-Service) disponible mais dans ce cours, nous utilisons Heroku et surge.sh. Nous vous montrerons également comment utiliser des conteneurs Docker dans le déploiement dans le lab\_4 du cours Inf3523 – Architecture Logicielle.

### Déploiement du backend

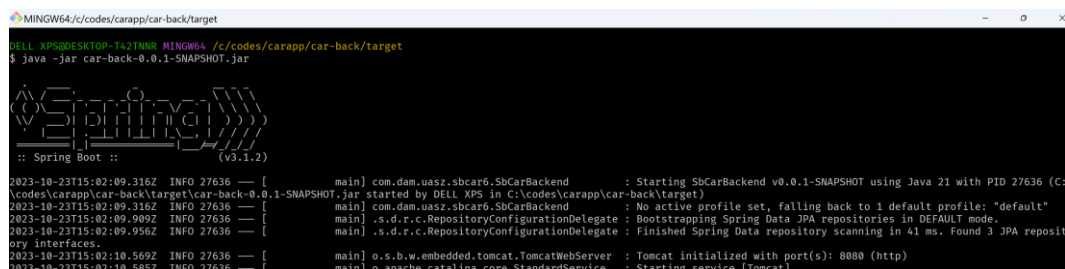
Si vous allez utiliser votre propre serveur, la manière la plus simple de déployer l'application Spring Boot est d'utiliser un fichier **Java ARchive (JAR)** exécutable. Si vous utilisez Maven, un fichier JAR exécutable peut être créé en utilisant le plugin Spring Boot Maven et en ajoutant les lignes de code suivantes à votre fichier **pom.xml** (Ligne 84-86) :



Ensuite, vous devez construire votre projet en utilisant la commande `'mvn clean install'`. Cette commande crée un fichier JAR dans le dossier **"target"**, comme illustré dans la capture d'écran suivante :



Dans ce cas, vous n'avez pas besoin d'installer un serveur d'application séparé, car il est intégré dans votre fichier JAR. Ensuite, il vous suffit d'exécuter le fichier JAR en utilisant la commande Java `'java -jar votrefichierapp.jar'`, comme illustré dans la capture d'écran suivante :



Le fichier .jar peut être copié et exécuté dans n'importe quel dossier de votre machine.

De nos jours, les serveurs cloud sont le principal moyen de fournir votre application aux utilisateurs finaux. Ensuite, nous allons déployer notre backend sur le serveur cloud Heroku (<https://www.heroku.com/>). Heroku propose un compte gratuit que vous pouvez utiliser pour déployer vos propres applications. Avec le compte gratuit, vos applications se mettent en veille après 30 minutes d'inactivité, et il faut un peu plus de temps pour redémarrer l'application. Cependant, le compte gratuit est suffisant à des fins de test et de loisir.

Nous allons donc déployer une nouvelle copie de votre backend dans heroku et configurer une BD à cette fin. Etant donné que quand vous créez une BD cela prend un peu de temps pour que vous puissiez travailler dessus, nous allons commencer par déployer une application vide qui va juste nous permettre de configurer une nouvelle BD. Pour cela, suivre les étapes ci-dessous.

Placez-vous dans votre dossier de travail, et créer un nouveau dossier nommé **db-deployed**. Puis initialisez un dépôt git avec. Ensuite ouvrir un prompt (dans cmd faire prompt puis @\$f dans windows) et se loguer dans heroku (installez heroku-cli : <https://devcenter.heroku.com/articles/heroku-cli>). Après une connexion à heroku (**heroku login**), créez une nouvelle application dans heroku avec la commande **heroku create** suivi du nom de l'application que j'appelle **mariadb-deployed**. Après cela créez un fichier texte contenant juste la chaîne "BD deployment" puis effectuer les commandes **git add .**, suivi de **git commit -am "db deployed"** et enfin la commande **git push heroku main** (ignorer les erreurs). Enfin faire un **heroku apps** pour voir les applications déployées et vérifiez que la nouvelle application est présente. Ces commandes sont présentées ci-dessous :

```
C:\Windows\System32\cmd.exe
Microsoft Windows [version 10.0.22621.2428]
(c) Microsoft Corporation. Tous droits réservés.

C:\codes\carapp\db-deployed>prompt @$f

@)git init
Initialized empty Git repository in C:/codes/carapp/db-deployed/.git/

@)git remote -v

@)heroku login
heroku: Press any key to open up the browser to login or q to exit:
Opening browser to https://cli-auth.heroku.com/auth/cli/browser/8b55a623-c94e-4597-b7b6-4ba344575645?requestor=SFMNTY.g2gDbQAAAA4xNT
QM7TILJE3MS43M24GALhkk2eLAWIAAVGA.uTZe5rKuh54nNaSdas0wxs0A5eQerd3BJU-QC7w6vGh
Logging in... done
Logged in as bassirou.toure@univ-zig.sn

@)heroku create mariadb-deployed
Creating • mariadb-deployed... done
https://mariadb-deployed-16338dc18c0d.herokuapp.com/ | https://git.heroku.com/mariadb-deployed.git

@)code .

@)git add .

@)git commit -am "db deployed"
[main (root-commit) 5dfc171] db deployed
1 file changed, 1 insertion(+)
create mode 100644 hello.txt

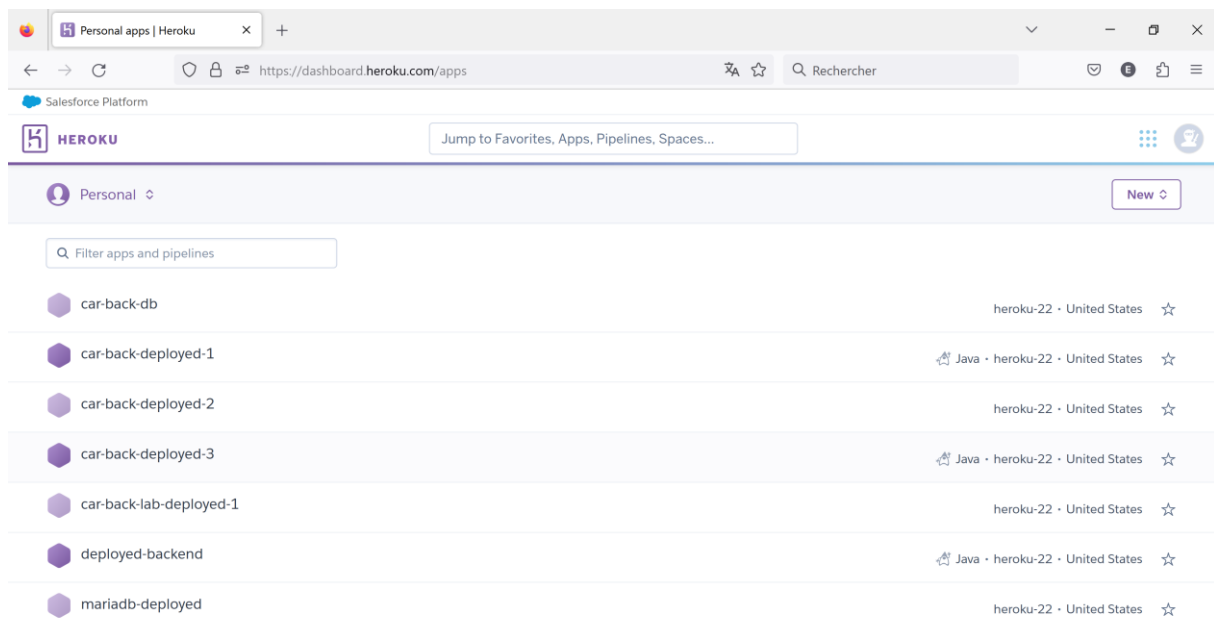
@)git remote -v
heroku https://git.heroku.com/mariadb-deployed.git (fetch)
heroku https://git.heroku.com/mariadb-deployed.git (push)

@)git push heroku main
```

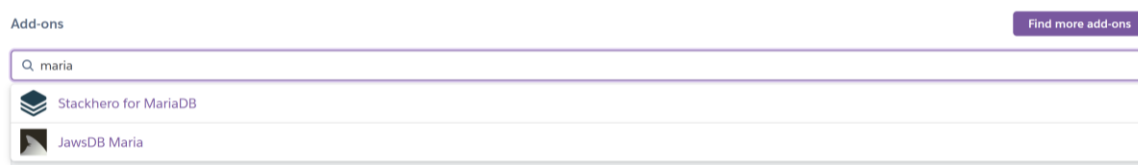
```
C:\Windows\System32\cmd.e x + v
@)git push heroku main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 230 bytes | 230.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote: Updated 1 path from 8c706ce
remote: Compressing source files... done.
remote: Building source:
remote:
remote: -----> Building on the Heroku-22 stack
remote: -----> Determining which buildpack to use for this app
remote: ! No default language could be detected for this app.
remote: HINT: This occurs when Heroku cannot detect the buildpack to use for this application automatically.
remote: See https://devcenter.heroku.com/articles/buildpacks
remote:
remote: ! Push failed
remote: Verifying deploy...
remote:
remote: ! Push rejected to mariadb-deployed.
remote:
To https://git.heroku.com/mariadb-deployed.git
! [remote rejected] main -> main (pre-receive hook declined)
error: failed to push some refs to 'https://git.heroku.com/mariadb-deployed.git'

@)heroku apps
=== bassirou.toure@univ-zig.sn Apps
car-back-db
car-back-deployed-1
car-back-deployed-2
car-back-deployed-3
car-back-lab-deployed-1
deployed-backend
mariadb-deployed
```

Après vous allez dans <https://dashboard.heroku.com/apps> et sélectionnez la nouvelle application mariadb-deployed :



Après avoir sélectionné l'application, cliquez sur **resources** puis dans la barre de recherche des **add-ons**, recherchez maria et choisir dans la liste déroulante **JawsDB Maria** et cliquez sur **submit** pour lier cette BD à notre application :



Ensuite cliquez sur **Manage Attachments** puis sur le lien **jawsdb-maria-polished-48399** (vous aurez d'autres valeurs) de la fenêtre popup qui s'affiche.

The screenshot shows the Heroku Add-ons page for the 'jawsdb-maria' add-on. A notification at the top states: 'The add-on jawsdb-maria has been installed. Check out the documentation in its Dev Center article to get started.' Below this, a search bar prompts 'Quickly add add-ons from Elements'. The 'JawsDB Maria' add-on is listed with a status of 'Attached as JAWSDB\_MARIA', a 'Kitefin Shared' provider, and a 'Free' price tag. An 'Estimated Monthly Cost' of '\$0' is shown. A 'Support' button is available. A 'Manage Attachments' button is also present. A popup window titled 'Add-on Attachments' is open, showing the 'JawsDB Maria' add-on and its attachment to the application 'This app as JAWSDB\_MARIA'. Below the popup, a browser window shows the 'JawsDB' dashboard with connection information.

**Connection Info**

Connection String

```
mysql://t3js7umahhanmwiq:wwlwt260q3eary9a@un0jueuv2mam78uv.cbetxkdyhsb.us-east-1.rds.amazonaws.com:3306/xukti1kzhnlvsns0
```

You can use your connection information to connect manually through a client such as [HeidiSQL](#) to administer your database.

Property	Value	Action
Host	un0jueuv2mam78uv.cbetxkdyhsb.us-east-1.rds.amazonaws.com	
Username	t3js7umahhanmwiq	
Password	wwlwt260q3eary9a	<button>Reset</button>
Port	3306	
Database	xukti1kzhnlvsns0	

Ces informations (**url de la BD, username, password**) sont à mettre dans votre fichier **application.properties** de la copie de votre backend à déployer plus tard.

Après cela naviguez dans votre dossier contenant votre backend puis initialisez un nouveau dépôt git, et créez une nouvelle application dans heroku puis suivre les commandes ci-dessous :

```
C:\Windows\System32\cmd.e x + v
Microsoft Windows [version 10.0.22621.2428]
(c) Microsoft Corporation. Tous droits réservés.

C:\codes\carapp\backend-deployed>prompt @$f

@)heroku login
heroku: Press any key to open up the browser to login or q to exit:
Opening browser to https://cli-auth.heroku.com/auth/cli/browser/18194ff9-3ba3-4110-b16f-e8943978fa1b?requestor=SFMyNTY.g2gDbQAAAA4xNT
QuMTi1LjE3MS43M24GAPCIW2eLAWIAAVGA.qv9wWDZb5OG0MFIr2baj0f83YNwtYW5EaV9LLyR3wY
Logging in... done
Logged in as bassirou.toure@univ-zig.sn

@)git init
Initialized empty Git repository in C:\codes\carapp\backend-deployed/.git/

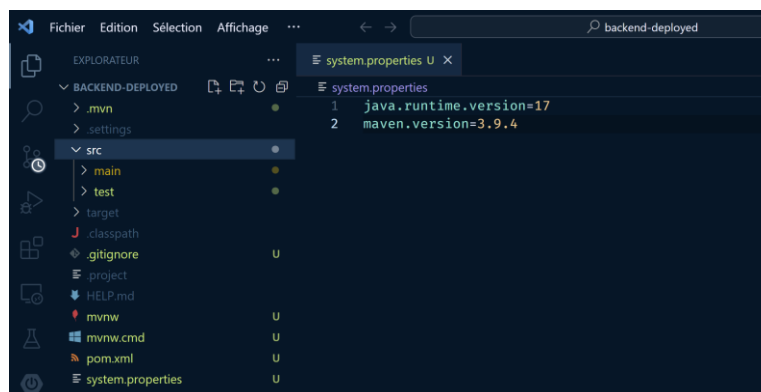
@)git remote -v

@)heroku create carbackend
Creating • carbackend... !
! Name carbackend is already taken

@)heroku create carbackend-2
Creating • carbackend-2... done
https://carbackend-2-d03705863ad3.herokuapp.com/ | https://git.heroku.com/carbackend-2.git

@)
```

Ensuite modifiez le code en ajoutant un fichier **system.properties** dans la racine de votre application backend et y mettre le contenu ci-dessous :



Ensuite modifiez le fichier **application.properties** pour y mettre la configuration précédente comme suit (ces informations à prendre de votre BD JawsDB Maria précédente) :



Faites les commandes **git add .**, suivi de **git commit -am "backend deployed"** et **git push heroku main**.

```
C:\Windows\System32\cmd.e x + v
@)git commit -am "backend deployed"
[main (root-commit) 744fddc] backend deployed
24 files changed, 1351 insertions(+)
create mode 100644 .gitignore
create mode 100644 .mvn/wrapper/maven-wrapper.jar
create mode 100644 .mvn/wrapper/maven-wrapper.properties
create mode 100644 mvnw
create mode 100644 mvnw.cmd
create mode 100644 pom.xml
create mode 100644 src/main/java/com/dam/uasz/sbcar6/AuthEntryPoint.java
create mode 100644 src/main/java/com/dam/uasz/sbcar6/AuthenticationFilter.java
create mode 100644 src/main/java/com/dam/uasz/sbcar6/SbCarBackend.java
create mode 100644 src/main/java/com/dam/uasz/sbcar6/SecurityConfig.java
create mode 100644 src/main/java/com/dam/uasz/sbcar6/domain/AccountCredentials.java
create mode 100644 src/main/java/com/dam/uasz/sbcar6/domain/BaseEntity.java
create mode 100644 src/main/java/com/dam/uasz/sbcar6/domain/Car.java
create mode 100644 src/main/java/com/dam/uasz/sbcar6/domain/CarRepository.java
create mode 100644 src/main/java/com/dam/uasz/sbcar6/domain/Owner.java
create mode 100644 src/main/java/com/dam/uasz/sbcar6/domain/OwnerRepository.java
create mode 100644 src/main/java/com/dam/uasz/sbcar6/domain/User.java
create mode 100644 src/main/java/com/dam/uasz/sbcar6/domain/UserRepository.java
create mode 100644 src/main/java/com/dam/uasz/sbcar6/service/JwtService.java
create mode 100644 src/main/java/com/dam/uasz/sbcar6/service/UserDetailsServiceImpl.java
create mode 100644 src/main/java/com/dam/uasz/sbcar6/web/LoginController.java
create mode 100644 src/main/resources/application.properties
create mode 100644 src/test/java/com/dam/uasz/sbcar6/CarBackApplicationTests.java
create mode 100644 system.properties

@)git push heroku main
Enumerating objects: 45, done.
Counting objects: 100% (45/45), done.
Delta compression using up to 8 threads
Compressing objects: 100% (33/33), done.
Writing objects: 100% (45/45), 70.66 KiB | 7.07 MiB/s, done.
Total 45 (delta 2), reused 0 (delta 0), pack-reused 0
```

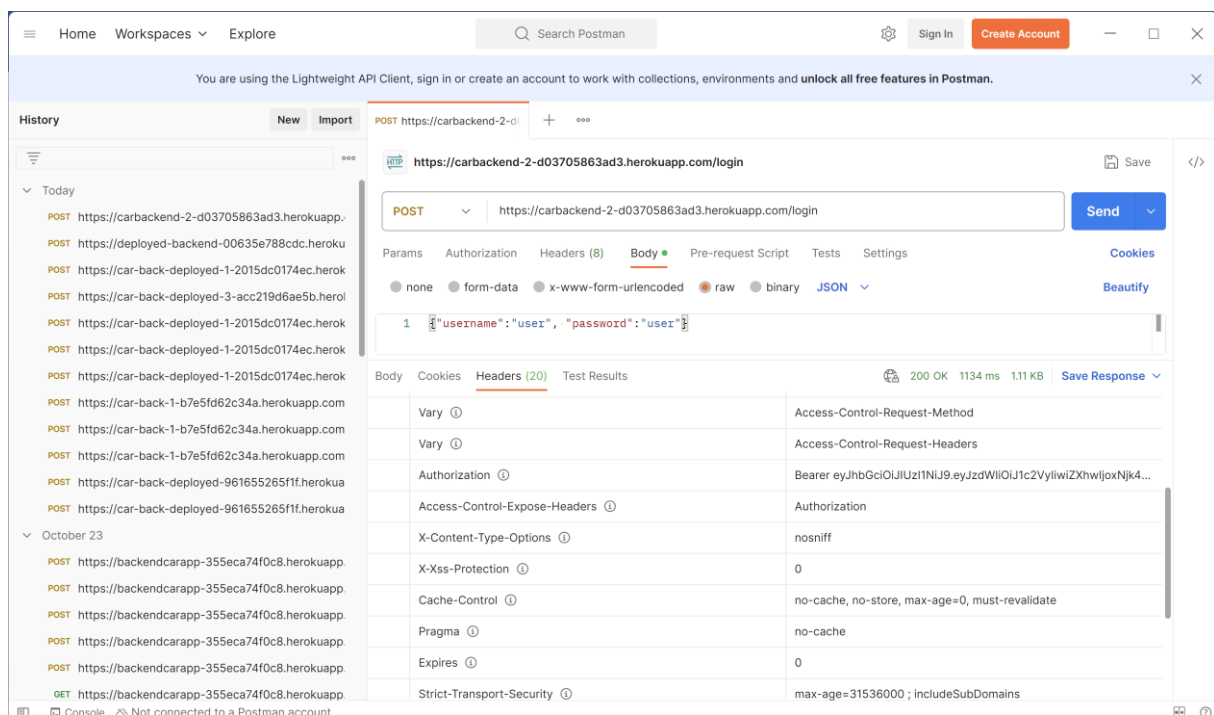
```
C:\Windows\System32\cmd.e x + v
nnotations-1.3.jar (8.8 kB at 54 kB/s)
remote: [INFO] Downloaded from central: https://repo.maven.apache.org/maven2/org/checkerframework/checker-compat-qual/2.5.5/checker-compat-qual-2.5.5.jar (5.9 kB at 35 kB/s)
remote: [INFO] Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/commons/commons-lang3/3.7/commons-lang3-3.7.jar (500 kB at 2.9 MB/s)
remote: [INFO] Downloaded from central: https://repo.maven.apache.org/maven2/com/google/guava/guava/28.2-android/guava-28.2-android.jar (2.6 MB at 14 MB/s)
remote: [INFO] Replacing main artifact /tmp/build_fef396e5/target/car-back-0.0.1-SNAPSHOT.jar with repackaged archive, adding nested dependencies in BOOT-INF/.
remote: [INFO] The original artifact has been renamed to /tmp/build_fef396e5/target/car-back-0.0.1-SNAPSHOT.jar.original
remote: [INFO] --- install:3.1.1:install (default-install) @ car-back ---
remote: [INFO] Installing /tmp/build_fef396e5/pom.xml to /tmp/codon/tmp/cache/.m2/repository/com/dam/uasz/car-back/0.0.1-SNAPSHOT/car-back-0.0.1-SNAPSHOT.pom
remote: [INFO] Installing /tmp/build_fef396e5/target/car-back-0.0.1-SNAPSHOT.jar to /tmp/codon/tmp/cache/.m2/repository/com/dam/uasz/car-back/0.0.1-SNAPSHOT/car-back-0.0.1-SNAPSHOT.jar
remote: [INFO] -----
remote: [INFO] BUILD SUCCESS
remote: [INFO] -----
remote: [INFO] Total time: 12.850 s
remote: [INFO] Finished at: 2023-10-25T15:10:04Z
remote: [INFO] -----
remote: ----> Discovering process types
remote: Procfile declares types -> (none)
remote: Default types for buildpack -> web
remote: ----> Compressing...
remote: Done: 111.1M
remote: ----> Launching...
remote: Released v3
remote: https://carbackend-2-d03705863ad3.herokuapp.com/ deployed to Heroku
remote: Verifying deploy... done.
To https://git.heroku.com/carbackend-2.git
* [new branch] main -> main

@)
```

Effectuez la commande **heroku apps** pour voir que notre application déployée est bien présente puis **heroku open** pour la lancer. Effectivement ça va générer une erreur vu la configuration actuelle de notre serveur (erreur à ignorer). Tapez la commande **heroku logs** pour voir le logs. On obtient **resource not found 404** (voir capture suivante) parce que la seule route reconnue actuellement est **/login**.

```
C:\Windows\System32\cmd.e x + v
oryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2023-10-25T15:10:21.146030+00:00 app[web.1]: 2023-10-25T15:10:21.145Z INFO 2 --- [main] o.s.d.j.r.query.QueryEnhancerFact
ory : Hibernate is in classpath; If applicable, HQL parser will be used.
2023-10-25T15:10:21.540620+00:00 app[web.1]: 2023-10-25T15:10:21.540Z WARN 2 --- [main] JpaBaseConfiguration$JpaWebConfig
uration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be performed during view rendering. Explicit
ly configure spring.jpa.open-in-view to disable this warning
2023-10-25T15:10:21.700059+00:00 app[web.1]: 2023-10-25T15:10:21.699Z INFO 2 --- [main] o.s.s.web.DefaultSecurityFilterCh
ain : Will secure any request with [org.springframework.security.web.session.DisableEncodeUrlFilter@f245bdd, org.springframework
.security.web.context.request.async.WebAsyncManagerIntegrationFilter@06af609ea, org.springframework.security.web.context.SecurityConte
xtholderFilter@ccc9ef8d, org.springframework.security.web.header.HeaderWriterFilter@19f02280, org.springframework.web.filter.CorsFilter
@280fafd, org.springframework.security.web.authentication.logout.LogoutFilter@339f3a55, org.springframework.security.web.savedrequest
.RequestCacheAwareFilter@c412556, org.springframework.security.web.servletapi.SecurityContextHolderAwareRequestFilter@450f0235, org.s
pringframework.security.web.authentication.AnonymousAuthenticationFilter@39e53bef, org.springframework.security.web.access.ExceptionT
ranslationFilter@7fef0b40, org.springframework.security.web.access.intercept.AuthorizationFilter@77524ca7]
2023-10-25T15:10:22.181031+00:00 app[web.1]: 2023-10-25T15:10:22.180Z INFO 2 --- [main] o.s.b.w.embedded.tomcat.TomcatWeb
Server : Tomcat started on port(s): 15046 (http) with context path ''
2023-10-25T15:10:22.191768+00:00 app[web.1]: 2023-10-25T15:10:22.191Z INFO 2 --- [main] com.dam.uaszbcar6.SbCarBackend
: Started SbCarBackend in 4.762 seconds (process running for 5.233)
2023-10-25T15:10:22.340500+00:00 app[web.1]: 2023-10-25T15:10:22.340Z INFO 2 --- [main] com.dam.uaszbcar6.SbCarBackend
: Ford Mustang
2023-10-25T15:10:22.340534+00:00 app[web.1]: 2023-10-25T15:10:22.340Z INFO 2 --- [main] com.dam.uaszbcar6.SbCarBackend
: Nissan Leaf
2023-10-25T15:10:22.340568+00:00 app[web.1]: 2023-10-25T15:10:22.340Z INFO 2 --- [main] com.dam.uaszbcar6.SbCarBackend
: Toyota Prius
2023-10-25T15:10:22.457240+00:00 heroku[web.1]: State changed from starting to up
2023-10-25T15:12:18.930804+00:00 app[web.1]: 2023-10-25T15:12:18.930Z INFO 2 --- [io-15046-exec-4] o.a.c.c.c.[Tomcat].[/lo
calhost].[/]
: Initializing Spring DispatcherServlet 'dispatcherServlet'
2023-10-25T15:12:18.931078+00:00 app[web.1]: 2023-10-25T15:12:18.930Z INFO 2 --- [io-15046-exec-4] o.s.web.servlet.DispatcherServlet
: Initializing Servlet 'dispatcherServlet'
2023-10-25T15:12:18.932546+00:00 app[web.1]: 2023-10-25T15:12:18.932Z INFO 2 --- [io-15046-exec-4] o.s.web.servlet.DispatcherServlet
: Completed initialization in 2 ms
2023-10-25T15:12:18.994023+00:00 heroku[router]: at=info method=GET path="/" host=carbackend-2-d03705863ad3.herokuapp.com request_id=
40a08837-f485-4ab4-a6c1-dab2d3a2f717 fwd="154.125.171.73" dyno=web.1 connect=0ms service=85ms status=404 bytes=757 protocol=https
2023-10-25T15:12:18.502376+00:00 heroku[router]: at=info method=GET path="/favicon.ico" host=carbackend-2-d03705863ad3.herokuapp.com
request_id=d2295c28-6a5a-4cfa-a75e-8240cb842fe9 fwd="154.125.171.73" dyno=web.1 connect=0ms service=62ms status=404 bytes=568 proto
col=https
```

Copiez le lien heroku de notre serveur et le tester dans postman comme suit :

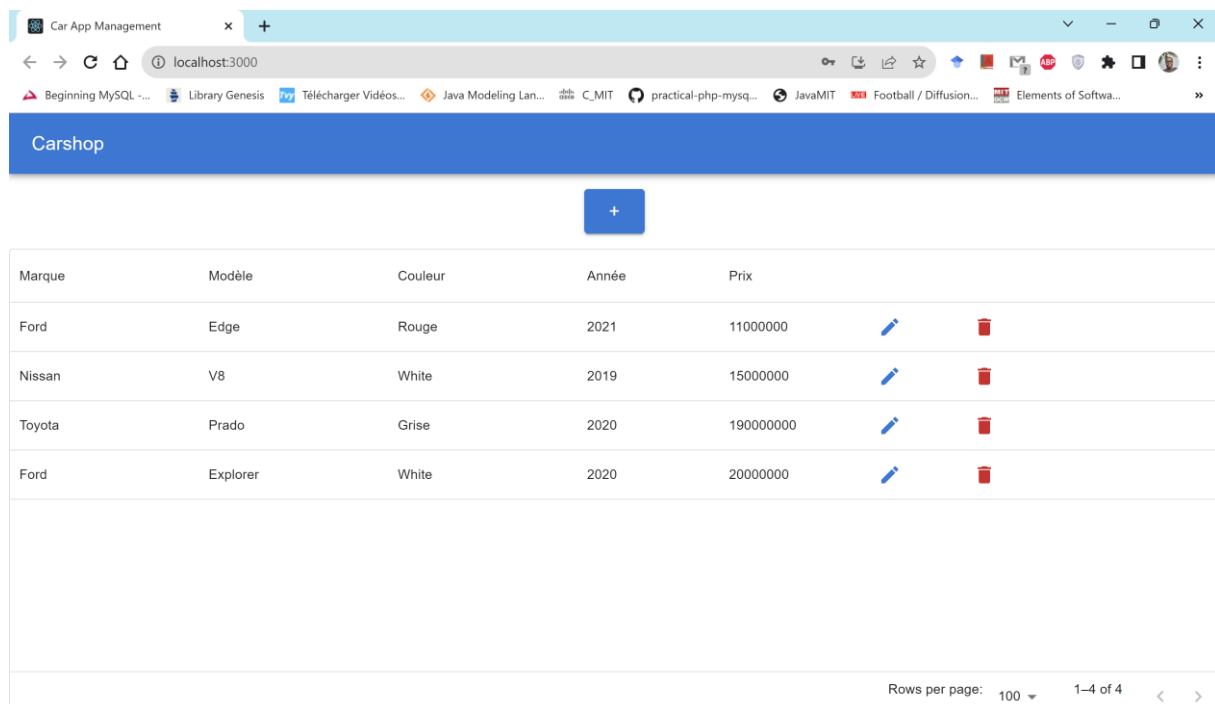










Nous voyons que le jeton est présent dans la réponse dans **Authorization**. Pour tester ce serveur avec notre frontend, nous allons modifier l'url de notre serveur backend dans le fichier `constants.js` comme suit :

```
JS constants.js x
src > JS constants.js > ...
1 export const SERVER_URL = "https://carbackend-2-d03705863ad3.herokuapp.com/";
2 |
```



Enfin nous pouvons lancer notre frontend avec `npm start` qui va utiliser comme backend celui qui est déployé dans heroku : <https://carbackend-2-d03705863ad3.herokuapp.com/>



Marque	Modèle	Couleur	Année	Prix		
Ford	Edge	Rouge	2021	11000000		
Nissan	V8	White	2019	15000000		
Toyota	Prado	Grise	2020	190000000		
Ford	Explorer	White	2020	20000000		

Rows per page: 100 1-4 of 4

Maintenant nous pouvons déployer notre frontend.

### Déploiement du frontend

Nous allons surge (<https://surge.sh/>) pour un déploiement rapide de notre frontend.

Le déploiement d'une application React sur Surge est un processus simple. Surge est un service d'hébergement statique qui vous permet de publier des sites web statiques rapidement. Voici comment déployer une application React sur Surge :

Installez Surge : Vous pouvez installer Surge en utilisant npm en ouvrant votre terminal et en exécutant la commande suivante :

```
npm install -g surge
```

Créez un build de votre application React : Avant de déployer votre application, assurez-vous de créer une version de production de votre application React en exécutant la commande suivante dans le répertoire racine de votre projet React :

```
npm run build
```

Cela générera les fichiers statiques de votre application dans le dossier 'build'.

Déployez votre application sur Surge : Utilisez la commande 'surge' pour déployer votre application. Exécutez la commande suivante dans le répertoire 'build' (ajoutez build dans le chemin proposé) de votre projet React :

```
surge
```



Cette commande lancera un processus interactif où vous devrez choisir un sous-domaine pour votre site (par exemple, `mon-site.surge.sh`). Si c'est la première fois que vous utilisez Surge, vous devrez créer un compte ou vous connecter. Suivez les instructions à l'écran pour terminer le déploiement.

Confirmez le déploiement : Une fois le processus de déploiement terminé, Surge vous fournira un lien vers votre site web. Vous pourrez accéder à votre application React en utilisant ce lien.

Votre application React est maintenant déployée sur Surge et accessible publiquement. Vous pouvez partager le lien avec d'autres personnes ou l'utiliser pour tester votre application dans un environnement de production.

```
MINGW64/c/codes/carapp/car-front-sec
HELL XPSDESKTOP-T42TNR MINGW64 /c/codes/carapp/car-front-sec
$ surge

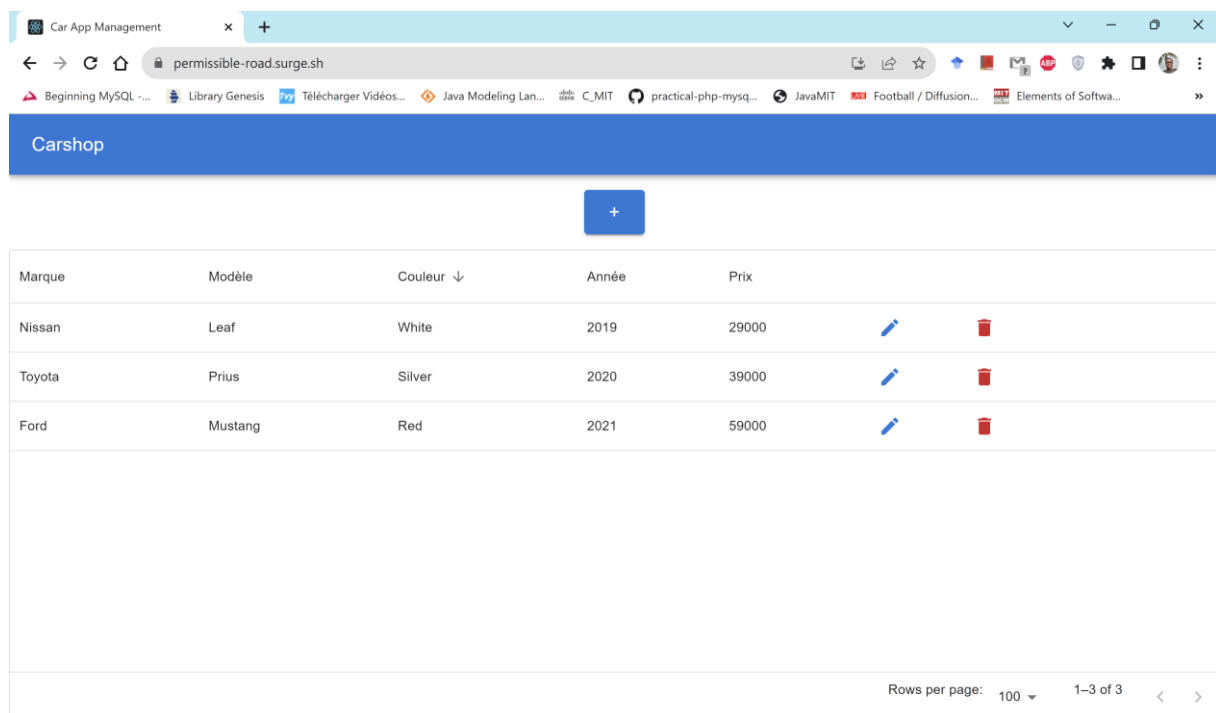
Running as bassirou.toure@univ-zig.sn (Student)

project: C:\codes\carapp\car-front-sec\build
domain: permissible-road.surge.sh

encryption: *.surge.sh, surge.sh (206 days)
IP: 138.197.235.123

Success! - Published to permissible-road.surge.sh
```

Maintenant si nous essayons d'accéder à l'url : <https://permissible-road.surge.sh/>, nous obtenons :



## Résumé

Dans ce lab, vous avez appris comment déployer l'application Spring Boot. Nous avons déployé l'application sur Heroku. Ensuite, nous avons déployé notre interface React sur surge sh.

Enfin, nous avons utilisé Docker dans le lab\_4, du cours INF3523 : Architecture et Génie des Logiciels, pour créer des conteneurs à partir de notre application Spring Boot et de la base de données MariaDB.