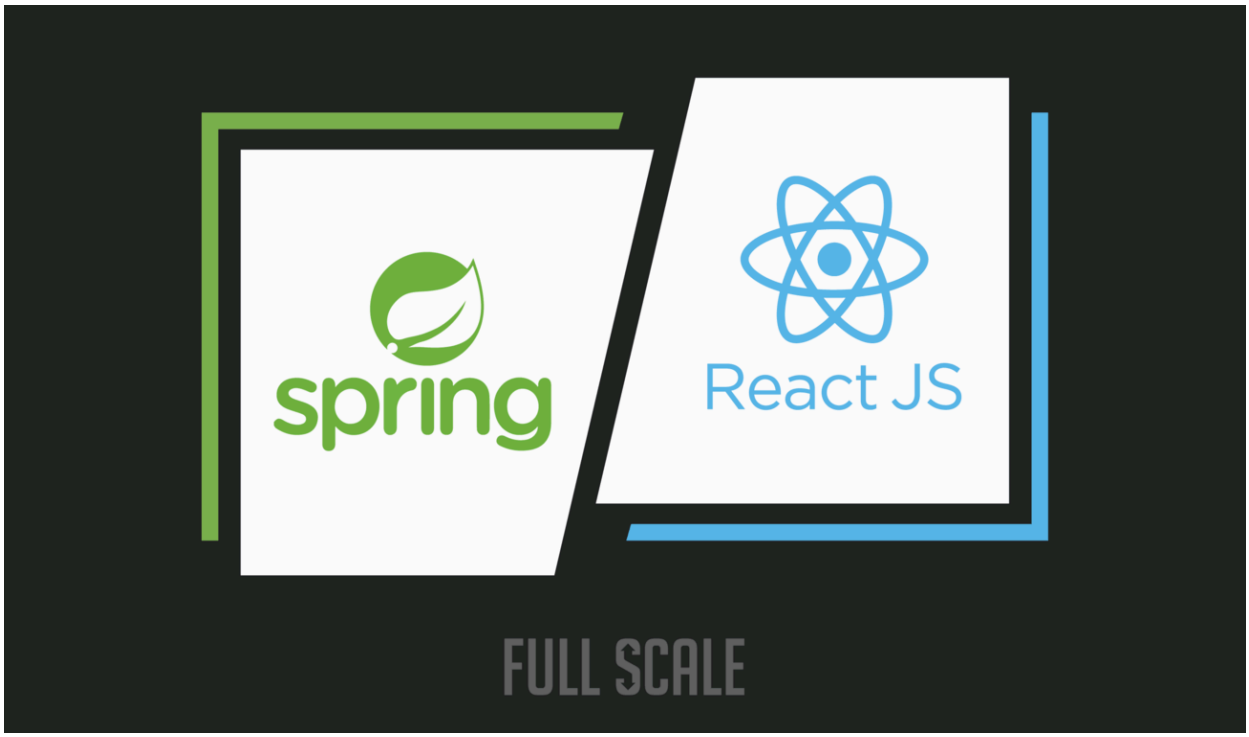
Developer Guide
Pathfindr



Introduction

The project is divided into two repositories, a server repository and a client repository. You need to clone the two repositories from Moule.

The server : https://moule.informatique.univ-paris-diderot.fr/groupe-a/projet

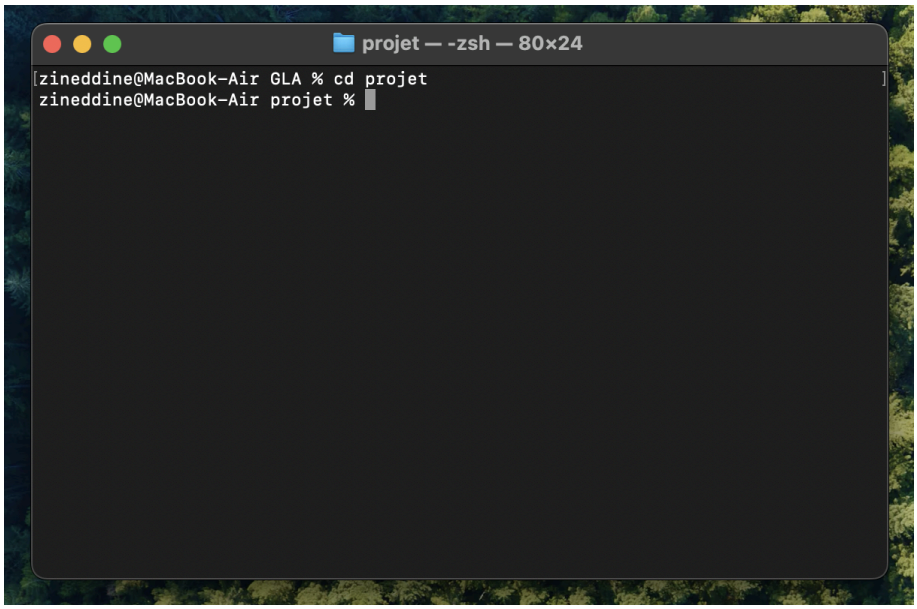
The client : https://moule.informatique.univ-paris-diderot.fr/groupe-a/project-frontend

Execute the project

for that you will need two terminals

- ## Server

Before all, you must have **Java** with **JDK 17** and also **Maven** must be installed in your machine.
After the installation, you have to open a terminal, and change the directory into the folder **projet**.



To install the dependencies of the project you must execute the command **mvn install.**

```
projet — -zsh — 80×24
[INFO] Tests run: 44, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] --- jacoco:0.8.11:report (post-unit-test) @ project ---
[INFO] Loading execution data file /Users/zineddine/Downloads/Archive/University
/GLA/projet/target/jacoco.exec
[INFO] Analyzed bundle 'Project Base' with 51 classes
[INFO]
[INFO] --- jar:3.3.0:jar (default-jar) @ project ---
[INFO]
[INFO] --- install:3.1.1:install (default-install) @ project ---
[INFO] Installing /Users/zineddine/Downloads/Archive/University/GLA/projet/pom.x
ml to /Users/zineddine/.m2/repository/fr/u-paris/gla/project/2024.1.0.0-SNAPSHOT
/project-2024.1.0.0-SNAPSHOT.pom
[INFO] Installing /Users/zineddine/Downloads/Archive/University/GLA/projet/targe
t/project-2024.1.0.0-SNAPSHOT.jar to /Users/zineddine/.m2/repository/fr/u-paris/
gla/project/2024.1.0.0-SNAPSHOT/project-2024.1.0.0-SNAPSHOT.jar
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  2.957 s
[INFO] Finished at: 2024-05-05T15:18:37+02:00
[INFO] ------------------------------------------------------------------------
zineddine@MacBook-Air projet %
```

Finally execute the command **mvn spring-boot:run**

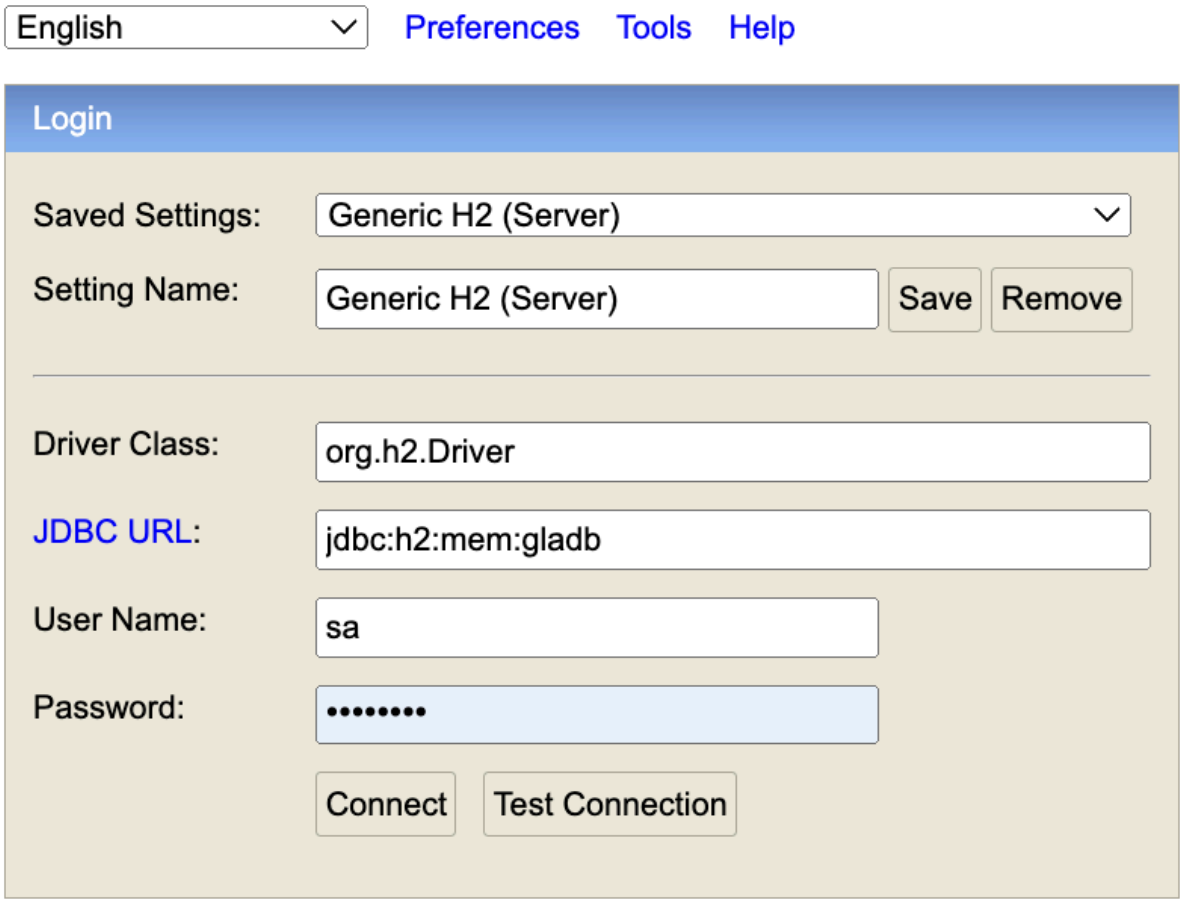
```
projet — java ◂ java -classpath /opt/homebrew/Cellar/maven/3.9.6/libexec/...
2024-05-05T15:21:16.452+02:00  INFO 8098 --- [  restartedMain] o.s.o.j.p.SpringP
ersistenceUnitInfo      : No LoadTimeWeaver setup: ignoring JPA class transforme
r
2024-05-05T15:21:16.818+02:00  INFO 8098 --- [  restartedMain] o.h.e.t.j.p.i.Jta
PlatformInitiator       : HHH000489: No JTA platform available (set 'hibernate.t
ransaction.jta.platform' to enable JTA platform integration)
2024-05-05T15:21:16.844+02:00  INFO 8098 --- [  restartedMain] j.LocalContainerE
ntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence u
nit 'default'
2024-05-05T15:21:17.035+02:00  WARN 8098 --- [  restartedMain] JpaBaseConfigurat
ion$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefo
re, database queries may be performed during view rendering. Explicitly configur
e spring.jpa.open-in-view to disable this warning
2024-05-05T15:21:17.291+02:00  INFO 8098 --- [  restartedMain] o.s.b.d.a.Optiona
lLiveReloadServer       : LiveReload server is running on port 35729
2024-05-05T15:21:17.296+02:00  INFO 8098 --- [  restartedMain] o.s.b.a.e.web.End
pointLinksResolver      : Exposing 1 endpoint(s) beneath base path '/actuator'
2024-05-05T15:21:17.324+02:00  INFO 8098 --- [  restartedMain] o.s.b.w.embedded.
tomcat.TomcatWebServer  : Tomcat started on port 8080 (http) with context path '
/api/v1'
2024-05-05T15:21:17.331+02:00  INFO 8098 --- [  restartedMain] f.u_paris.gla.pro
ject.ServerApplication  : Started ServerApplication in 1.969 seconds (process ru
nning for 2.125)
```

The server will be running in the localhost at the port 8080.

If you want to access to the database go to this link :

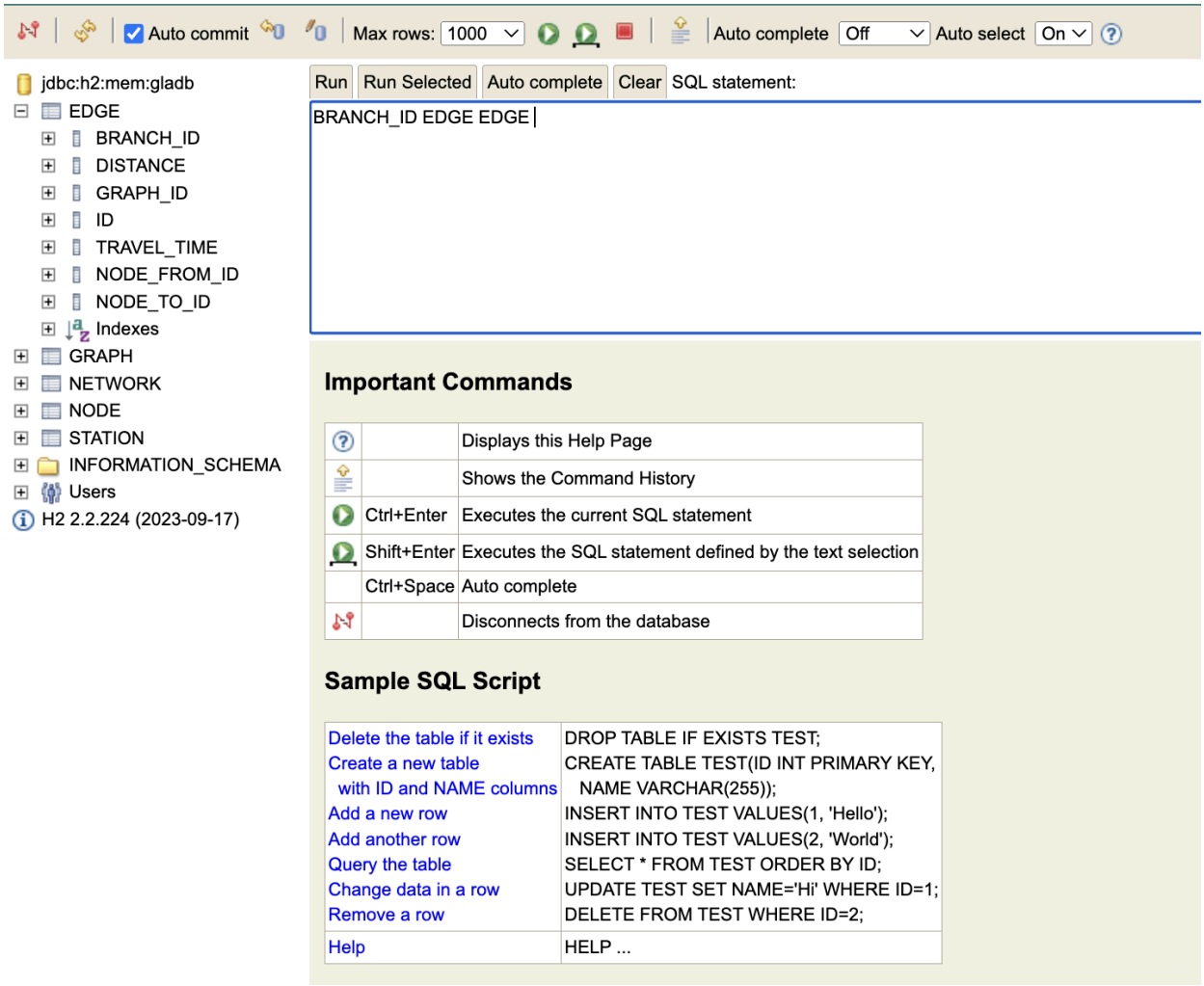
http://localhost:8080/api/v1/h2-console

you will need these information to access :



The password is : **password**

You will have this interface :

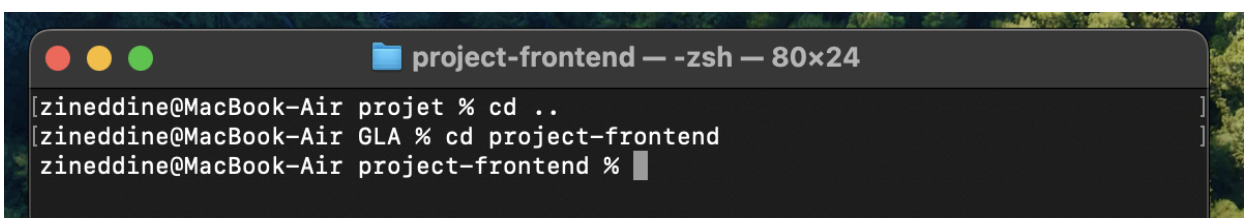


- ## Client

You will need a second terminal for testing the front-end.

Before all, you will need to install **npm** and also a recent version of **Node (20.12 or later)**.

After that, you change the directory into **project-frontend.**



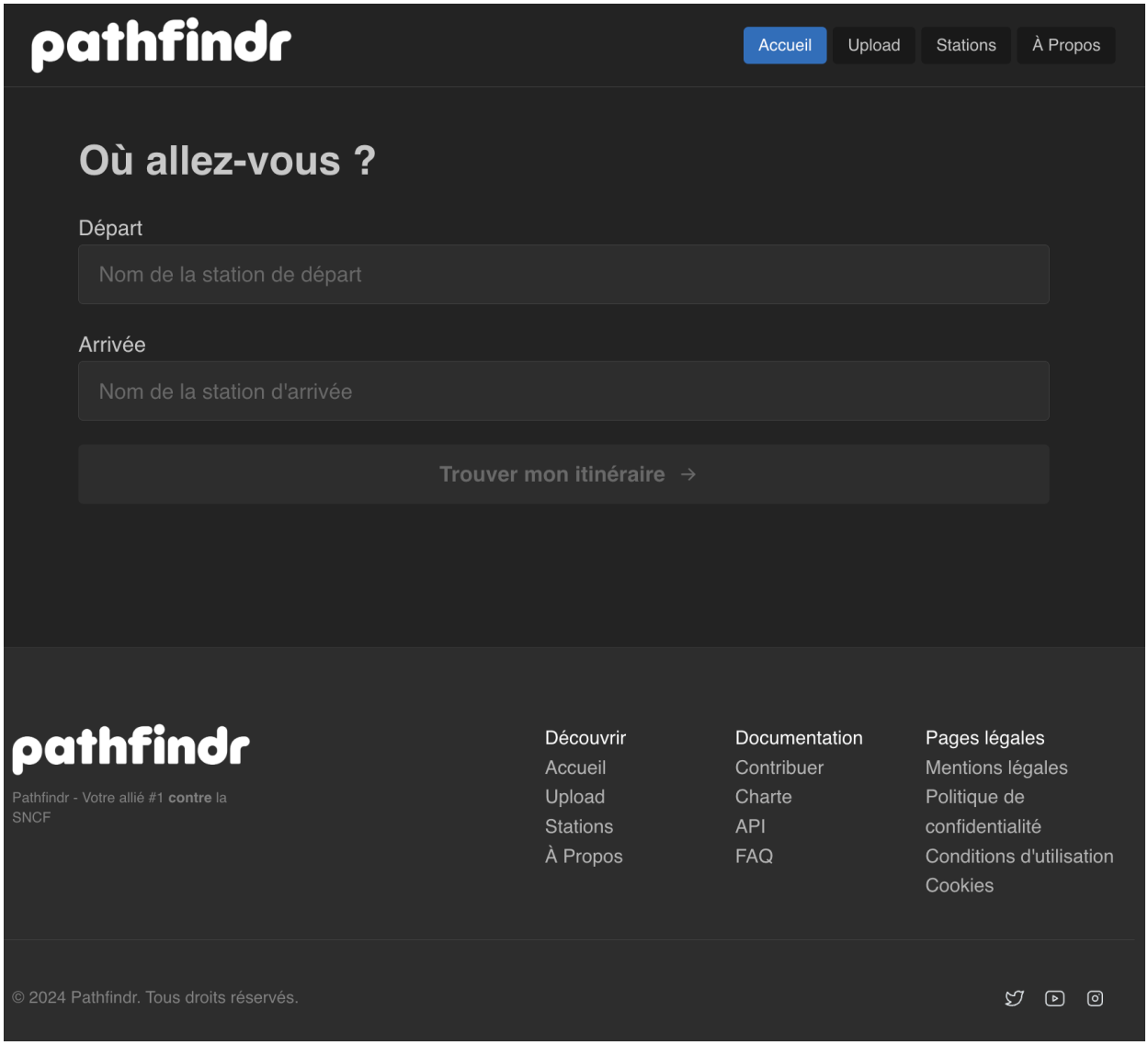
To install the dependencies execute **npm install**.

```
project-frontend — -zsh — 80×24
zineddine@MacBook-Air project-frontend % npm install

up to date, audited 498 packages in 528ms

173 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
zineddine@MacBook-Air project-frontend %
```

And finally to execute the front-end execute **npm run dev**.

```
project-frontend — esbuild ◂ npm run dev __CFBundleIdentifier=com.app...
zineddine@MacBook-Air project-frontend % npm run dev

> project-frontend@0.0.0 dev
> vite

Port 5173 is in use, trying another one...

  VITE v5.2.8  ready in 76 ms

  ➜  Local:   http://localhost:5174/
  ➜  Network: use --host to expose
  ➜  press h + enter to show help
```

You will find the client web site in http://localhost:5174/



Now you can use our software Pathfindr.

Client deployment:

To deploy to production, generate static assets using **npm run build.**

Afterwards, your build project will be available in the dist/ directory.

You can then statically serve it using an Apache or NGINX server.

API documentation of the project

You will find the API documentation of the project in this link

Architecture of the project

You will find the uml of the project in this link

Complete Documentation and conventions

You will find the wiki page of the project in this link

How to improve the project

- **Adding the Read Line feature**

The function readLine is already in the project (in the folder reading), all you have to do is create a new controller in the server and also a route that you can name lines.

- **Functionality of having several networks in the app**

The current version of the app only supports the use of one network at a time. However, in a future update, we plan to expand this functionality to allow for the use of multiple networks simultaneously. For example, one for Paris and one for Lyon.

- **Add the schedules models and functionalities**

in the branch **41-implement-schedules-functions** which is the dev branch, you will find the beginning of the implementation of the schedules, you can continue from this branch to implement the feature of reading schedules from the csv file. You can for example create a function that takes a station and a time as arguments and return the next schedule where a transport means will pass. After that you can integrate this function into the pathfinder function or add conditions to the dijkstra algorithm.

- ### Add an Api to the front end

For the front end, we can add additional APIs by adding a file in src/api like NetworkAPI to manage multiple networks. It's as simple as:

1. Adding a component. fonc
2. Calling the API in the component and fetching the data.
3. Displaying the data with components.