

RAPPORT DE PROJET

SHAPE

Titre Professionnel

Concepteur Développeur D'application

Adrien Houen

2023





Remerciements

La réalisation de ce projet a été possible grâce au concours de plusieurs personnes à qui je voudrais témoigner toute ma gratitude.

Je souhaite remercier toute l'équipe de l'AFPA de Champs-sur-Marne, qui m'a donné la chance de suivre cette formation. Je tiens à remercier en particulier Monsieur Paul Edide, mon professeur, ainsi que Madame Valérie Onillon, responsable de formation pour leur écoute pendant la formation.

Merci encore à l'ensemble de mes camarades de classe qui m'a aidé et avec qui j'ai pu travailler efficacement et progresser tout au long de ma formation.

Je pense également à Monsieur Félix Barnaud, fondateur de l'entreprise Au-delà, qui a cru en mon potentiel et m'a accueilli au sein de son entreprise pour mon stage.

Mes proches ont été d'une grande aide en me soutenant et faisant preuve de patience, ce qui m'a permis de me concentrer sur ce projet et cette formation. Je tiens donc à les remercier également.

Enfin je tiens à remercier la région Ile-de-France et Pôle emploi pour le financement de cette formation.



SOMMAIRE :

I. Analyse et conception d'une application multicouche	7
A. Le cahier des charges.....	7
1. L'objectif du projet	7
2. Fonctionnalité de l'application	7
3. Exigences techniques	8
4. Contrainte de temps	8
5. Conclusion.....	8
B. Méthodologie et développement.....	8
C. Collaborer à la gestion du projet informatique et à l'organisation de l'environnement de développement.....	10
1. Logiciels utilisés.....	10
2. Utilisation de GitHub.....	10
3. Diagramme de Gantt	13
D. Diagramme de cas d'utilisation	13
E. Diagramme de séquence	15
F. Diagramme de classe	16
G. Création du projet Spring	17
1. Spring Initializr	17
2. Dependancies.....	18
3. Maven.....	19
II. Concevoir et développer des composants d'interface utilisateur	21
A. Maquetter une application	21
1. Logiciels Utilisés	21
2. Le zoning	21
3. Le Wireframe.....	24
4. Le mockup ou maquette graphique	24



B. Développer une interface utilisateur de type : Desktop	27
1. Le cahier des charges	27
2. Logiciel utilisé	27
3. Aperçu de l'application	28
4. Source et Design	28
C. Développer une interface utilisation de type : Mobile	31
1. Logiciels utilisés.....	31
2. Interface graphique Android Studio.....	31
3. Maquette graphique sur téléphone.....	32
III. Développement de l'application.....	35
A. Différence entre back-end et le front-end	35
B. Le back-end	36
1. Logiciel utilisé	36
2. Domain ou Entity	36
3. Repository.....	38
4. Controller	39
5. Dao	41
C. Le Front-end	43
1. Logiciels et framework.....	43
2. Qu'est-ce que Angular et à quoi sert-il ?.....	43
3. Création du projet Angular Shape	43
4. Création de component	45
5. Les models	46
6. Les services.....	47
7. Le HTML.....	48
8. Le CSS	50
9. Tableau des routes.....	50
10. Démarrage du serveur.....	51



IV. Concevoir et développer la persistance des données	54
A. Concevoir une base de données.....	54
1. Logiciel utilisé	54
2. Les étapes d'une base de données	55
3. Diagramme de base de données.....	55
B. Mise en place d'une base de données.....	58
1. Le fichier application.properties.....	58
2. ORM	58
3. La magie de JPA.....	60
4. Insertion des données	61
5. Développer les composants dans le langage d'une base de données	62
C. Développer des composants d'accès aux données	63
1. Design Pattern: Data Access Object (DAO).....	63
2. Quel est le but de ce design pattern ?	64
3. Création du package DAO.....	64
4. Présentation d'une classDao.....	65
V. Sécurité et mise en place.....	67
A. Pourquoi sécuriser une application ?.....	67
B. Comment sécuriser une application ?.....	68
1. Comment faire ?	68
2. Construire une application organisée en couches	69
3. Le MVC.....	69
C. Mise en place de la sécurité dans le back-end.....	71
1. Schéma de la mise en place.....	71
2. Autorisation et authentification	71
3. Le corsFilter	72
4. L'authentification et JwtToken	73
5. Security Configuration	75
6. BCryptPasswordEncoder.....	77
7. Les constantes - Autorisation suivant les rôles	79



8. Les énumérations et les rôles.....	80
D. Mise en place de la sécurité dans le front-end.....	81
1. AuthenticationGuard	81
2. Le service d'authentification	82
3. Component - Login.....	83
VI. Les tests.....	84
A. Tester le back-end.....	84
1. Logiciel utilisé	84
2. Test unitaire	84
B. Tester de front-end	85
C. Tester la base de données	85
1. Logiciel utilisé	85
2. Requête avec Postman	85
VII. Conclusion et évolution	87
A. Conclusion.....	87
1. Bilan du projet	87
2. Difficultés rencontrées	87
B. Evolution	87
1. Evolution de l'application.....	87
2. Perspective pour l'avenir	89



I. Analyse et conception d'une application multicouche

A. Le cahier des charges

Conception et développement d'une application de sport qui permet de proposer des exercices, d'importer des photos triées par date, de saisir des données correspondant aux mensurations des muscles, de créer un entraînement avec le choix du muscle, de l'exercice, des répétitions, séries, poids, temps et distance. Il sera possible de calculer l'IMC sur le site.

1. L'objectif du projet

- Développer une application de sport qui permettra aux utilisateurs de planifier et de suivre leurs entraînements de manière efficace.
- L'application permettra également de saisir des données correspondant aux mensurations des muscles.
- L'application permettra d'importer des photos triées par date.
- L'application permettra de calculer son IMC.

2. Fonctionnalité de l'application

- Proposition d'exercices : l'application proposera une bibliothèque d'exercices pour chaque groupe musculaire. Les exercices seront classés par niveau de difficulté et illustrés par des images et des descriptions.
- Importation de photos triées par date, l'application permettra aux utilisateurs d'importer des photos triées par date pour suivre leurs progressions physiques.
- Saisie de données correspondant aux mensurations des muscles. L'application permettra aux utilisateurs de saisir les mesures de chaque groupe musculaire, telles que la circonférence du bras, de la poitrine, etc.
- Création d'un entraînement personnalisé : l'application permettra aux utilisateurs de créer leurs propres programmes d'entraînement en choisissant le groupe musculaire, l'exercice, le nombre de répétitions, le poids et la durée.
- Calcul de l'IMC : l'application permettra aux utilisateurs de calculer leur indice de masse corporelle (IMC) à partir des données saisies.



3. Exigences techniques

- L'application devra fonctionner sur différents moteurs de recherche tels que Edge, Chrome ou Firefox.
- L'application sera développée en langage de programmation Java.
- L'application utilisera une base de données pour stocker les informations de l'utilisateur, y compris les données de mesures et de progression.
- L'application devra garantir la sécurité des données de l'utilisateur.

4. Contrainte de temps

Date de livraison : le projet devra être livré dans un délai de sept semaines.

5. Conclusion

En développant cette application, nous espérons aider les utilisateurs à suivre leurs progressions physiques et à atteindre leurs objectifs de fitness. L'application devra être conviviale, sûre et performante pour répondre aux besoins des utilisateurs.

B. Méthodologie et développement

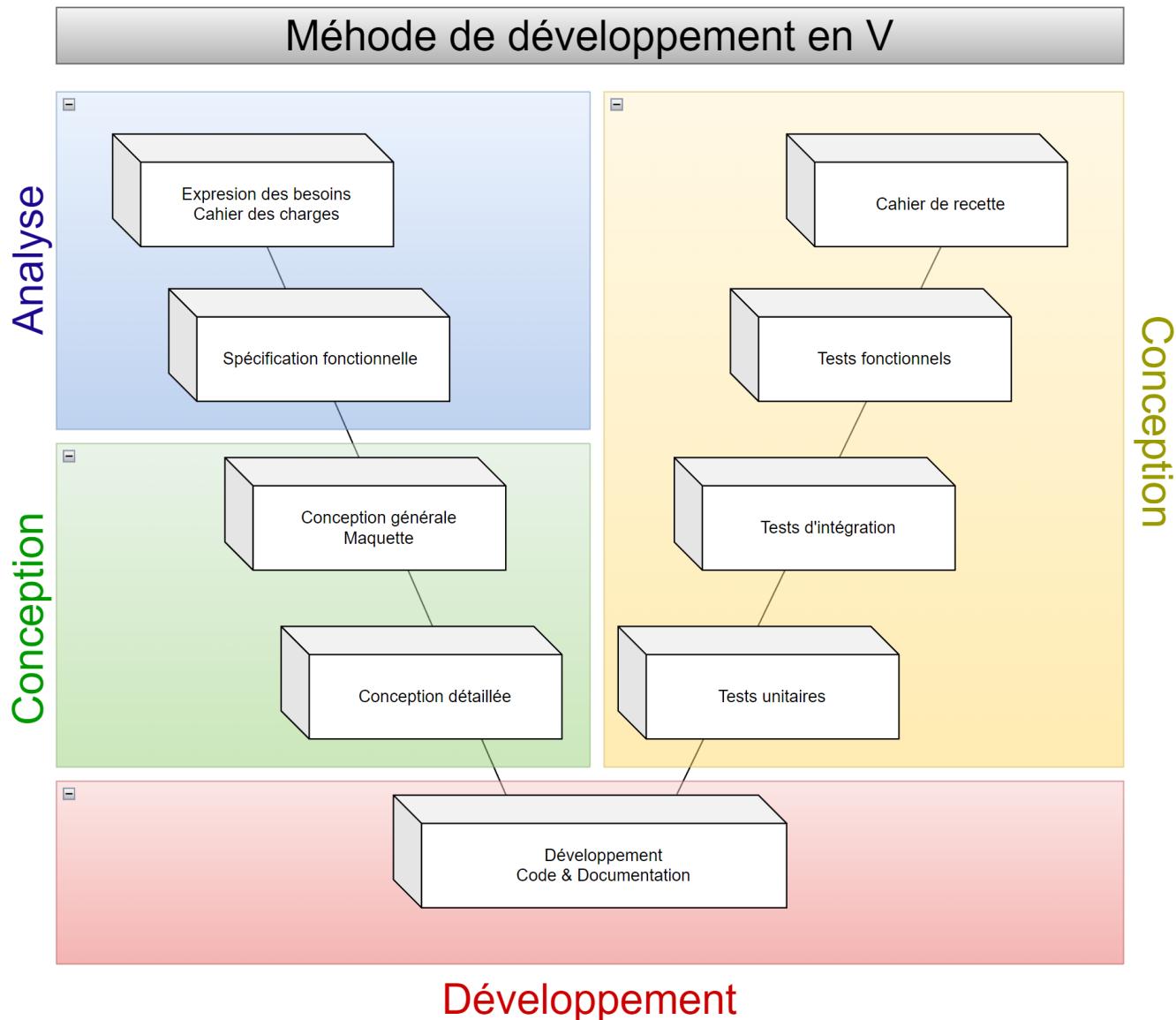
Il existe plusieurs méthodes de développement de logiciels, voici les principales :

- **La méthode en cascade** : les étapes du développement sont réalisées de manière séquentielle et linéaire. Les phases du projet sont clairement définies (analyse, conception, développement, tests, maintenance).
- **La méthode agile** : cette méthode est basée sur des cycles de développement courts et itératifs. Elle se concentre sur la collaboration entre les membres de l'équipe de développement et sur l'adaptabilité aux changements.
- **La méthode en V** : elle est similaire à la méthode en cascade, mais avec un accent particulier sur les tests et la validation. Elle met en correspondance les phases du développement avec les phases de tests correspondantes.
- **La méthode RAD (Rapid Application Development)** : elle vise à produire des résultats rapidement grâce à une forte implication des utilisateurs finaux. Elle repose sur des prototypes qui sont régulièrement présentés pour recueillir des commentaires et des améliorations.



- **La méthode DevOps** : elle implique une collaboration étroite entre les équipes de développement et d'opérations. Elle se concentre sur l'automatisation des processus de déploiement et de maintenance, ainsi que sur l'amélioration continue de la qualité du logiciel.

Pour le développement de l'application Shape, j'ai appliqué la méthode en V :





C. Collaborer à la gestion du projet informatique et à l'organisation de l'environnement de développement

1. Logiciels utilisés



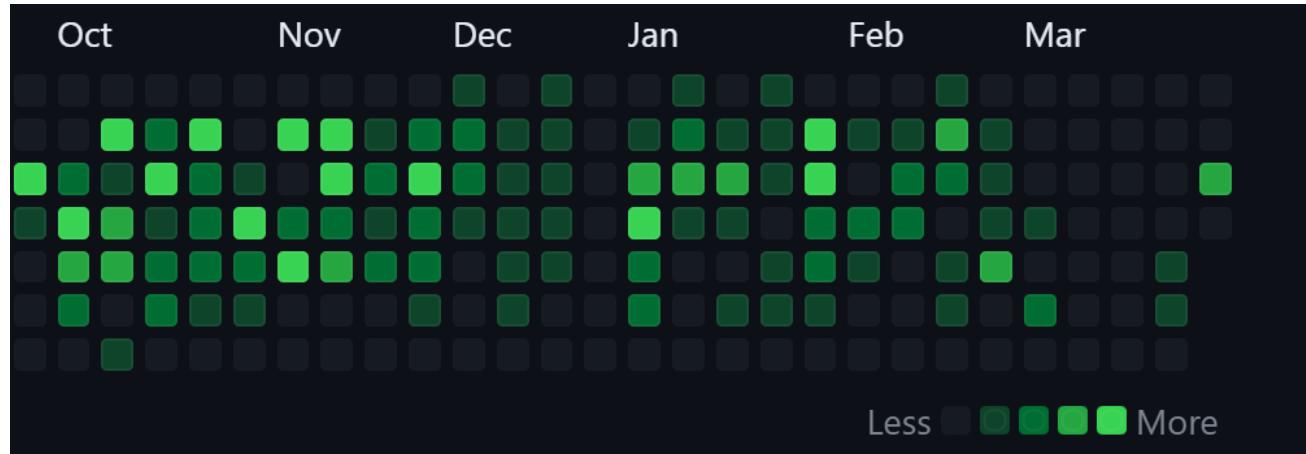
GitHub est une plateforme web de gestion de code source et de collaboration pour les projets de développement de logiciels. Elle permet aux développeurs de travailler ensemble sur un même code source en le stockant dans des dépôts (repositories) accessibles en ligne.



Excel est un logiciel de tableur développé par Microsoft qui permet de créer, modifier et analyser des feuilles de calcul contenant des données sous forme de tableaux. Excel permet également de créer des graphiques, des tableaux croisés dynamiques et d'appliquer des formules et des fonctions pour effectuer des calculs sur les données.

2. Utilisation de GitHub

- GitHub permet de télécharger les modifications locales enregistrées dans le dépôt local vers un dépôt distant sur GitHub. Il permet de suivre la contribution apportée au projet.



- Création d'un repository

Sur le site GitHub, il faut créer un nouveau repository. Le repository pourra stocker le code source.



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner *



AdHouen ▾

Repository name *

shapeApp



Great repository names are short and memorable. Need inspiration? How about [curly-dollop](#)?

Description (optional)

Public

Anyone on the internet can see this repository. You choose who can commit.

Private

You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

Add a README file

This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore

Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: **None** ▾

Choose a license

A license tells others what they can and can't do with your code. [Learn more.](#)

License: **MIT License** ▾

This will set **main** as the default branch. Change the default name in your [settings](#).

You are creating a public repository in your personal account.

Create repository



- Ensuite dans le dossier en local, il est nécessaire de « push » son code source.

git init : La commande est utilisée pour initialiser un nouveau dépôt Git dans un répertoire local. Cela crée un nouveau sous-répertoire .git dans le répertoire courant, qui contient tous les fichiers et répertoires nécessaires pour utiliser Git pour gérer les versions du projet.

git add : La commande est utilisée pour ajouter des fichiers à la zone de staging (zone de préparation) dans Git, pour qu'ils soient prêts à être inclus dans la prochaine validation (commit).

git commit -m « commentaire » : La commande est utilisée pour enregistrer les modifications apportées au projet dans Git. Cela crée un nouveau commit (ou une nouvelle version) du projet, qui enregistre toutes les modifications apportées depuis le dernier commit.

git push : La commande est utilisée pour pousser (envoyer) les modifications enregistrées localement dans le dépôt Git vers un dépôt distant sur GitHub.

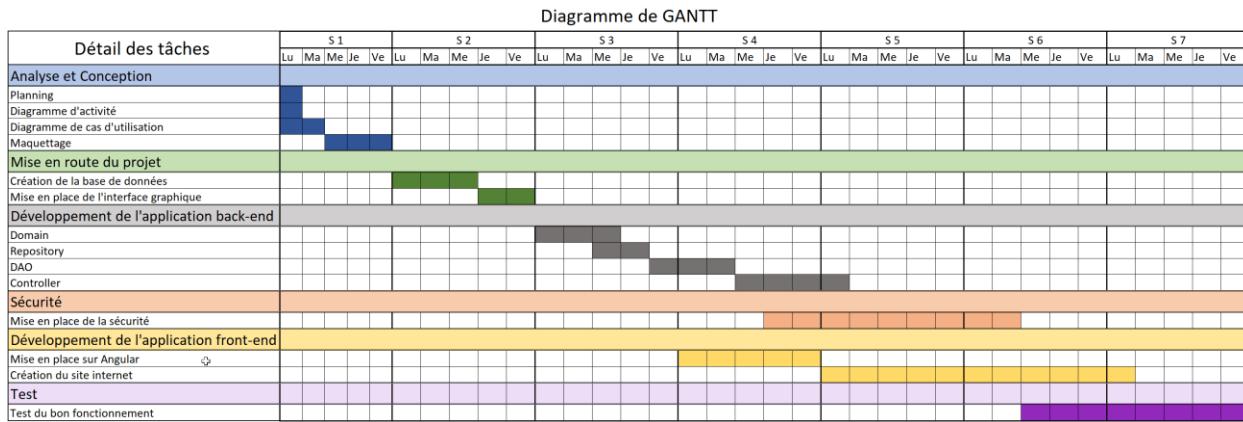
- Visualiser l'historique

The screenshot shows a GitHub repository history for the branch 'main'. The commit was made by 'AdHouen secu' yesterday. The commit message is 'secu'. The repository path is 'ShapeWithSecu / ShapeAppSpringWithSecu / src / main / java / com / shape / ShapeAppSpring /'. The commit details show a list of files and their status: configuration, constant, controller, dao, domain, enumeration, exception/domain, listener, repository, resource, service, utility, and ShapeAppSpringApplication.java, all marked as 'secu' and 'yesterday'. At the bottom, there is a 'Give feedback' button.



3. Diagramme de Gantt

Un diagramme de Gantt est un outil visuel de gestion de projet qui permet de représenter les tâches à accomplir, leurs durées, leurs ordres d'exécution et leurs avancements dans le temps.

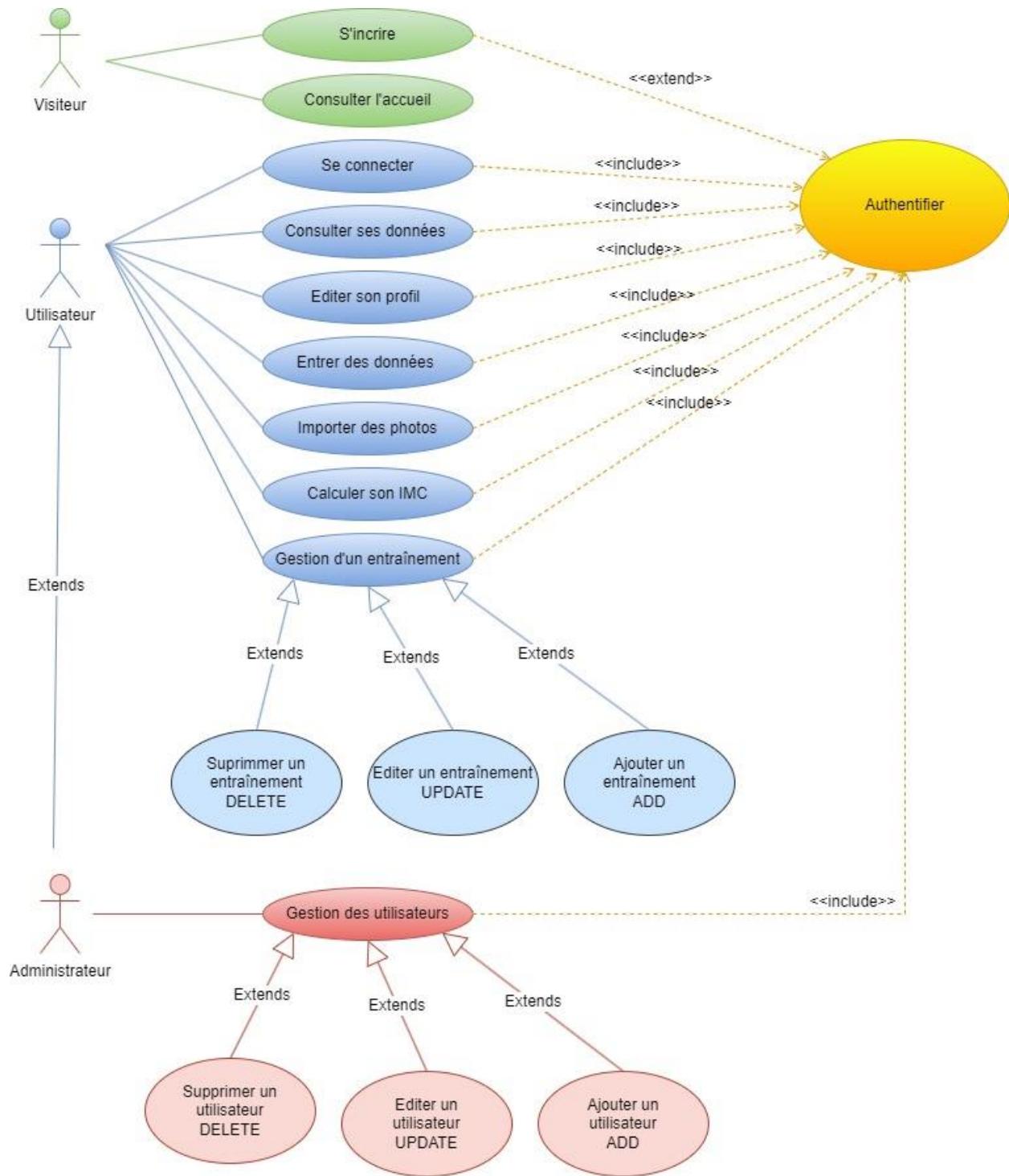


D. Diagramme de cas d'utilisation

Le diagramme de cas d'utilisation est un outil de modélisation qui permet de représenter graphiquement les interactions entre les acteurs (utilisateurs) et le système informatique. Il permet de décrire les différentes actions que les utilisateurs peuvent effectuer avec le système et les résultats attendus.



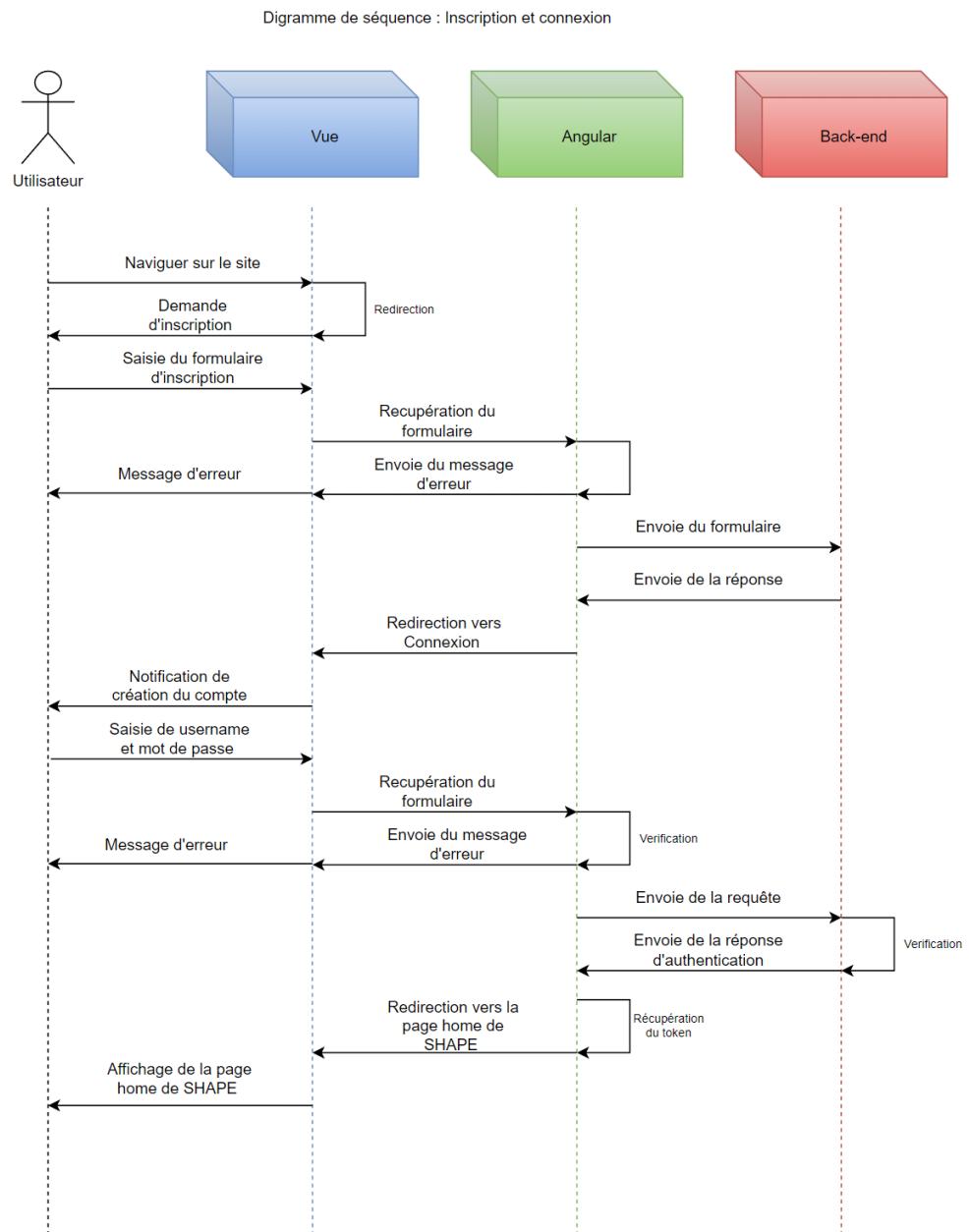
Voici le diagramme d'utilisation de l'application SHAPE sur le web :





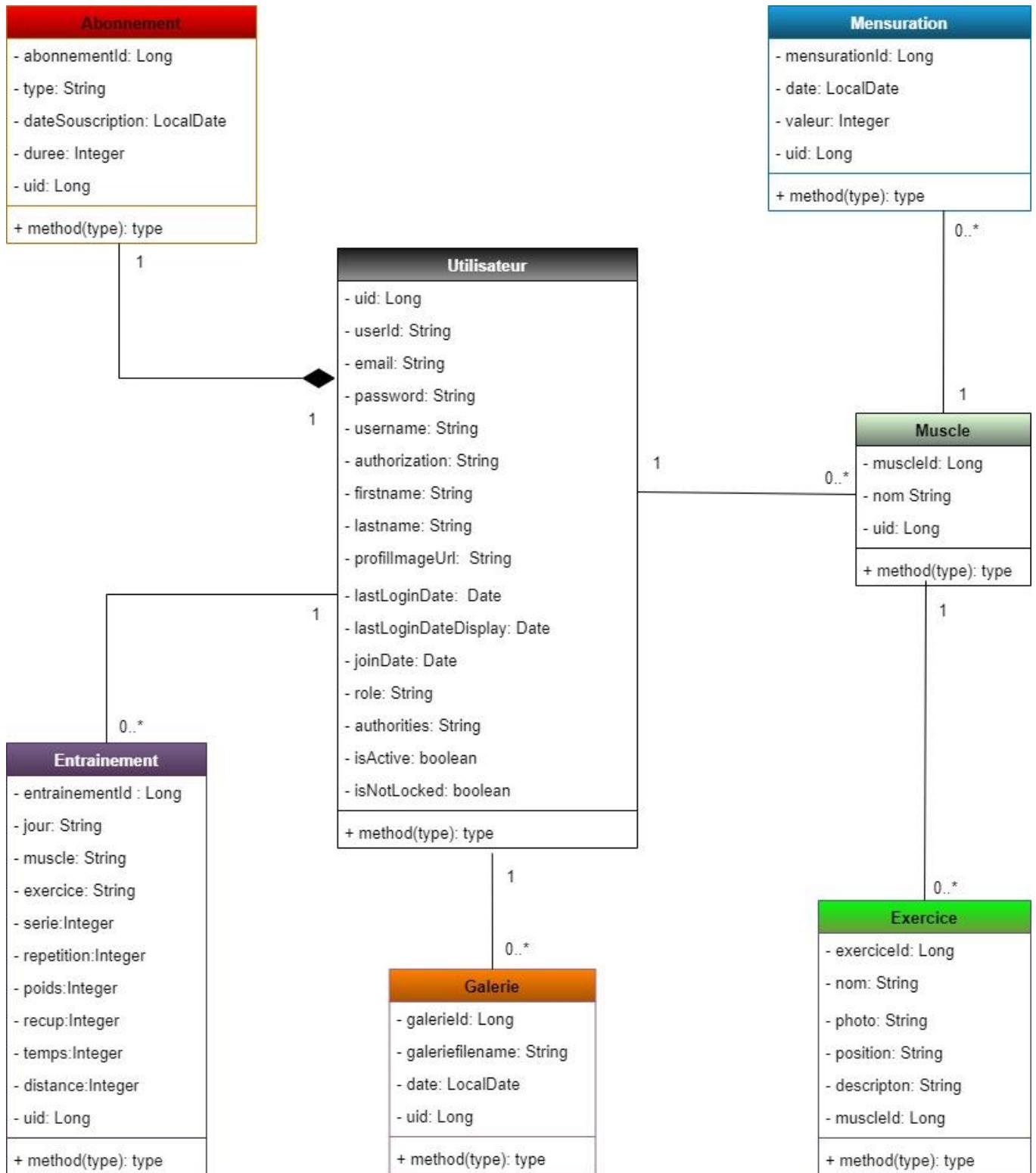
E. Diagramme de séquence

Le diagramme de séquence est un outil de modélisation qui permet de représenter graphiquement les interactions entre les différents objets ou composants d'un système informatique au fil du temps. Il met en évidence les messages échangés entre les objets, ainsi que l'ordre dans lequel ces messages sont envoyés et reçus.





F. Diagramme de classe





G. Création du projet Spring

1. Spring Initializr



Spring Initializr est un outil web qui permet de générer rapidement un squelette de projet Spring Boot avec les dépendances et configurations nécessaires pour commencer à développer une application.

Il permet de sélectionner différentes options telles que la version de Spring Boot, le langage de programmation, les dépendances, les plugins, etc. Une fois que les options sont sélectionnées, Spring Initializr génère un projet prêt à l'emploi avec une structure de fichiers initiale, des fichiers de configuration et des dépendances préconfigurées.

The screenshot shows the Spring Initializr interface with the following settings:

- Project:** Maven (selected)
- Language:** Java (selected)
- Spring Boot:** 3.1.0 (SNAPSHOT) (selected)
- Project Metadata:**
 - Group: com.Shape
 - Artifact: shape
 - Name: shape
 - Description: Project shape for Spring Boot
 - Package name: com.Shape.ShappApp
- Packaging:** Jar (selected)
- Java:** 17 (selected)



2. Dependancies

Dependencies ADD DEPENDENCIES... CTRL + B

Spring Data JPA SQL
Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

MySQL Driver SQL
MySQL JDBC driver.

Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Spring Security SECURITY
Highly customizable authentication and access-control framework for Spring applications.

Spring Boot DevTools DEVELOPER TOOLS
Provides fast application restarts, LiveReload, and configurations for enhanced development experience.

Lombok DEVELOPER TOOLS
Java annotation library which helps to reduce boilerplate code.

Les dependancies serviront à la création du pom.xml



3. Maven

Maven est un outil de gestion de projets de logiciels qui utilise une structure de projet standardisée et des fichiers de configuration XML appelés Project Object Model (POM). Un projet Maven est donc un projet logiciel construit à l'aide de l'outil Maven.

Le POM est le fichier de configuration principal de Maven, qui définit les dépendances du projet, les plugins à utiliser pour compiler, tester et empaqueter le code, les propriétés du projet et autres informations importantes pour la construction du projet.

Maven fournit également des fonctionnalités pour automatiser les tâches récurrentes telles que la compilation, le test, la construction et le déploiement de projets. Cela facilite la collaboration entre les développeurs et améliore la qualité du code.

Fichier pom.xml de l'application Shape :

```
<!-- Cette dépendance fournit des fonctionnalités de persistance de données pour Spring Boot en utilisant JPA (Java Persistence API). Elle inclut des bibliothèques telles que Hibernate, qui permettent la gestion de la persistance des données.-->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<!-- Cette dépendance fournit des fonctionnalités de création de services Web RESTful en utilisant Jersey, qui est un framework de développement pour les services Web RESTful.-->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-jersey</artifactId>
</dependency>
<!-- Cette dépendance fournit des fonctionnalités de sécurité pour les applications Spring Boot. Elle inclut des bibliothèques telles que Spring Security, qui permettent de gérer la sécurité de l'application.-->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
</dependency>
<!-- Cette dépendance fournit un moteur de template pour les applications Spring Boot. Elle inclut Thymeleaf, qui est un moteur de template pour les applications Web basées sur Java.-->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>
<!-- Cette dépendance fournit des fonctionnalités pour le développement d'applications Web en utilisant Spring Boot. Elle inclut des bibliothèques telles que Spring MVC, qui permettent de développer des applications Web.-->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
```



```
<!-- Cette dépendance fournit des bibliothèques pour la gestion du cache en Java.-->
<dependency>
    <groupId>com.google.guava</groupId>
    <artifactId>guava</artifactId>
    <version>28.1-jre</version>
</dependency>
<!-- Cette dépendance fournit des fonctionnalités pour la création de services RESTful en utilisant Spring Boot.-->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-rest</artifactId>
</dependency>
<!-- Cette dépendance fournit des outils pour le développement des applications Spring Boot.
Elle inclut des fonctionnalités telles que le rechargement automatique des modifications, la configuration automatique, etc.-->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <scope>runtime</scope>
    <optional>true</optional>
</dependency>
<!-- Cette dépendance fournit des fonctionnalités de connexion à une base de données MySQL.-->
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <scope>runtime</scope>
</dependency>
<!-- Cette dépendance fournit des fonctionnalités pour réduire le code répétitif et éviter les erreurs courantes.
Elle inclut des annotations telles que @Getter, @Setter, @NoArgsConstructor, etc.-->
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <optional>true</optional>
</dependency>
<!-- Cette dépendance fournit des fonctionnalités pour la validation des données en utilisant Spring Boot.-->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-validation</artifactId>
</dependency>
<!-- Cette dépendance fournit des fonctionnalités pour la sérialisation/désérialisation de données en utilisant Jackson et Hibernate.-->
<dependency>
    <groupId>com.fasterxml.jackson.datatype</groupId>
    <artifactId>jackson-datatype-hibernate5</artifactId>
</dependency>
```

```
<!-- Cette dépendance est une bibliothèque Java qui permet de réduire la quantité de code boilerplate ou
répétitif nécessaire pour les classes Java, tels que les getters/setters,
les constructeurs, les méthodes equals/hashcode/toString, etc. -->
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <optional>true</optional>
</dependency>
<!-- Cette dépendance fournit des fonctionnalités pour les tests unitaires en utilisant Spring Boot.-->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
</dependency>
<!-- Cette dépendance fournit des fonctionnalités pour les tests de sécurité en utilisant Spring Security.-->
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-test</artifactId>
    <scope>test</scope>
</dependency>
<!-- Cette dépendance fournit des fonctionnalités pour la création et la validation de JSON Web Tokens (JWT) en Java.-->
<dependency>
    <groupId>com.auth0</groupId>
    <artifactId>java-jwt</artifactId>
    <version>3.19.1</version>
</dependency>
```



II. Concevoir et développer des composants d'interface utilisateur

A. Maquetter une application

1. Logiciels Utilisés

Les outils de conception utilisés sont :



Diagrams.net ou draw.io permet de créer des diagrammes et des organigrammes personnalisables.



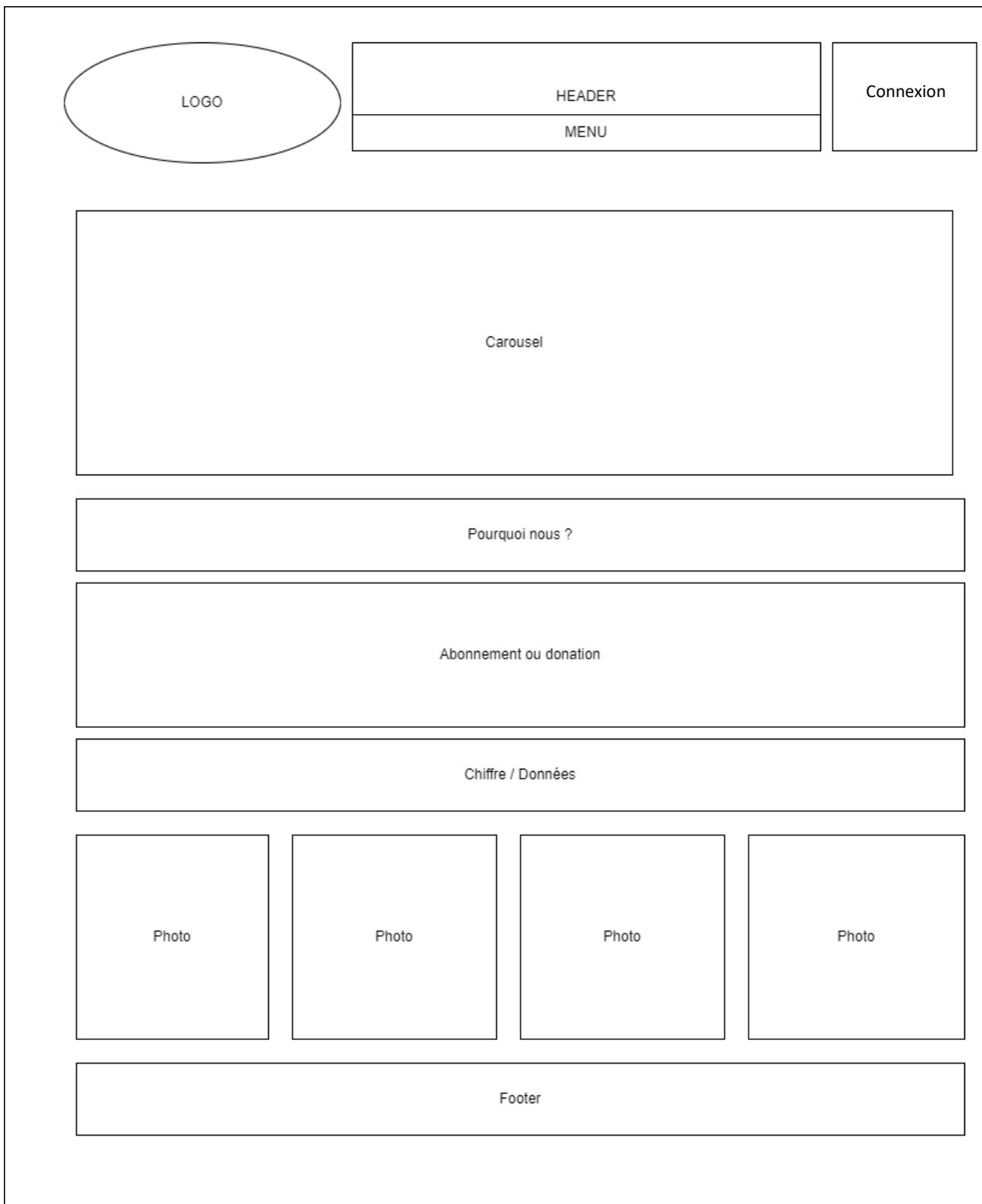
Figma est une plateforme collaborative pour éditer des graphiques vectoriels et faire du prototypage. Elle permet de concevoir des design systems pour faciliter la création de sites web et d'applications mobiles. C'est une solution à destination des UI et UX designers et des développeurs.

2. Le zoning

Le zoning est une schématisation grossière de ce que sera la future page web. On utilise des blocs pour déterminer où se trouveront les contenus et fonctionnalités.



Zoning : accueil





Zoning : galerie





3. Le Wireframe

Le wireframe (on parle de « maquette fil de fer » en français) est la suite logique du zoning. Chaque bloc réalisé lors de l'étape précédente se voit doté d'image(s), de texte(s) ou de vidéo(s). Ce contenu peut être fictif car les informations finales ne sont pas toujours connues à ce stade du projet. L'objectif est de définir l'organisation des éléments et des formes sans travailler l'aspect visuel, le graphisme n'interviendra que plus tard. On se base davantage sur les standards et souhaits ergonomiques pour orienter la réflexion.

4. Le mockup ou maquette graphique

Maquette graphique : Calcul IMC

The screenshot shows the Shape website's IMC calculator page. At the top, there is a header with the Shape logo, contact information (shape@gmail.com, +33 (0)29 39 05 26), and social media links. Below the header, there is a navigation menu with links to ACCUEIL, PLANNING, GALERIE, MENSURATION+, EXERCICE, IMC, ADMIN, and CONTACT. A green button labeled "COMMENCER" is prominently displayed. To the right of the button is a "SE CONNECTER" link. The main content area is titled "Calcul ton IMC". It contains instructions about what IMC is and how it is calculated. There are input fields for "Taille en cm" (181) and "Poids en kg" (80). A "VALIDER" button is located below these fields. The calculated result is shown as "IMC: 24.42 En bonne santé" with a color-coded scale from green to red. Below this, there is a section titled "Qu'est ce que l'indice de masse corporelle ? (IMC)" with a detailed explanation of its calculation. A table titled "IMC Résultat:" provides a breakdown of the IMC range and its corresponding health status:

IMC	ETAT DU POIDS
Inférieur à 18,5	Gros perteur
18,5 - 24,9	En bonne santé
25,0 - 29,9	Surcharge pondérale
30,0 - et plus	Obèse

At the bottom left, there is a "CONTACT" section with address, email, and phone number. On the right, there is a "NEWSLETTER" section with a "INSCRIRE VOUS POUR RECEVOIR LA NEWSLETTER" button and a "Your Email" input field.



Maquette graphique : galerie

The screenshot shows the Shape application's gallery page. At the top, there is a header with the Shape logo, contact information (shape@gmail.com, +33 (6) 29 39 05 26), and navigation links (ACCUEIL, PLANNING, GALERIE, MENSURATION+, EXERCICE, IMC, ADMIN+, CONTACT). Below the header, there is a "CHoisir une Image" (Select an image) button and a "VALIDER" (Validate) button. The main content area displays a grid of user profiles. The first row is dated "2023-04-08" and contains two profile pictures. The second row is dated "2023-04-20" and contains four profile pictures. Each profile picture has a small red trash icon in the bottom right corner. In the bottom right corner of the page, there is a "NEWSLETTER" section with a "INSCRIRE VOUS POUR RECEVOIR LA NEWSLETTER" (Subscribe to receive the newsletter) heading, a "Newsletter" input field, and a "S'INScrire" (Subscribe) button.

Maquette graphique : contact

The screenshot shows the Shape application's contact page. At the top, there is a header with the Shape logo, contact information (shape@gmail.com, +33 (6) 29 39 05 26), and navigation links (ACCUEIL, PLANNING, GALERIE, MENSURATION+, EXERCICE, IMC, ADMIN+, CONTACT). Below the header, there is a "CONTACTEZ-NOUS" (Contact us) section titled "GET IN TOUCH". It includes a location pin pointing to "83, rue de Paris 93260 Les Lilas", a phone icon with "+France +33 (6) 29 39 05 26", and an email icon with "shape@gmail.com". To the right of the contact information is a "Formulaire" (Form) with fields for "Nom" (Name), "Email", "Website", and "Comment" (Comment). A "SOUMETTRE" (Submit) button is located below the form. Below the contact form is a map showing the location of "83 Rue de Paris" in Les Lilas, with various landmarks and streets labeled. In the bottom right corner, there is a "NEWSLETTER" section with a "INSCRIRE VOUS POUR RECEVOIR LA NEWSLETTER" (Subscribe to receive the newsletter) heading, a "Newsletter" input field, and a "S'INScrire" (Subscribe) button. The footer of the page includes the copyright notice "© Shape by Adrien Houen. All Rights Reserved." and the contact information "83, rue de Paris 93260 Les Lilas", "shape@gmail.com", and "+33 (6) 29 39 05 26".



Maquette graphique : ajouter un entraînement

The screenshot shows the Shape application's workout creation interface. At the top, there are navigation links: ACCUEIL, PLANNING, GALERIE, MENSURATION, EXERCICE, IWC, ADMIN, and CONTACT. On the right, there are social media sharing icons and a 'SE CONNECTER' button. The main title is 'Ajouter un entraînement'. Below it, a day selector shows 'Lundi' as the active day. A sidebar on the left lists muscle groups: Épaules, Biceps, Triceps, Pectoraux, Dos, Abdominaux, Fessiers, Quadriceps, Ischio-jambiers, and Mollets. The central area displays a list of exercises under the 'Épaules' group, with 'Développé militaire' selected. To the right of the list are input fields for 'Série', 'Répétition', 'Poids', 'Temps', and 'Distance'. A 'Sauvegarder' (Save) button is at the bottom. At the bottom of the page, there is a 'CONTACT' section with address and contact information, and a 'NEWSLETTER' sign-up form.

Maquette graphique : éditer un entraînement

The screenshot shows the Shape application's workout editor interface. The top navigation and sidebar are identical to the previous screenshot. The main title is 'Editor : Entraînement'. A modal window titled 'Lundi' contains a table with three rows of exercise data: Développé militaire, Face pull, and Pec deck inversé. Each row has 'Edit' and 'Remove' buttons. Below the modal, a table lists all exercises for the day, with columns for Muscle, Exercice, Série, Répétition, Poids, Réception, Temps, Distance, Edit, and Remove. The 'CONTACT' and 'NEWSLETTER' sections are also present at the bottom.



B. Développer une interface utilisateur de type : Desktop

1. Le cahier des charges

Application avec un cahier des charges :

« Ecrire un logiciel qui permettra, à terme, de rechercher des mots dans un texte. La zone de texte, à droite, affiche de contenu d'un fichier texte constant dont le nom est visualisé dans la partie basse, en rouge. La partie gauche est une liste contenant, par ordre alphabétique, les mots du texte en un seul exemplaire. La partie supérieure contient un label et un bouton sans effet. »

2. Logiciel utilisé



Eclipse est un IDE, Integrated Development Environment (EDI environnement de développement intégré en français), c'est-à-dire un logiciel qui simplifie la programmation en proposant un certain nombre de raccourcis et d'aide à la programmation.



WindowBuilder

Le plugin Eclipse WindowBuilder est un concepteur Java GUI visuel, puissant et facile à utiliser permettant la création d'applications GUI

Java sans écrire du code pour afficher des objets graphiques simples comme fenêtres, boutons de commandes, champs de textes... Avec ce plugin, nous pouvons créer des fenêtres compliquées en quelques minutes, il suffit d'utiliser le concepteur visuel et le code Java sera automatiquement généré.



3. Aperçu de l'application

The screenshot shows a Windows application window with a teal header bar containing a logo, a search bar labeled "Recherche du mot" with a "Chercher" button, and a scrollable text area. The text area contains a story about Shadoks and their rocket. A vertical list of words from the story is displayed on the left side of the text area, with the word "fusée" highlighted in blue. The status bar at the bottom shows the path "C:/INTERNE/Github/JAVA_SWING_SQL/TP304_Swing_Evenement/classes/shadock.txt".

Pourquoi les Shadoks pompaient-ils ?

Pour aller sur la Terre, les Gibis ont construit une fusée. Elle fonctionne grâce à un combustible super-puissant, le COSMOGOL 999.

Mais quel est ce combustible miracle ? D'où les Gibis l'extraient-ils ?

Comme vous le savez, la planète Gibi est plate. Dès que l'on creuse un trou, on tombe de l'autre côté. Les Gibis ne peuvent donc rien extraire du sous-sol, sans cela leur planète serait une vraie passoire.

Les Gibis extraient le Cosmogol 999 de l'atmosphère. Celle-ci est recueillie dans d'énormes entonnoirs à atmosphère, puis pompée dans des pompes à atmosphère. Par un procédé secret, les Gibis en extraient le Cosmogol brut, dit Cosmogol de première pression qui, une fois raffiné, concentré et enrichi, donne le fameux Cosmogol 999 qui propulsera la fusée Gibi.

Sous la direction du Professeur SHADOKO, les Shadoks ont construit une fusée interplanétaire.

Elle était munie des derniers perfectionnements techniques tels que casseroles à retardement, batteurs d'air et tire-bouchons hypersustentateurs. La seule difficulté, c'est qu'elle ne marchait que d'un côté: de haut en bas uniquement, car les Shadoks n'avaient pas de carburant suffisamment puissant.

Alors, pour subtiliser celui des Gibis, le Cosmogol 999, le Professeur Shadoko avait mis au point un plan. Nous avons demandé au Professeur lui-même de nous dire deux mots de ce plan :

"C'est très simple. Pour que le Cosmogol Gibi vienne chez nous, il suffit que nous le POMPIONS à travers le cosmos. Et nous le pompons grâce à cette COSMOPOMPE de mon invention d'une puissance incroyable de trois millions de Shadoks-Vapeur."

deux
difficulté
dire
direction
dit
donc
donne
du
dès
elle
en
enrichi
entonnoirs
est
et
extraient
extraire
fameux
fois
fonctionne
fusée
gibi
gibis
grâce
haut
hypersustentateurs
il
incroyable

C:/INTERNE/Github/JAVA_SWING_SQL/TP304_Swing_Evenement/classes/shadock.txt

4. Source et Design

Grace à l'installation de Windows Builder nous avons accès à deux parties :



La partie source :

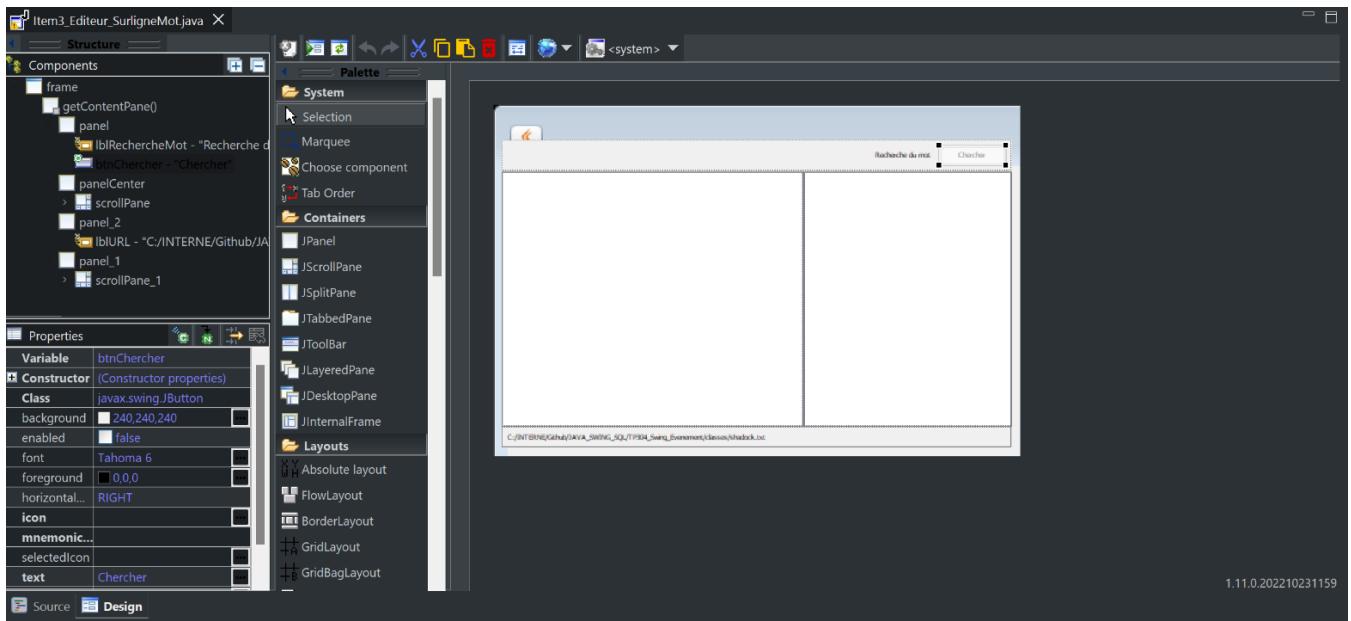
```
1 public class Item3_Editeur_SurligneMot {
2     private JFrame frame;
3     private JList list;
4     private JList list_1;
5     private JButton btnChercher;
6     private String texteAffiche = "";
7     private String texteDroite = "";
8     private JTextArea txtDansLePlancher;
9
10    /**
11     * Launch the application.
12     */
13    public static void main(String[] args) {
14        EventQueue.invokeLater(new Runnable() {
15            public void run() {
16                try {
17                    Item3_Editeur_SurligneMot window = new Item3_Editeur_SurligneMot();
18                    window.frame.setVisible(true);
19                } catch (Exception e) {
20                    e.printStackTrace();
21                }
22            }
23        });
24    }
25
26    /**
27     * Create the application.
28     */
29    public Item3_Editeur_SurligneMot() {
30        initialize();
31    }
32
33    /**
34     * Initialize the contents of the frame.
35     */
36    private void initialize() {
37        frame = new JFrame();
38        frame.setBounds(100, 100, 450, 300);
39        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
40        frame.getContentPane().setLayout(new BorderLayout(0, 0));
41
42        JPanel panel = new JPanel();
43        frame.getContentPane().add(panel, BorderLayout.NORTH);
44        panel.setLayout(new FlowLayout(FlowLayout.RIGHT, 5, 5));
45
46        JLabel lblRechercheMot = new JLabel("Recherche du mot");
47        lblRechercheMot.setHorizontalTextPosition(SwingConstants.RIGHT);
48        panel.add(lblRechercheMot);
49
50        btnChercher = new JButton("Chercher");
51        btnChercher.setEnabled(false);
52        btnChercher.setHorizontalTextPosition(SwingConstants.RIGHT);
53        panel.add(btnChercher);
54
55        JPanel panelCenter = new JPanel();
56        frame.getContentPane().add(panelCenter, BorderLayout.CENTER);
57        panelCenter.setLayout(new BoxLayout(panelCenter, BoxLayout.X_AXIS));
58
59        JScrollPane scrollPane = new JScrollPane();
60        panelCenter.add(scrollPane);
61
62        JPanel panel_2 = new JPanel();
63        FlowLayout flowLayout = (FlowLayout) panel_2.getLayout();
64        flowLayout.setAlignment(FlowLayout.LEFT);
65        frame.getContentPane().add(panel_2, BorderLayout.SOUTH);
66
67        JLabel lblURL = new JLabel("New label");
68        panel_2.add(lblURL);
69
70        txtDansLePlancher = new JTextArea();
71        String fichier = "shadock.txt";
72        String way = getClass().getClassLoader().getResource(fichier).getPath().replaceAll("/", "\\");
73        Path path = Paths.get(way);
74
75        try (BufferedReader r = Files.newBufferedReader(path, StandardCharsets.UTF_8)) {
76            String line = null;
77            try {
78                while ((line = r.readLine()) != null) {
79                    texteDroite += line;
80                    texteAffiche += line + "\n";
81                }
82            } finally {
83                r.close();
84            }
85            /* Texte du label URL (si OK) */
86            lblURL.setForeground(Color.black);
87            lblURL.setText(way);
88        } catch (NoSuchElementException e) {
89            /* Texte du label URL (si NOK) */
90            lblURL.setForeground(Color.red);
91            lblURL.setText("Fichier Inconnu");
92        } catch (IOException e) {
93            System.out.println(e.getMessage());
94        }
95    }
96}
```



```
215     private void surlignerBouton(ActionEvent e) {
216         Highlighter highlighter = txtrDansLePlancher.getHighlighter();
217         highlighter.removeAllHighlights();
218         getHighlight((String) list_1.getSelectedValue());
219     }
220
221     private void surlignerSouris(MouseEvent e) {
222         Highlighter highlighter = txtrDansLePlancher.getHighlighter();
223         highlighter.removeAllHighlights();
224         getHighlight((String) list_1.getSelectedValue());
225     }
226 }
227
228     private void getHighlight(String mot) {
229         Pattern p = Pattern.compile("(?i)\\b" + mot + "\\b");
230         Matcher m = p.matcher(texteAffiche);
231         while (m.find()) {
232             if (!m.group().isEmpty()) {
233                 Highlighter highlighter = txtrDansLePlancher.getHighlighter();
234                 HighlightPainter painter = new DefaultHighlighter.DefaultHighlightPainter(Color.pink);
235                 try {
236                     highlighter.addHighlight(m.start(), m.end(), painter);
237                 } catch (BadLocationException e) {
238                     e.printStackTrace();
239                 }
240             }
241         }
242     }
243
244 } /* getHighlight( mot ) */
245
246 /* Item3_Editeur_SurligneMot */
```

Source Design

Et la partie Design :





C. Développer une interface utilisation de type : Mobile

1. Logiciels utilisés

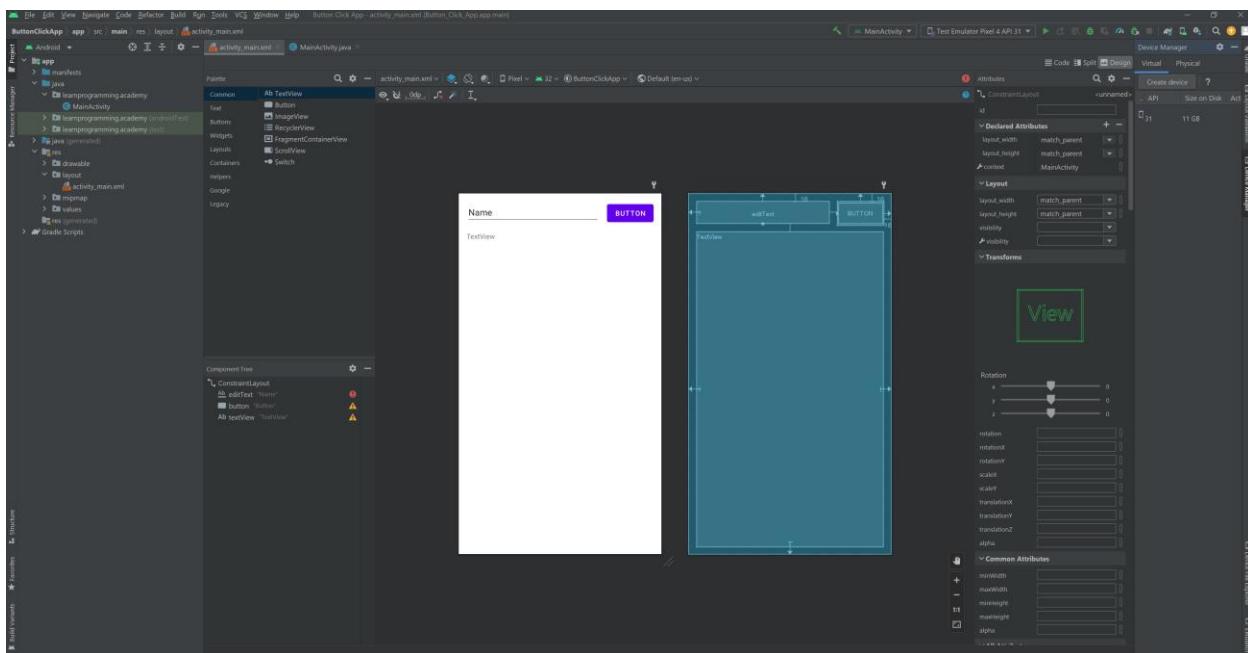


Figma est un outil de conception d'interface utilisateur (UI) et d'expérience utilisateur (UX) basé sur le cloud. Il permet aux designers de créer des maquettes d'interface utilisateur interactives, des prototypes, des designs graphiques et des spécifications de conception pour les applications web, mobiles et de bureau.



Android Studio : Android Studio est un environnement de développement intégré (IDE) créé par Google pour le développement d'applications mobiles Android. Il est utilisé pour créer, éditer, déboguer, tester et publier des applications Android.

2. Interface graphique Android Studio





3. Maquette graphique sur téléphone

The mobile screen displays the Shape app's landing page. At the top, there's a navigation bar with the word "SHAPE" in white and a menu icon. Below the header is a dark banner featuring three people in a gym setting. The text "WELCOME" and "SHAPE YOUR BODY" are prominently displayed, along with "JOIN US" and "CONTACT US" buttons. A large section below is titled "POURQUOI UTILISER SHAPE ?" with the sub-section "POUSSE TES LIMITES PLUS LOIN". It includes two cards: "Suis ton évolution" (with a bar chart icon) and "Calcul ton BMI" (with a scale and BMI calculator icon). A small paragraph of placeholder text follows each card.

The mobile screen shows the "LISTE DES EXERCICES" (Exercise List) page. The title "CHOISIS TON MUSCLE" is centered. Below the title are several categories of exercises: ÉPAPULES (highlighted in blue), BICEPS, TRICEPS, PECTORAUX, DOS, ABDOMINAUX, FESSIERS, QUADRICEPS, ISCHIO-JAMBIERS, and MOLLETS. At the bottom, there's a table with columns for Date, Mesure, Evolution, Edit, and Remove, containing three rows of data.

Date	Mesure	Evolution	Edit	Remove
2023-04-07	55	Avoir		
2023-04-05	52	Avoir		
2023-04-02	50	Avoir		



SHAPE

Calcul ton IMC

IMC : L'indice de Masse Corporelle, plus communément appelé IMC, est calculé à partir de la taille et du poids selon la formule suivante : $IMC = \text{poids en kg} / \text{taille}^2$. Il permet d'évaluer le statut pondéral.

Taille en cm
181

Poids en kg
81

VALIDER

IMC: 24.72 En bonne santé



Qu'est ce que l'indice de masse corporelle ? (IMC)

IMC : L'indice de Masse Corporelle, plus communément appelé IMC, est calculé à partir de la taille et du poids selon la formule suivante : $IMC = \text{poids en kg} / \text{taille}^2$. Il permet d'évaluer le statut pondéral.

IMC Résultat :

IMC	ÉTAT DU POIDS
Inférieur à 18,5	Sous-pondération
18,5 - 24,9	En bonne santé
25,0 - 29,9	Surcharge pondérale
30,0 - et plus	Obèse

CONTACT

📍 83, rue de paris 93260 Les Lilas
✉️ shape@gmail.com
📞 +33 (6) 29 39 05 26

NEWSLETTER

INSCRIVEZ VOUS POUR RECEVOIR LA NEWSLETTER

Inscrivez vous dès maintenant pour recevoir prochainement notre newsletter. Des recettes adaptées ainsi que des exercices personnalisés

Your Email

S'INSCRIRE

© Shape by Adrien Houen. All Rights Reserved.

SHAPE

CONTACTEZ-NOUS

GET IN TOUCH

📍 83, rue de Paris
93260
Les Lilas

📍 France
+33 (6) 29 39 05 26

✉️ shape@gmail.com

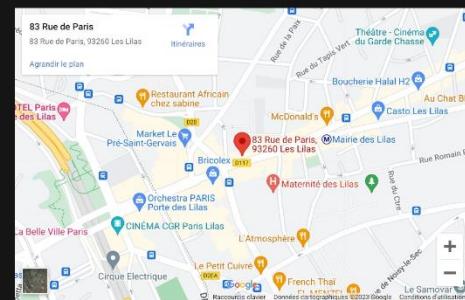
Name

Email

Website

Comment

SOUMETTRE



CONTACT

📍 83, rue de paris 93260 Les Lilas
✉️ shape@gmail.com
📞 +33 (6) 29 39 05 26

NEWSLETTER

INSCRIVEZ VOUS POUR RECEVOIR LA NEWSLETTER

Inscrivez vous dès maintenant pour recevoir prochainement notre newsletter. Des recettes adaptées ainsi que des exercices personnalisés

Your Email **S'INSCRIRE**

© Shape by Adrien Houen. All Rights Reserved.



SHAPE

Ajouter un entraînement

Lundi Mardi Mercredi Jeudi Vendredi
Samedi Dimanche

- Épaules**
- Biceps
 - Triceps
 - Pectoraux
 - Dos
 - Abdominaux
 - Fessiers
 - Quadriceps
 - Ischio-jambiers
 - Mollets

- Développé militaire
- Développé Arnold
- Face pull
- Élévations latérales à la poule
- Élévations latérales à la machine
- Développé épaules avec haltères
- Développé épaules debout à la landmine
- Rotation externe de l'épaule à la poule
- Pec deck inversé**
- Élévations frontales
- Élévations latérales
- Presse à épaules inclinée

Série

Répétition

Poids

Temps

Distance

SOUMETTRE

CONTACT

📍 83, rue de paris 93260 Les Lilas
✉️ shape@gmail.com
📞 +33 (6) 29 39 05 26

NEWSLETTER

INSCRIVEZ VOUS POUR RECEVOIR LA NEWSLETTER

Inscrivez vous dès maintenant pour recevoir prochainement notre newsletter. Des recettes adaptées ainsi que des exercices personnalisés

Your Email

S'INSCRIRE

© Shape by Adrien Houen. All Rights Reserved.

SHAPE

TON PLANNING

CHOISIS TON ENTRAINEMENT

AJOUTER UN ENTRAINEMENT

LUNDI MARDI MERCRIDI JEUDI VENDREDI SAMEDI DIMANCHE

Lundi

Muscle	Exercice	Série	Répétition	Poids	Récupération	Temps	Distance	Edit	Remove
Épaules	Développé militaire	12	12	12 kg	sec				
Épaules	Face pull	12	13	12 kg	sec				
Épaules	Pec deck inversé	12	12	35 kg	sec				

CONTACT

📍 83, rue de paris 93260 Les Lilas
✉️ shape@gmail.com
📞 +33 (6) 29 39 05 26

NEWSLETTER

INSCRIVEZ VOUS POUR RECEVOIR LA NEWSLETTER

Inscrivez vous dès maintenant pour recevoir prochainement notre newsletter. Des recettes adaptées ainsi que des exercices personnalisés

Your Email

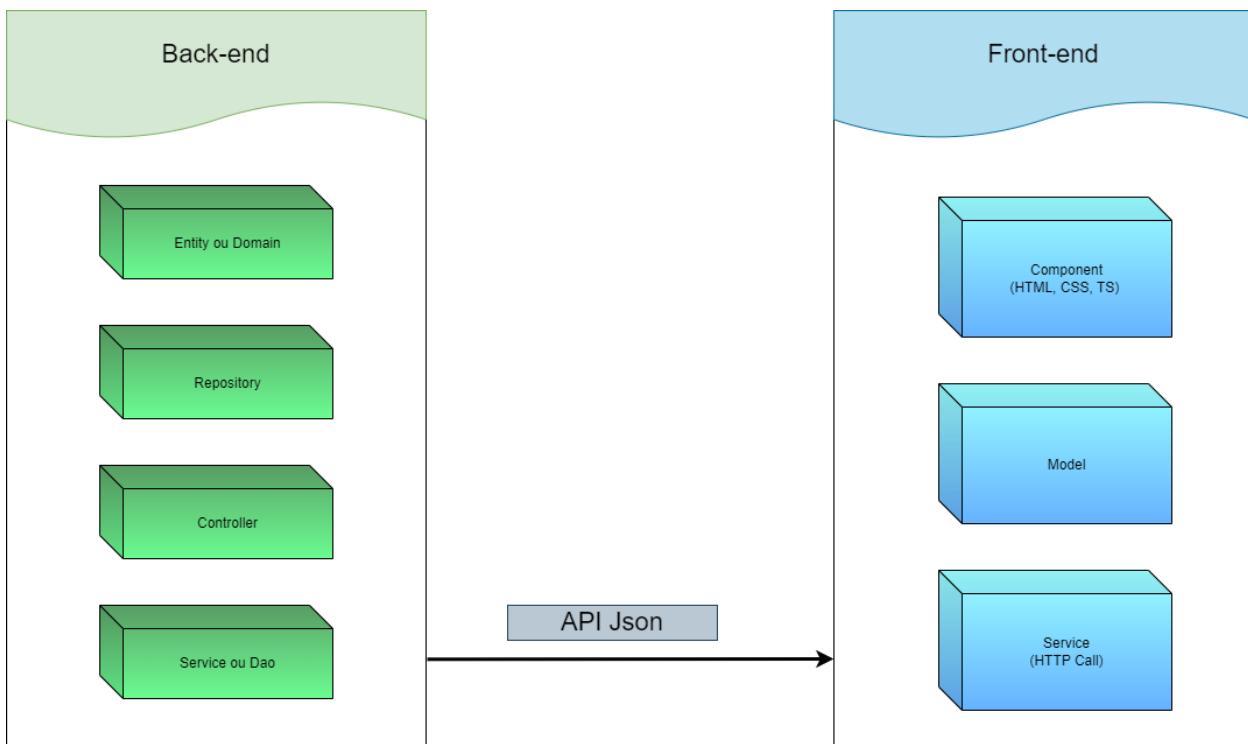
S'INSCRIRE

© Shape by Adrien Houen. All Rights Reserved.



III. Développement de l'application

A. Différence entre back-end et le front-end



Le front-end et le back-end sont les deux parties essentielles d'un site web ou d'une application web.

Le back-end, également appelé "partie serveur", se réfère à la partie invisible d'un site web ou d'une application web qui traite les requêtes de l'utilisateur, récupère et stocke les données, et génère les réponses pour le front-end. Le back-end est généralement développé avec des langages de programmation tels que PHP, Python, Java et utilise des bases de données pour stocker les informations.

Le front-end, également appelé "partie client", fait référence à la partie visible d'un site web ou d'une application web, c'est-à-dire tout ce que l'utilisateur voit et avec quoi il interagit. Cela inclut l'interface utilisateur, le design, le contenu et les fonctionnalités telles que les boutons, les formulaires et les animations. Les technologies couramment utilisées pour développer le front-end sont HTML, CSS et JavaScript.



En somme, le front-end est responsable de l'aspect visuel et interactif de l'application, tandis que le back-end est responsable de la logique métier, du traitement des données et de la gestion des requêtes de l'utilisateur.

B. Le back-end

1. Logiciel utilisé



Spring Tool Suite 4 (STS 4) est un environnement de développement intégré (IDE) basé sur Eclipse et conçu pour faciliter le développement d'applications basées sur le framework Spring. Il propose des fonctionnalités telles que l'autocomplétion, la vérification de syntaxe, la navigation de code et la détection d'erreurs, ainsi que des outils pour la création de projets.

2. Domain ou Entity

Une entité ou (entity en Anglais) est un concept abstrait qui représente une donnée ou un ensemble de données dans un système. Elle peut contenir des attributs (ou propriétés) qui définissent les caractéristiques de la donnée, ainsi que des méthodes (ou fonctions) qui permettent de manipuler cette donnée.

Création du package : domain

```
▼ com.shape.ShapeAppSpring.domain
  > Abonnement.java
  > Entrainement.java
  > Exercice.java
  > Galerie.java
  > HttpResponse.java
  > Mensuration.java
  > Muscle.java
  > User.java
  > UserPrincipal.java
```



Prenons comme exemple la class Entrainement :

```
package com.shape.ShapeAppSpring.domain;

import java.io.Serializable;

import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.Table;

public class Entrainement implements Serializable{
    private Long entrainementId;
    private String jour;
    private String muscle;
    private String exercice;
    private Integer serie;
    private Integer repetition;
    private Integer poids;
    private Integer recuperation;
    private Integer temps;
    private Integer distance;
    private Long uid;

    // GETTER
    public Long getEntrainementId() {
        return entrainementId;
    }
    public String getJour() {
        return jour;
    }
    public String getMuscle() {
        return muscle;
    }
    public String getExercice() {
        return exercice;
    }
    public Integer getSerie() {
        return serie;
    }
    public Integer getRepetition() {
        return repetition;
    }
    public Integer getPoids() {
        return poids;
    }
    public Integer getRecuperation() {
        return recuperation;
    }
    public Integer getTemps() {
        return temps;
    }
    public Integer getDistance() {
        return distance;
    }
    public Long getUid() {
        return uid;
    }

    // SETTER
    public void setEntrainementId(Long entrainementId) {
        this.entrainementId = entrainementId;
    }
    public void setJour(String jour) {
        this.jour = jour;
    }
    public void setMuscle(String muscle) {
        this.muscle = muscle;
    }
    public void setExercice(String exercice) {
        this.exercice = exercice;
    }
    public void setSerie(Integer serie) {
        this.serie = serie;
    }
    public void setRepetition(Integer repetition) {
        this.repetition = repetition;
    }
    public void setPoids(Integer poids) {
        this.poids = poids;
    }
    public void setRecuperation(Integer recuperation) {
        this.recuperation = recuperation;
    }
    public void setTemps(Integer temps) {
        this.temps = temps;
    }
    public void setDistance(Integer distance) {
        this.distance = distance;
    }
    public void setUid(Long uid) {
        this.uid = uid;
    }

    // CONSTRUCTEUR
    public Entrainement() {
        super();
    }
    public Entrainement(Long entrainementId, String jour, String muscle,
String exercice, Integer serie, Integer repetition, Integer poids,
Integer recuperation, Integer temps, Integer distance, Long uid) {
        super();
        this.entrainementId = entrainementId;
        this.jour = jour;
        this.muscle = muscle;
        this.exercice = exercice;
        this.serie = serie;
        this.repetition = repetition;
        this.poids = poids;
        this.recuperation = recuperation;
        this.temps = temps;
        this.distance = distance;
        this.uid = uid;
    }
}
```

Public : avant la déclaration de classe signifie que la classe est accessible à partir de n'importe quelle autre classe dans le même package ou dans un autre package.

Implements : implémente l'interface "Serializable".

Serializable : L'interface "Serializable" est utilisée en Java pour permettre à un objet d'être sérialisé, c'est-à-dire converti en un flux d'octets qui peut être stocké dans un fichier ou transféré sur un réseau.

Les variables d'une classe sont utilisées pour stocker des données à l'intérieur d'une classe

L'utilisation du mot-clé "private" devant la déclaration de la variable signifie que la variable ne peut être accédée que depuis l'intérieur de la classe dans laquelle elle est déclarée. Elle n'est pas accessible à partir d'autres classes, sauf si elle est exposée via des méthodes publiques.

Long / String / Integer : sont des types de données en Java, qui sont utilisés pour stocker différents types de valeurs.

Les "getters" (ou méthodes d'accès) sont des méthodes publiques qui sont définies dans une classe pour accéder aux valeurs privées des variables de la classe.

Les "setters" (ou méthodes de mutation) sont des méthodes publiques qui sont définies dans une classe pour modifier les valeurs des variables privées de la classe.

Les constructeurs sont des méthodes spéciales qui sont définies dans une classe pour initialiser les objets créés à partir de cette classe. Les constructeurs sont appellés automatiquement lorsqu'un objet est créé à l'aide du mot-clé "new" en Java.

Le rôle principal d'un constructeur est d'initialiser les variables d'instance de la classe avec des valeurs par défaut ou avec des valeurs spécifiées par l'utilisateur au moment de la création de l'objet.

Les constructeurs peuvent prendre des paramètres ou non. Si un constructeur prend des paramètres, ces paramètres sont utilisés pour initialiser les variables d'instance de la classe.

Il est possible de définir plusieurs constructeurs dans une classe en Java, chacun avec des paramètres différents. Dans ce cas, les constructeurs sont appelés en fonction des arguments fournis lors de la création de l'objet



3. Repository

Un repository est un composant du modèle de conception "Architecture en couches" utilisé dans le développement de logiciels. Il est utilisé pour stocker et récupérer des données à partir d'une source de données, comme une base de données ou un service web.

Un repository agit comme une couche d'abstraction entre la couche de présentation (interface utilisateur) et la couche d'accès aux données (base de données, API web, etc.). Il fournit une interface simple et standardisée pour interagir avec la source de données, en cachant les détails de mise en réseau et d'accès aux données.

Les avantages de l'utilisation d'un "repository" sont :

- La séparation des responsabilités entre les différentes couches de l'application, en isolant la couche de présentation de la couche d'accès aux données.
- La réutilisabilité du code, en fournissant une interface standardisée pour interagir avec la source de données, qui peut être utilisée dans différentes parties de l'application.
- La possibilité de changer facilement la source de données, en modifiant simplement la mise en œuvre du "repository" sans affecter les autres parties de l'application.
- En résumé, un "repository" sert à abstraire la couche d'accès aux données pour simplifier la gestion des données et faciliter la maintenance de l'application.

Création du package repository :





Exemple de la classe IEntrainementRepository :

```
1 package com.shape.ShapeAppSpring.repository;
2
3 import org.springframework.data.jpa.repository.JpaRepository;
4
5 import com.shape.ShapeAppSpring.domain.Entrainement;
6
7 public interface IEntrainementRepository extends JpaRepository<Entrainement, Long>{
8
9 }
```

Public : avant la déclaration de classe signifie que la classe est accessible à partir de n'importe quelle autre classe dans le même package ou dans un autre package.

Une interface est une spécification de méthodes et de constantes qui doit être implémentée par une classe.

Le mot-clé "extends" est utilisé pour définir l'héritage entre les classes. L'héritage permet à une classe de définir une nouvelle classe en utilisant les caractéristiques d'une classe existante.

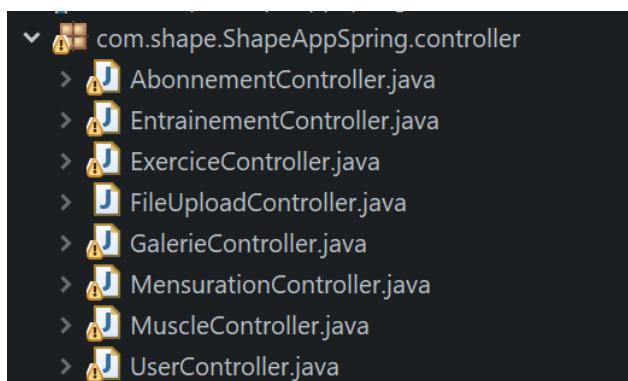
JPA est une spécification d'ORM (Object Relational Mapping) pour Java.

4. Controller

En programmation orientée objet, un package "controller" est souvent utilisé dans le cadre de l'architecture Modèle-Vue-Contrôleur (MVC) pour regrouper les classes qui gèrent les interactions entre les vues (interface utilisateur) et les modèles (données et logique de l'application).

Le package "controller" sert à encapsuler la logique de l'application et à faciliter la communication entre la vue et le modèle dans le cadre de l'architecture MVC.

Création du package controller :





Prenons comme exemple la classe EntrainementController :

```
1 package com.shape.ShapeAppSpring.controller;
2
3 import java.util.List;
4
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.http.ResponseEntity;
7 import org.springframework.validation.annotation.Validated;
8 import org.springframework.web.bind.annotation.CrossOrigin;
9 import org.springframework.web.bind.annotation.DeleteMapping;
10 import org.springframework.web.bind.annotation.GetMapping;
11 import org.springframework.web.bind.annotation.PathVariable;
12 import org.springframework.web.bind.annotation.PostMapping;
13 import org.springframework.web.bind.annotation.PutMapping;
14 import org.springframework.web.bind.annotation.RequestBody;
15 import org.springframework.web.bind.annotation.RequestMapping;
16 import org.springframework.web.bind.annotation.RestController;
17
18 import com.shape.ShapeAppSpring.dao.EntrainementDao;
19 import com.shape.ShapeAppSpring.domain.Entrainement;
20
21
22
23
24 @RestController
25 @RequestMapping
26 @CrossOrigin("*")
27 public class EntrainementController {
28
29     @Autowired
30     EntrainementDao entrainementDao;
31
32     @GetMapping("/entrainements")
33     public List<Entrainement> getAllEntrainements(@Validated @RequestBody(required = false) Entrainement entrainement) {
34         return entrainementDao.getEntrainements();
35     }
36
37
38     @PostMapping("/entrainements")
39     public Entrainement createEntrainement(@Validated @RequestBody(required = false) Entrainement entrainement) {
40         return entrainementDao.saveEntrainement(entrainement);
41     }
42
43
44     @GetMapping("/entrainements/{entrainementId}")
45     public ResponseEntity<Entrainement> findEntrainementById(@PathVariable(name = "entrainementId")Long entrainementId){
46         if (entrainementId == null) {
47             return ResponseEntity.badRequest().body("Je ne trouve pas l'entraînement avec son ID");
48         }
49
50         Entrainement entrainement = entrainementDao.getEntrainementByID(entrainementId);
51
52         if (entrainement == null) {
53             return ResponseEntity.notFound().build();
54         }
55
56         return ResponseEntity.ok().body(entrainement);
57     }
58
59
60     @PutMapping("/entrainements/{entrainementId}")
61     public ResponseEntity<Entrainement> updateEntrainement (@Validated @PathVariable(name = "entrainementId")Long entrainementId,
62     @RequestBody(required = false) Entrainement entrainement) {
63         if (entrainement == null) {
64             return ResponseEntity.notFound().build();
65         }
66         entrainement.setEntrainementId(entrainementId);
67         entrainementDao.updateEntrainement(entrainement);
68         return ResponseEntity.ok().body(entrainement);
69     }
70
71     @DeleteMapping("/entrainements/{entrainementId}")
72     public ResponseEntity<Entrainement> deleteEntrainement (@Validated @PathVariable(name = "entrainementId")Long entrainementId) {
73
74         Entrainement entrainement = entrainementDao.getEntrainementByID(entrainementId);
75
76         if (entrainement == null) {
77             return ResponseEntity.notFound().build();
78         }
79         entrainementDao.deleteEntrainement(entrainement);
80         return ResponseEntity.ok().body(entrainement);
81
82     }
83
84 }
```



L'annotation `@RestController` est une combinaison des annotations `@Controller` et `@ResponseBody`. Elle indique que les méthodes de la classe sont des points de terminaison REST et que leurs résultats seront directement envoyés en tant que corps de réponse HTTP, sans vue intermédiaire. Elle est utilisée pour marquer une classe comme étant un contrôleur REST, qui définit des points de terminaison d'API REST pour une application et retourne des réponses HTTP directement, sans vue intermédiaire.

L'annotation `@RequestMapping` est utilisée pour associer une méthode de contrôleur à un point de terminaison d'API REST, en spécifiant l'URL de la requête, le type de requête HTTP et les paramètres de requête optionnels.

L'annotation `@Autowired` est utilisée pour injecter automatiquement les dépendances dans une classe.

L'annotation `@GetMapping` est utilisée pour mapper une méthode à une requête HTTP GET.

L'annotation `@PostMapping` est utilisée pour mapper une méthode à une requête HTTP POST.

L'annotation `@DeleteMapping` est utilisée pour mapper une méthode à une requête HTTP DELETE.

L'annotation `@RequestBody` est utilisée pour lier le corps d'une requête HTTP à un objet Java dans une méthode de contrôleur de Spring.

5. Dao

DAO signifie "Data Access Object" en anglais, qui peut être traduit par "Objet d'accès aux données". En programmation orientée objet, une DAO est une classe qui fournit une interface entre l'application et la source de données, telle qu'une base de données, un fichier ou un service web.

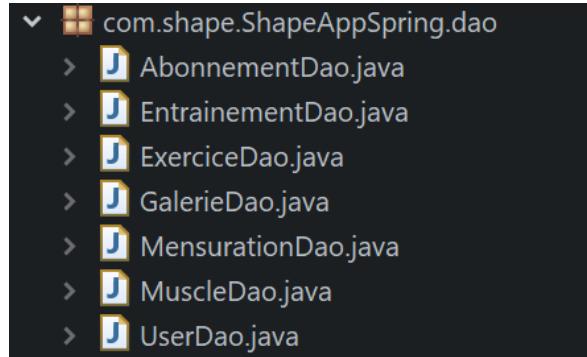
Le rôle principal de la DAO est de fournir un moyen d'accéder aux données de manière structurée et indépendante de la source de données sous-jacente. Elle encapsule la logique d'accès aux données en fournissant des méthodes d'accès pour effectuer des opérations de CRUD (Create, Read, Update, Delete) sur les données.

En utilisant une DAO, on peut séparer la logique de l'application de la logique d'accès aux données, ce qui permet de rendre l'application plus modulaire et plus facile à maintenir. Cela facilite également le remplacement de la source de données sous-jacente sans avoir à modifier l'application.



En résumé, la DAO est utilisée pour fournir une interface structurée et indépendante de la source de données pour accéder aux données et effectuer des opérations de CRUD sur celles-ci. Cela permet de séparer la logique de l'application de la logique d'accès aux données et facilite la maintenance et la modularité de l'application.

Création du package Dao :



Prenons par exemple la classe EntrainementDao :

```
1 package com.shape.ShapeAppSpring.dao;
2
3 import java.util.List;
4
5
6 @Service
7 public class EntrainementDao {
8
9     @Autowired
10    IEntrainementRepository entrainementRepository;
11
12
13
14    // Liste des Entraînements
15    public List<Entrainement> getEntrainements() {
16        return entrainementRepository.findAll();
17    }
18
19
20    // Save un Entrainement
21    public Entrainement saveEntrainement(Entrainement entrainement) {
22        return entrainementRepository.save(entrainement);
23    }
24
25
26    // get un Entrainement by ID
27    public Entrainement getEntrainementByID(Long entrainementId) {
28        return entrainementRepository.findById(entrainementId).get();
29    }
30
31
32    // Delete un Entrainement
33
34    public void deleteEntrainement(Entrainement entrainement) {
35        entrainementRepository.delete(entrainement);
36    }
37
38    // Update un Entrainement
39
40    public Entrainement updateEntrainement(Entrainement entrainement) {
41        return entrainementRepository.save(entrainement);
42    }
43
44
45}
```



C. Le Front-end

1. Logiciels et framework



Visual Studio Code est un éditeur de code extensible développé par Microsoft pour Windows, Linux et macOS.



Angular est un framework pour clients, open source, basé sur TypeScript.

2. Qu'est-ce que Angular et à quoi sert-il ?

Angular est une plateforme de développement, construite sur TypeScript.

Développé par Google, Angular est un Framework open source écrit en JavaScript qui permet la création d'applications Web et plus particulièrement de ce qu'on appelle des « Single Page Applications » : des applications web accessibles via une page web unique qui permet de fluidifier l'expérience utilisateur et d'éviter les chargements de pages à chaque nouvelle action.

En tant que plateforme, Angular comprend :

- Un cadre basé sur des composants pour la création d'applications web évolutives.
- Une collection de bibliothèques bien intégrées qui couvrent une grande variété de fonctionnalités, notamment le routage, la gestion des formulaires, la communication client-serveur, etc.
- Une suite d'outils de développement pour aider à développer, construire, tester et mettre à jour son code.

3. Crédit du projet Angular Shape

Création du dossier Shape dans un premier temps, puis dans un second temps, lancer le CMD dans ce même dossier et taper « `ng new shapeApp` » puis activer le routing avec Angular.



```
C:\INTERNE\IDK\shape> ng new shapeApp  
? Would you like to add Angular routing? (y/N) y|
```

Choisir le format pour le stylesheet (nous avons choisi CSS)

```
C:\INTERNE\IDK\shape> ng new shapeApp  
? Would you like to add Angular routing? Yes  
? Which stylesheet format would you like to use? (Use arrow keys)  
> CSS  
SCSS [ https://sass-lang.com/documentation/syntax#scss ]  
Sass [ https://sass-lang.com/documentation/syntax#the-indented-syntax ]  
Less [ http://lesscss.org ]
```

L'installation des packages s'effectue :

```
C:\INTERNE\IDK\shape> ng new shapeApp  
? Would you like to add Angular routing? Yes  
? Which stylesheet format would you like to use? CSS  
CREATE shapeApp/angular.json (2711 bytes)  
CREATE shapeApp/package.json (1040 bytes)  
CREATE shapeApp/README.md (1062 bytes)  
CREATE shapeApp/tsconfig.json (901 bytes)  
CREATE shapeApp/.editorconfig (274 bytes)  
CREATE shapeApp/.gitignore (548 bytes)  
CREATE shapeApp/tsconfig.app.json (263 bytes)  
CREATE shapeApp/tsconfig.spec.json (273 bytes)  
CREATE shapeApp/.vscode/extensions.json (130 bytes)  
CREATE shapeApp/.vscode/launch.json (474 bytes)  
CREATE shapeApp/.vscode/tasks.json (938 bytes)  
CREATE shapeApp/src/favicon.ico (948 bytes)  
CREATE shapeApp/src/index.html (294 bytes)  
CREATE shapeApp/src/main.ts (214 bytes)  
CREATE shapeApp/src/styles.css (80 bytes)  
CREATE shapeApp/src/assets/.gitkeep (0 bytes)  
CREATE shapeApp/src/app/app-routing.module.ts (245 bytes)  
CREATE shapeApp/src/app/app.module.ts (393 bytes)  
CREATE shapeApp/src/app/app.component.html (23115 bytes)  
CREATE shapeApp/src/app/app.component.spec.ts (1079 bytes)  
CREATE shapeApp/src/app/app.component.ts (212 bytes)  
CREATE shapeApp/src/app/app.component.css (0 bytes)  
✓ Packages installed successfully.
```

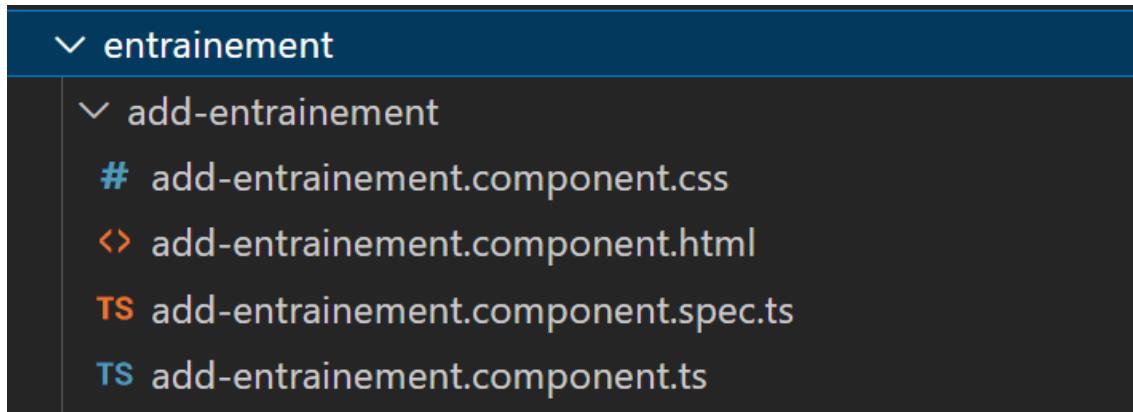


4. Création de component

Il faut commencer par créer un dossier components dans src. Pour chaque création d'un nouveau component, il faut saisir la commande « `ng g c xxxx` ». Par exemple, pour la création du component Entrainement, il faut saisir la commande « `ng g c Entrainement` ».

```
C:\INTERNE\IDK\shape\shapeApp\src\app\components>ng g c entraînement
CREATE src/app/components/entraînement/entraînement.component.html (27 bytes)
CREATE src/app/components/entraînement/entraînement.component.spec.ts (641 bytes)
CREATE src/app/components/entraînement/entraînement.component.ts (226 bytes)
CREATE src/app/components/entraînement/entraînement.component.css (0 bytes)
UPDATE src/app/app.module.ts (510 bytes)
```

Un composant est un élément fondamental d'une application Angular. Un composant est un module qui contient du code HTML, CSS et JavaScript pour afficher une partie de l'interface utilisateur d'une application.



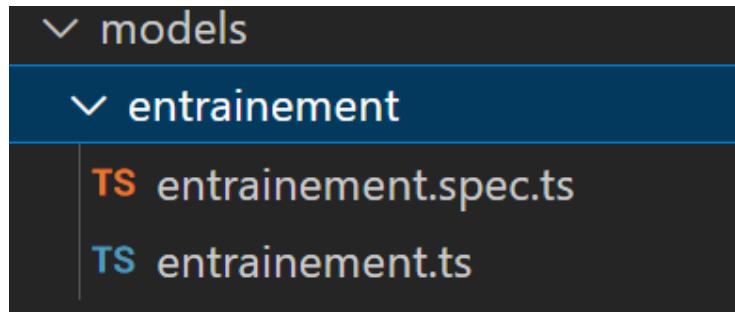
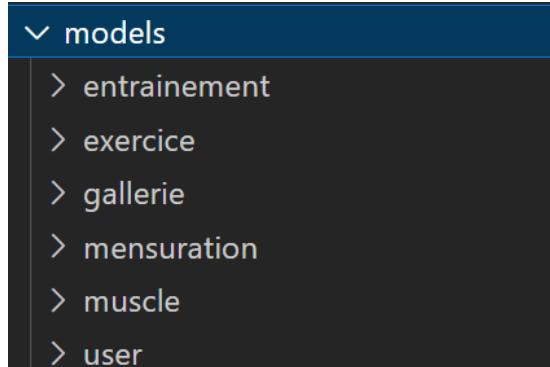
La création de ce nouveau component met à jour directement app.module.ts, EntrainementComponent est ajouté automatiquement.



```
1 import { NgModule } from '@angular/core';
2 import { BrowserModule } from '@angular/platform-browser';
3
4 import { AppRoutingModule } from './app-routing.module';
5 import { AppComponent } from './app.component';
6 import { EntrainementComponent } from './components/entrainement/entrainement.component';
7
8 @NgModule({
9   declarations: [
10     AppComponent,
11     EntrainementComponent
12   ],
13   imports: [
14     BrowserModule,
15     AppRoutingModule
16   ],
17   providers: [],
18   bootstrap: [AppComponent]
19 })
20 export class AppModule { }
```

5. Les models

Je crée un dossier « models » qui contiendra les différentes classes :



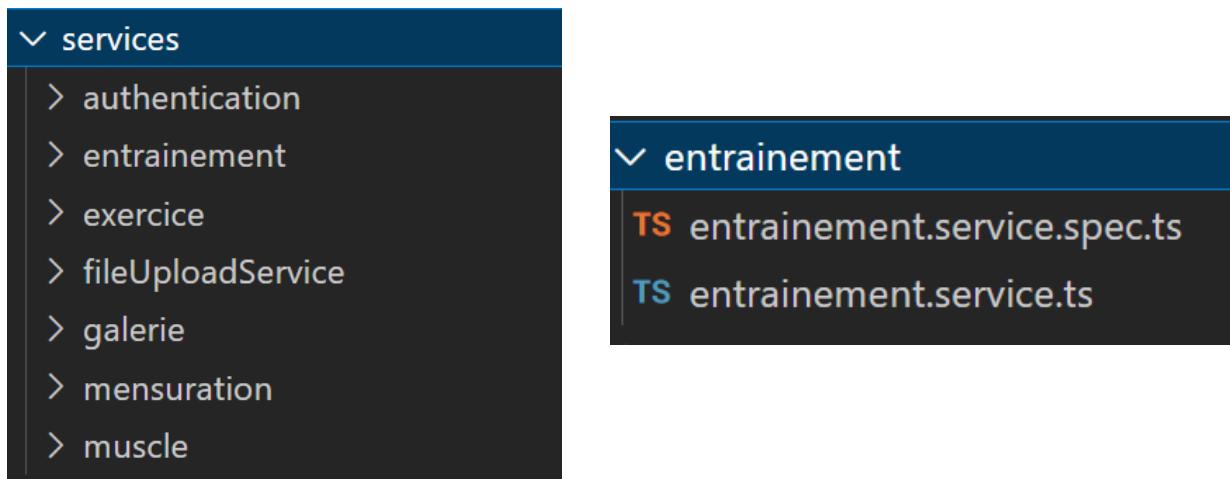


Par exemple pour le fichier entrainement.ts, nous avons :

```
1  export class Entrainement {  
2  
3      public entrainementId : number;  
4      public jour: string;  
5      public muscle: string;  
6      public exercice: string;  
7      public serie: number;  
8      public repetition: number;  
9      public poids: number;  
10     public recuperation: number;  
11     public temps: number;  
12     public distance: number;  
13     public uid: number;  
14     constructor(){  
15         this.entrainementId= 0;  
16         this.muscle= "";  
17         this.exercice= "";  
18         this.jour= "";  
19         this.serie= 0;  
20         this.repetition= 0;  
21         this.poids= 0;  
22         this.recuperation= 0;  
23         this.temps= 0;  
24         this.distance= 0;  
25         this.uid= 0;  
26     }  
27 }  
28 }
```

6. Les services

Je crée un dossier « services » qui contiendra les différents services :





Par exemple pour le fichier entrainement.service.ts, nous avons :

```
1 import { AppSettings } from '../../../../../settings/app.setting';
2 import { Observable } from 'rxjs';
3 import { Entrainement } from '../../../../../models/entrainement/entrainement';
4 import { HttpClient, HttpHeaders } from '@angular/common/http';
5 import { Injectable } from '@angular/core';
6 import { environment } from 'src/environments/environment';
7
8 @Injectable({
9   providedIn: 'root'
10 })
11 export class EntrainementService {
12
13   httpOptions = {
14     headers : new HttpHeaders ({'Content-Type' : 'application/json'})
15   }
16
17   private host = environment.apiUrl;
18   constructor(private http: HttpClient) {
19   }
20
21   /*On récupère les entraînements du backend via l'URL /entrainements */
22   public getAllEntrainements(): Observable<Entrainement[]>{
23     return this.http.get<Entrainement[]>(`${this.host}/entrainements`)
24   }
25
26   /* FormData injecte les données via un form Data voir dans postman (formulaire); il faut une key et une value */
27   public addEntrainement(formData: FormData): Observable<Entrainement>{
28     return this.http.post<Entrainement>(` ${this.host}/entrainements`, formData)
29   }
30
31   /* FormData injecte les données à modifier via un formulaire (voir dans PostMan section FormData pour les tests)
32   | | Attention il est possible de rencontrer une erreur plusieurs champs sont required pour le update() */
33   public updateEntrainement(formData: FormData): Observable<Entrainement>{
34     return this.http.post<Entrainement>(` ${this.host}/entrainements`, formData)
35   }
36
37   /* I delete un entraînement */
38   public deleteEntrainement(entrainementId: number): Observable< Entrainement >{
39     return this.http.delete<Entrainement>(` ${this.host}/entrainements/${entrainementId}`);
40   }
41
42   public editEntrainement(id:number) {
43     return this.http.get(AppSettings.APP_URL+'/entrainements/'+id)
44   }
45
46   public updateEntrainement2(entrainement:Entrainement){
47     return this.http.put(AppSettings.APP_URL+'/entrainements/'+ entrainement.entrainementId, JSON.stringify(entrainement),this.httpOptions);
48   }
49
50 }
```

7. Le HTML



Le HTML ou HyperText Markup Language (HTML) est le code utilisé pour structurer une page web et son contenu.



Voici le HTML de Entrainement :

```
1 <br>
2 <br>
3 <h2 class="text-center">Ajouter un entrainement</h2>
4 <br>
5 <div class="container">
6   <form [formGroup]="form" (ngSubmit)="create()">
7     <div class="form-group">
8
9       <!-- Radio choix jour de la semaine -->
10      <div class="radioJour">
11        <div class="container radioJour">
12          <input type="radio" value="Lundi" formControlName="jour" id="lundi">
13          <label for="lundi">Lundi</label>
14          <input type="radio" value="Mardi" formControlName="jour" id="mardi">
15          <label for="mardi">Mardi</label>
16          <input type="radio" value="Mercredi" formControlName="jour" id="mercredi">
17          <label for="mercredi">Mercredi</label>
18          <input type="radio" value="Jeudi" formControlName="jour" id="jeudi">
19          <label for="jeudi">Jeudi</label>
20          <input type="radio" value="Vendredi" formControlName="jour" id="vendredi">
21          <label for="vendredi">Vendredi</label>
22          <input type="radio" value="Samedi" formControlName="jour" id="samedi">
23          <label for="samedi">Samedi</label>
24          <input type="radio" value="Dimanche" formControlName="jour" id="dimanche">
25          <label for="dimanche">Dimanche</label>
26        </div>
27      </div>
28    <br>
29
30    <div class="container choixExos">
31      <!-- Choix radio muscle -->
32      <div class="radio" >
33        <div class="container radioMuscle" *ngFor="let m of muscles">
34          <input type="radio" id="muscle_{{m.muscleId}}" value="{{m.nom}}" formControlName="muscle" (change)="afficherExo(m.muscleId)">
35          <label for="muscle_{{m.muscleId}}">{{m.nom}}</label>
36        </div>
37      </div>
38
39      <!-- Choix radio exercice par muscle affichage ou non -->
40
41      <div>
42        <div [class]="'radio' + (typeMuscle != item.muscleId? ' hide':'') " *ngFor="let item of exercices" >
43          <div class="container radioExercice" *ngIf="item.muscleId==1" >
44            <input type="radio" id="exercicePapule_{{item.exerciceId}}" value="{{item.nom}}" formControlName="exercice">
45            <label for="exercicePapule_{{item.exerciceId}}">{{item.nom}}</label>
46          </div>
47        </div>
48      </div>
49
50
51      ...
52
53      <div>
54        <div [class]="'radio' + (typeMuscle != item.muscleId? ' hide':'') " *ngFor="let item of exercices" >
55          <div class="container radioExercice" *ngIf="item.muscleId==10" >
56            <input type="radio" id="exerciceMollets_{{item.exerciceId}}" value="{{item.nom}}" formControlName="exercice">
57            <label for="exerciceMollets_{{item.exerciceId}}">{{item.nom}}</label>
58          </div>
59        </div>
60      </div>
61    </div>
62
63    <div class="input-field">
64      <label for="serie">Série</label>
65      <input id="champs" type="text" class="form-control" formControlName="serie" data-length="10">
66    </div>
67    <div class="input-field">
68      <label for="repetition">Répétition</label>
69      <input id="champs" type="text" class="form-control" formControlName="repetition" data-length="10">
70    </div>
71    <div class="input-field">
72      <label for="poids">Poids</label>
73      <input id="champs" type="text" class="form-control" formControlName="poids" data-length="10">
74    </div>
75    <div class="input-field">
76      <label for="temps">Temps</label>
77      <input id="champs" type="text" class="form-control" formControlName="temps" data-length="10">
78    </div>
79    <div class="input-field">
80      <label for="distance">Distance</label>
81      <input id="champs" type="text" class="form-control" formControlName="distance" data-length="10">
82    </div>
83
84    <br>
85    <br>
86    <div class="container buttonSubmit">
87      <button class="btn btn-success" type="submit" style="align-items: center;">Soumettre</button>
88    </div>
89
90    <br>
91    <br>
92  </div>
93 </div>
94 </div>
95 {{form.value | json}}
```



8. Le CSS



Les feuilles de style en cascade, généralement appelées CSS de l'anglais Cascading Style Sheets, forment un langage informatique qui décrit la présentation des documents HTML et XML.

Voici le CSS d'Entrainement :

```
1 /* CSS Jours */
2 .container .radioJour{
3   display: flex;
4   flex-wrap: wrap;
5   justify-content: space-between;
6
7   color: #fff;
8   font-size: 2.2rem;
9 }
10 .radioJour input:checked + label{
11   background-image: linear-gradient(180deg, #03c7f4, #021e38);
12   padding: 0.5rem 1rem;
13   border-radius: 10px;
14   box-shadow: 6px 6px 5px #058c9131;
15 }
16
17 /* CSS Muscle */
18 .container .radioMuscle{
19   color: #fff;
20   font-size: 1.2rem;
21 }
22 .radioMuscle input:checked + label{
23   background-image: linear-gradient(180deg, #08dee6, #012d0d);
24   padding: 0.5rem 5rem;
25   border-radius: 10px;
26   box-shadow: 6px 6px 5px #058c9131;
27 }
28
29 /* CSS Exercice */
30 .container .radioExercice{
31   color: #fff;
32   font-size: 1.2rem;
33   display: flex;
34   flex-direction: column;
35 }
36 .radioExercice input:checked + label{
37   background-image: linear-gradient(180deg, #f49e0a, #282303);
38   padding: 0.5rem 5rem;
39   border-radius: 10px;
40   box-shadow: 6px 6px 5px #f49e0a43;
41 }
42
43
44 /* CSS Radio */
45 .radio input{
46   display: none;
47 }
48
49 .radio.hide{
50   display: none;
51 }
52
53 /* Titre de la section */
54 h2 {
55   color: white;
56 }
57
58
59 /* CSS Inputs Autres champs */
60 .input-field{
61   margin: auto;
62   color: #00c6f8bd;
63   font-size: 1rem;
64   width: 12rem;
65   display: flex;
66   flex-wrap: wrap;
67   justify-content: center;
68 }
69
70 #champs{
71   background: #222429;
72   color: #00c6f8bd;
73 }
74
75 /* CSS les deux */
76 .choixExos{
77   display: flex;
78   align-items: center;
79 }
80
81
82 /* Button submit */
83 .buttonSubmit{
84   display: flex;
85   justify-content: center;
86 }
87
88
89 
```

9. Tableau des routes

Ensuite, j'ai dû mettre à jour le fichier app-routing.module.ts pour pourvoir accéder au component créé. Le fichier app-routing.module.ts est une classe TypeScript exportée. Elle contient un tableau de routes nommé routes.



```
1 import { RegisterComponent } from './components/login-register/register/register.component';
2 import { ContactComponent } from './components/annexe/contact/contact.component';
3 import { LoginComponent } from './components/login-register/login/login.component';
4 import { AccueilComponent } from './components/annexe/accueil/accueil.component';
5 import { ListDataComponent } from './components/user/mensuration/data/list-data/list-data.component';
6 import { EditEntrainementComponent } from './components/user/entrainement/edit-entrainement.component';
7 import { UploadfileComponent } from './components/user/gallerie/upload-file/upload-file.component';
8 import { Routes, RouterModule } from '@angular/router';
9 import { AddEntrainementComponent } from './components/user/entrainement/add-entrainement/add-entrainement.component';
10 import { ListEntrainementComponent } from './components/user/entrainement/list-entrainement/list-entrainement.component';
11 import { ExerciceComponent } from './components/user/exercice/exercice.component';
12 import { GraphiqueComponent } from './components/user/graphique/graphique.component';
13 import { CalculBmiComponent } from './components/user/calcul-bmi/calcul-bmi.component';
14 import { NgModule } from '@angular/core';

15
16
17 const routes: Routes = [
18   {path: 'accueil', component: AccueilComponent}, // Accueil
19   {path: 'login', component: LoginComponent}, // Login
20   {path: 'register', component: RegisterComponent}, // Register
21   {path: 'contact', component: ContactComponent}, // Cobntact
22   {path: 'bmi', component: CalculBmiComponent}, // IMC
23   {path: 'graphique', component: GraphiqueComponent}, // Graphique
24   {path: 'exercice', component: ExerciceComponent}, // Exercice
25   {path: 'entrainement', component: ListEntrainementComponent}, // Entrainement
26   {path: 'entrainements/:id', component: ListEntrainementComponent},
27   {path: 'addEntrainement', component: AddEntrainementComponent}, // Ajouter un entrainement
28   {path: 'editEntrainement/:id', component: EditEntrainementComponent}, // Edit un entrainement
29   {path: 'galerie', component: UploadfileComponent}, // Galerie
30   {path: 'data', component: ListDataComponent}, // Mensuration
31   {path: 'data/:id', component: ListDataComponent},
32 ];
33
34 @NgModule({
35   imports: [RouterModule.forRoot(routes)],
36   exports: [RouterModule]
37 })
38 export class AppRoutingModule { }
```

10. Démarrage du serveur

Pour démarrer le serveur, il faut taper dans la console « ng serve -o », le « -o » ce qui permet d'ouvrir automatique la page web.

```
C:\INTERNE\Github\JAVA_PROJET_2023\ApplicationShapeFinaleWithSecu\ShapeAppAngularWithSecu\src>ng serve -o
```

Une fois la commande entrée, le serveur démarre.

```
Initial Chunk Files | Names           | Raw Size
vendor.js          | vendor          | 3.26 MB |
main.js            | main            | 527.74 kB |
styles.css, styles.js | styles          | 393.67 kB |
polyfills.js       | polyfills       | 314.28 kB |
runtime.js         | runtime         | 6.53 kB |

| Initial Total | 4.48 MB

Build at: 2023-04-04T16:02:33.906Z - Hash: 297448728dad15ab - Time: 6910ms

** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **

✓ Compiled successfully.
```



La page internet s'ouvre avec le site internet :

The screenshot shows the homepage of the Shape website. At the top, there is a navigation bar with links for ACCUEIL, PLANNING, GALERIE, MÉDURATION+, EXERCICE, TWC, ADMIN, and CONTACT. There are also buttons for S'INSCRIRE and SE CONNECTER. The main header features the text "WELCOME" and "SHAPE YOUR BODY". Below the header, there is a large image of three people in a gym setting. A call-to-action button labeled "ENVOI" is visible. The middle section has a dark background with icons for "POURQUOI UTILISER SHAPE ?" and "POUSSE TES LIMITES PLUS LOIN". It includes four cards: "Suis ton évolution", "Calcule ton BMI", "Accès à ta galerie", and "Trouve des exercices adaptés". The bottom section is titled "OUR PLAN" and "CHOISIS TON ABONNEMENT ADAPTÉ", showing three plan options: "Gratuit" (0€), "6 mois" (20€), and "12 Mois unlimited" (35€). Each plan card lists features like "Accès aux exercices", "Prépare ton planning", etc., and a "Enroll now" button. Below this, there are statistics: "EXPERIENCE 12345", "OUR TRAINERS 12345", "COMPLETE PROJECT 12345", and "HAPPY CLIENTS 12345". The "THE TEAM" section features three trainers: JOHN DOD, JAMES TAYLOR, and ADAM PHILLIPS. At the bottom, there is a "CONTACT" section with address and phone number, and a "NEWSLETTER" sign-up form.



Par exemple pour le component Entrainement, nous avons la page suivante :

The screenshot shows a dark-themed web application for managing workouts. At the top, there's a header with the logo (a blue octopus), contact information (shape@gmail.com, +33 (6) 29 39 05 26), social media links (Facebook, Twitter, LinkedIn, Instagram, YouTube), and navigation links (ACCUEIL, PLANNING, GALERIE, MENSURATION, EXERCICE, IMC, ADMIN, CONTACT, S'ENREGISTRER, SE CONNECTER).

The main content area is titled "Ajouter un entraînement" (Add a training). It features a weekly calendar grid:

Lundi	Mardi	Mercredi	Jeudi	Vendredi	Samedi	Dimanche
Épaules						
Biceps	Développé militaire					
Triceps	Développé Arnold					
Pectoraux	Face pull					
Dos	Élévations latérales à la poulie					
Abdominaux	Élévations latérales à la machine					
Fessiers	Développé épaules avec haltères					
Quadriceps	Développé épaules debout à la landmine					
Ischio-jambiers	Rotation externe de l'épaule à la poulie					
Mollets	Pec deck inversé					
	Élévations frontales					
	Élévations latérales					
	Presse à épaules inclinée					

Below the calendar, there are input fields for workout parameters:

Série	12
Répétition	12
Poids	35
Temps	
Distance	

A green "SOUMETTRE" (Submit) button is located at the bottom of the form.

At the bottom left, there's a "CONTACT" section with address, email, and phone number information. At the bottom right, there's a "NEWSLETTER" sign-up form with a "Your Email" input field and a "S'INSCRIRE" (Subscribe) button.

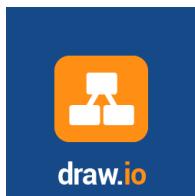
© Shape by Adrien Houen. All Rights Reserved.



IV. Concevoir et développer la persistance des données

A. Concevoir une base de données

1. Logiciel utilisé



Draw.io est un outil en ligne de création de diagrammes et de schémas, gratuit et open-source. Il permet de concevoir des organigrammes, des cartes mentales, des diagrammes de flux, des diagrammes de réseau, des diagrammes de Gantt, des plans de site web et bien plus encore. L'outil propose une large gamme de formes, de modèles et de fonctionnalités de personnalisation pour aider les utilisateurs à créer des diagrammes précis et professionnels. Draw.io peut être intégré à diverses applications telles que Google Drive, OneDrive, Dropbox, GitHub, etc., ce qui facilite le partage et la collaboration en temps réel.



MySQL Workbench est un outil de modélisation et d'administration de base de données open-source développé par Oracle. Il permet de concevoir, de modéliser et de visualiser des bases de données MySQL à l'aide d'une interface graphique intuitive. L'outil offre également des fonctionnalités d'administration, telles que la gestion des utilisateurs et des priviléges, la configuration du serveur MySQL, la sauvegarde et la restauration de données. MySQL Workbench peut être utilisé sur plusieurs plateformes, notamment Windows, Linux et macOS, et est compatible avec les dernières versions de MySQL.



Spring Tool Suite 4 (STS 4) est un environnement de développement intégré (IDE) basé sur Eclipse et conçu pour faciliter le développement d'applications basées sur le framework Spring. Il propose des fonctionnalités telles que l'autocomplétion, la vérification de syntaxe, la navigation de code et la détection d'erreurs, ainsi que des outils pour la création de projets, la gestion de dépendances et la mise en place de tests. STS 4 est compatible avec une large gamme de technologies liées à Spring, telles que Spring Boot, Spring Cloud, Spring Batch et Spring Integration, et peut être utilisé sur différentes plates-formes, notamment Windows, Linux et macOS.



2. Les étapes d'une base de données

Voici les étapes de base pour créer une base de données :

- Analyser les besoins : la première étape est de comprendre les besoins en matière de stockage des données de l'application ou du système qui utilisera la base de données. Il est important de déterminer les entités (tables) nécessaires, les relations entre elles, les types de données à stocker, etc.
- Concevoir le schéma : à partir de l'analyse des besoins, il est nécessaire de concevoir un schéma de base de données qui définit les tables, les colonnes, les clés primaires et étrangères, les contraintes, etc. Cette étape est importante car elle définit la structure de la base de données.
- Créer la base de données : une fois que le schéma est défini, la base de données peut être créée. Créer les tables, les colonnes et les contraintes.
- Ajouter des données : la base de données est désormais prête à recevoir des données. Les données peuvent être ajoutées manuellement à l'aide d'outils de gestion de base de données ou programmées via une application ou un script.

Tester et maintenir : enfin, il est important de tester la base de données pour s'assurer qu'elle fonctionne correctement et qu'elle répond aux besoins de l'application ou du système. Il est également important de surveiller et de maintenir la base de données pour éviter les erreurs et les pannes.

3. Diagramme de base de données

Un diagramme de base de données est une représentation visuelle des relations entre les entités dans une base de données.

- Le diagramme est une représentation graphique des tables et de leurs relations dans une base de données.
- Chaque table est composée d'attributs (colonnes) et a une clé primaire unique.
- Les tables peuvent être reliées entre elles par des clés étrangères pour former des relations.
- Des contraintes peuvent être ajoutées pour garantir la cohérence des données entre les tables.
- Le diagramme peut inclure des notes et des commentaires pour fournir des informations supplémentaires sur la structure de la base de données.



- Les clés étrangères sont utilisées pour lier les tables entre elles, en faisant référence à la clé primaire d'une autre table.
- Les tables peuvent également avoir des contraintes, telles que des contraintes d'intégrité référentielle pour garantir que les données sont cohérentes entre les tables.

Explanation :

L'annotation `@OneToOne` définit une relation 1:1 entre deux entités. Si cette relation n'est pas forcément très courante dans un modèle relationnel de base de données, elle se rencontre très souvent dans une modèle objet.



L'annotation `@OneToMany` définit une relation 1:n entre deux entités. Cette annotation ne peut être utilisée qu'avec une collection d'éléments puisqu'elle implique qu'il peut y avoir plusieurs entités associées.



L'annotation `@ManyToMany` définit une relation n:n entre deux entités. Cette annotation ne peut être utilisée qu'avec une collection d'éléments puisqu'elle implique qu'il peut y avoir plusieurs entités associées.

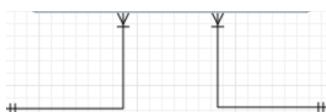
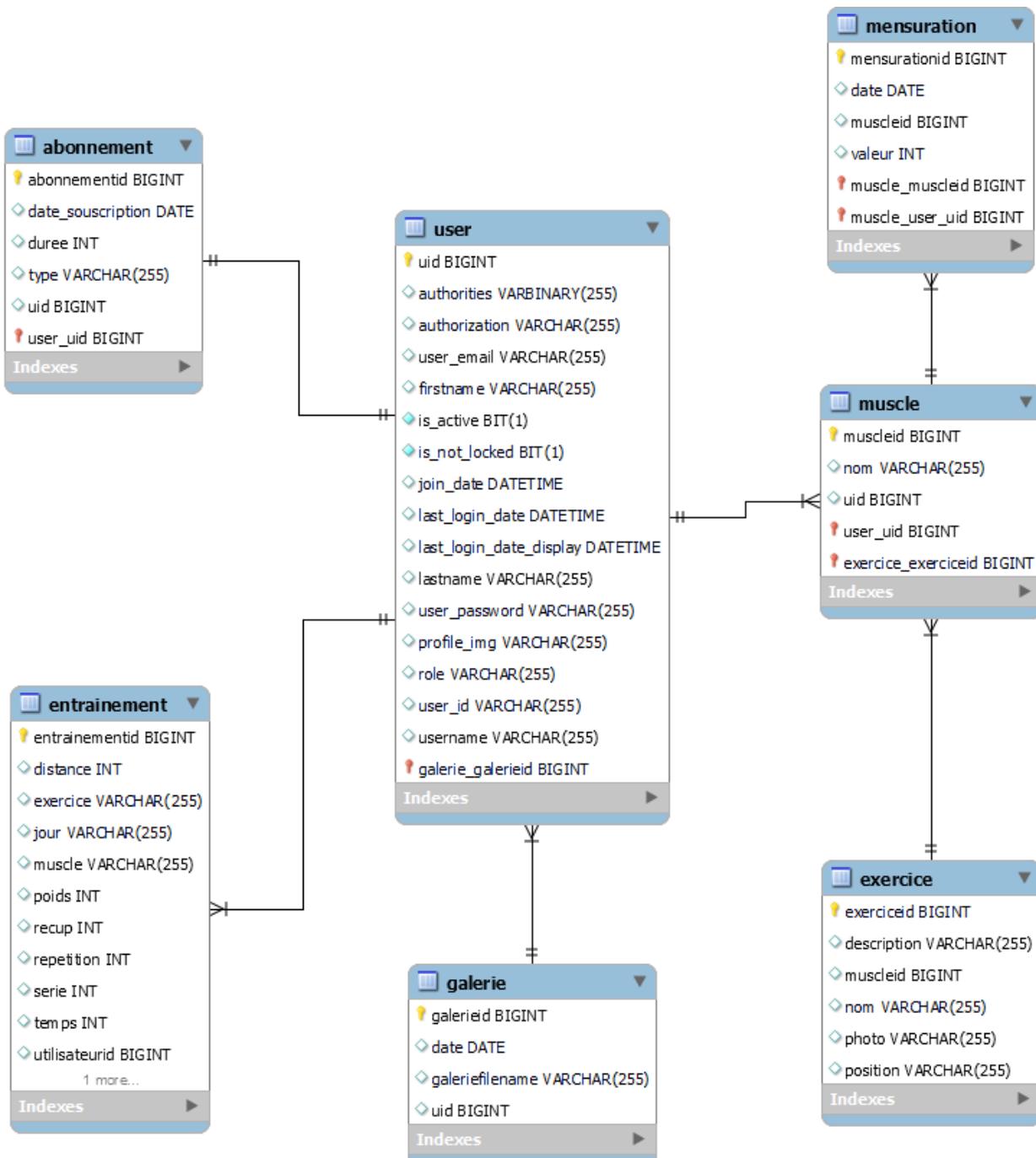




Diagramme de base données pour l'application Shape :





B. Mise en place d'une base de données

1. Le fichier application.properties

Le fichier application.properties est un fichier de configuration utilisé dans les applications Spring pour spécifier différents paramètres tels que la configuration de la base de données, les propriétés Hibernate, les ports de serveur, les chemins de fichiers, etc.

Fichier application.properties pour l'application SHAPE :

```
# Spécifie le nom de l'en-tête de la demande HTTP utilisé pour envoyer le jeton d'authentification JWT.
jwt.header=Authorization
# Spécifie la clé secrète utilisée pour signer et vérifier les jetons JWT.
jwt.secret='[a-zA-Z0-9._]^+$Guidelines89797987forAlphabeticalArraNumeralsandOtherSymbol$'
# Spécifie l'URL de la base de données MySQL utilisée par l'application.
spring.datasource.url=jdbc:mysql://localhost:3306/shapeappspring01?createDatabaseIfNotExist=true&serverTimezone=UTC
# Spécifie le nom d'utilisateur pour se connecter à la base de données.
spring.datasource.username=root
# Spécifie le mot de passe pour se connecter à la base de données.
spring.datasource.password=admin
# Spécifie le nom de la classe de pilote JDBC utilisée pour se connecter à la base de données MySQL.
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
# Indique comment Hibernate doit mettre à jour la base de données lors du démarrage de l'application.
spring.jpa.hibernate.ddl-auto=update
# Indique si Hibernate doit afficher les requêtes SQL générées.
spring.jpa.show-sql=true
# Spécifie le dialecte Hibernate à utiliser pour interagir avec la base de données MySQL.
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect
# Active le chargement paresseux pour les relations de l'entité.
spring.jpa.properties.hibernate.enable_lazy_load_no_trans=true
# Indique si Hibernate doit formater les requêtes SQL générées.
spring.jpa.properties.hibernate.format_sql=true
# Spécifie le port sur lequel l'application Spring sera exécutée.
server.port=8090
# Indique si Hibernate doit générer les scripts de la base de données au démarrage de l'application.
spring.jpa.generate-ddl=true
# Indique que la base de données utilisée est MySQL.
spring.jpa.database=mysql
```

2. ORM

Les annotations sont des instructions fournies dans le code source d'une application pour aider un framework ou un ORM (Object-Relational Mapping) à comprendre la structure et les relations entre les objets de l'application et les tables de la base de données. Les annotations sont donc essentielles pour la mise en place d'une base de données car elles permettent de mapper les objets de l'application aux tables de la base de données et d'établir les relations entre eux.

Les annotations facilitent également la création des requêtes SQL nécessaires à la création et à la manipulation de la base de données. Elles permettent de générer



automatiquement le code SQL pour créer les tables et les relations entre elles, ainsi que pour insérer, mettre à jour et supprimer des données.

Les annotations sont essentielles pour la mise en place d'une base de données car elles permettent de mapper les objets de l'application aux tables de la base de données, d'établir les relations entre eux, et de définir des contraintes et des attributs pour faciliter la gestion de la base de données.

Par exemple, prenons la class Muscle.java :

```
1 package com.shape.ShapeAppSpring.domain;
2
3 import java.io.Serializable;
4
5
6 @Entity
7 @Table(name="MUSCLE")
8 public class Muscle implements Serializable{
9
10     @Id
11     @GeneratedValue(strategy = GenerationType.IDENTITY)
12     @Column(name = "MUSCLEID")
13     private Long muscleId;
14
15     @Column(name = "NOM")
16     private String nom;
17     @Column(name = "UID")
18     private Long uid;
19
20     // ASSOCIATION
21     //Avec Mensuration
22     @OneToMany(fetch = FetchType.LAZY, mappedBy = "muscleId")
23     private List<Mensuration> listMensuration= new ArrayList<>();
24
25     //Avec Exercice
26     @OneToMany(fetch = FetchType.LAZY, mappedBy = "muscleId")
27     private List<Exercice> listExercice= new ArrayList<>();
28
29
30
31
32
33
34
35
36
37
38
39
40
```

`@Entity` est utilisée en Java pour indiquer qu'une classe représente une entité dans une base de données relationnelle. La classe doit avoir une clé primaire qui identifie les enregistrements dans la table correspondante. Les attributs de la classe sont mappés aux colonnes de la table, et les relations entre les entités sont définies à l'aide d'autres annotations.

`@Table` permet de définir le nom de la table pour une entité dans la base de données relationnelle en l'occurrence ici le nom sera « MUSCLE ».

`@Id` permet de définir la clé primaire d'une entité dans la base de données relationnelle.



`@GeneratedValue(strategy = GenerationType.IDENTITY)` permet de définir la stratégie de génération automatique de la clé primaire d'une entité dans la base de données relationnelle.

`@Column` permet de définir le nom et les propriétés d'une colonne dans une table de base de données relationnelle pour un attribut d'une classe annotée avec `@Entity`.

`@OneToMany` permet de définir une relation de type "un-à-plusieurs" entre deux entités dans une base de données relationnelle.

3. La magie de JPA

Grâce aux annotations et à l'API JPA, au lancement de l'application, la base de données se crée.

```
1 Hibernate:
2
3     create table abonnement (
4         abonnementid bigint not null auto_increment,
5         date_souscription date,
6         duree integer,
7         type varchar(255),
8         uid bigint,
9         primary key (abonnementid)
10    ) engine=MyISAM
11 Hibernate:
12
13     create table entrainement (
14         entrainemendid bigint not null auto_increment,
15         distance integer,
16         exercice varchar(255),
17         jour varchar(255),
18         muscle varchar(255),
19         poids integer,
20         recuperation integer,
21         repetition integer,
22         serie integer,
23         temps integer,
24         utilisateurid bigint,
25         primary key (entrainemendid)
26    ) engine=MyISAM
27 Hibernate:
28
29     create table exercice (
30         exerciceid bigint not null auto_increment,
31         description varchar(255),
32         muscleid bigint,
33         nom varchar(255),
34         photo varchar(255),
35         position varchar(255),
36         primary key (exerciceid)
37    ) engine=MyISAM
38 Hibernate:
39
40     create table galerie (
41         galerieid bigint not null auto_increment,
42         date date,
43         galeriefilename varchar(255),
44         uid bigint,
45         primary key (galerieid)
46    ) engine=MyISAM
```



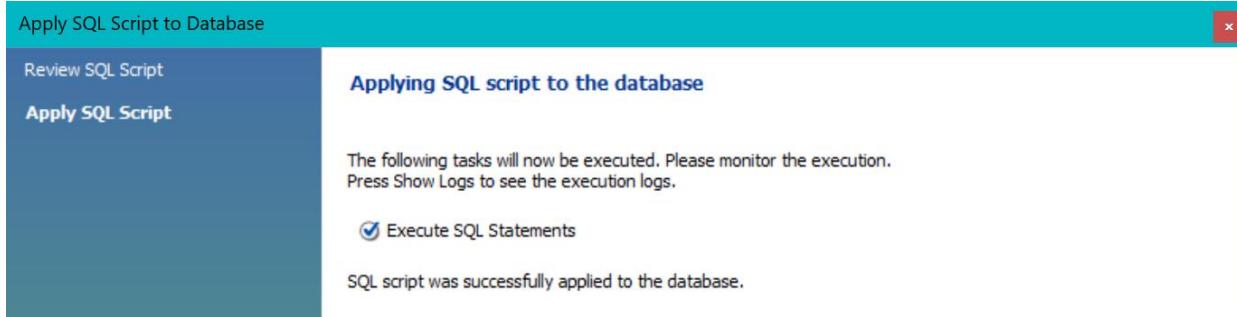
Sur notre logiciel MySQL WorkBench nous pouvons voir que la base de données a bien été créée.



4. Insertion des données

Les données peuvent être insérées directement via l'application MySQL Workbench en cliquant sur Apply.

Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:											
entraimentid	distance	exercice	jour	muscle	poids	recup	repetition	serie	temps	utilisateurid	
1	HULL	curl	Lundi	Bras	15	14	14	17	HULL	HULL	
22	HULL	Élévations latérales à la machine	Lundi	Épaules	99	HULL	99	99	HULL	HULL	
4	HULL	Soulevé de terre	Mardi	Dos	100	60	4	10	HULL	HULL	
5	HULL	Fentes	Mardi	Fesses	60	60	4	12	HULL	HULL	
25	HULL	Fentes	Vendredi	Fessiers	12	HULL	12	12	12	HULL	
7	HULL	Crunch	Mercredi	Pectoraux	99	HULL	12	4	HULL	HULL	
8	HULL	Crunch	Jeudi	Épaules	99	HULL	12	4	HULL	HULL	
24	HULL	Développé Arnold	Lundi	Épaules	13	HULL	13	13	HULL	HULL	
21	HULL	Curl à la barre	Lundi	Biceps	15	HULL	16	15	HULL	HULL	
20	HULL	Face pull	Mardi	Épaules	13	HULL	13	13	HULL	HULL	
23	HULL	Développé militaire	Lundi	Épaules	101	HULL	101	101	HULL	HULL	
**	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL	





5. Développer les composants dans le langage d'une base de données

L'utilisation d'un ORM facilite l'interaction entre une application et une base de données relationnelle en mappant les objets de l'application aux tables de la base de données. Il offre une abstraction de la base de données, permettant aux développeurs de se concentrer sur la logique métier de l'application plutôt que sur la gestion de la base de données. L'utilisation d'un ORM peut réduire le temps de développement en éliminant la nécessité d'écrire du code SQL.

Mais il est bien-sûr possible de développer sa base de données dans un langage de base de données.

Le langage SQL (Structured Query Language) :

SQL est un langage informatique normalisé servant à exploiter des bases de données relationnelles. La partie langage de manipulation des données de SQL permet de rechercher, d'ajouter, de modifier ou de supprimer des données dans les bases de données relationnelles.

Pour la création d'une table :

```
1  CREATE TABLE `entrainement` (
2      `entraimentid` bigint NOT NULL AUTO_INCREMENT,
3      `distance` int DEFAULT NULL,
4      `exercice` varchar(255) DEFAULT NULL,
5      `jour` varchar(255) DEFAULT NULL,
6      `muscle` varchar(255) DEFAULT NULL,
7      `poids` int DEFAULT NULL,
8      `recup` int DEFAULT NULL,
9      `repetition` int DEFAULT NULL,
10     `serie` int DEFAULT NULL,
11     `temps` int DEFAULT NULL,
12     `utilisateurid` bigint DEFAULT NULL,
13     PRIMARY KEY (`entraimentid`)
14 ) ENGINE=MyISAM AUTO_INCREMENT=26 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```



Pour insérer des données :

```
1  INSERT INTO `shapeappspring`.`entrainement`  
2    (`exercice`, `jour`, `muscle`, `poids`, `recup`, `repetition`, `serie`, `utilisateurid`)  
3  VALUES ('Fentes', 'Lundi', 'Épaules', '80', '1', '12', '4', '1');
```

Pour mettre à jour (UPDATE) une donnée :

```
1  UPDATE `shapeappspring`.`muscle` SET `nom` = 'Mollets' WHERE (`muscleid` = '10');
```

Pour supprimer (DELETE) une donnée :

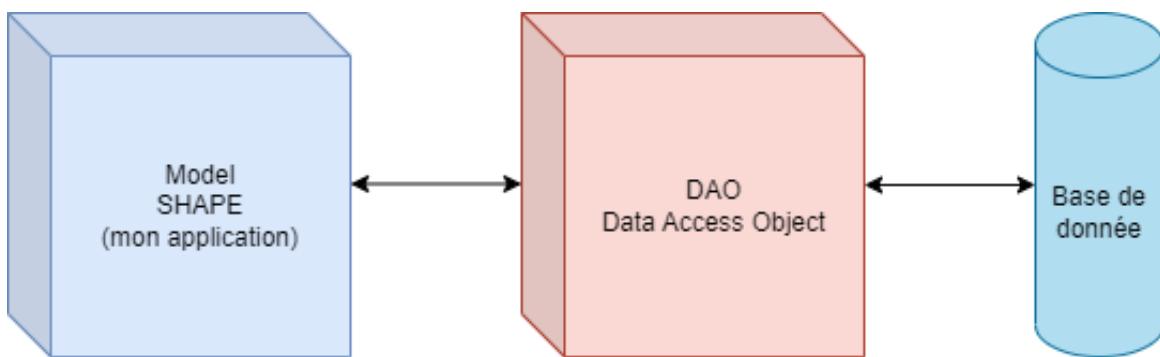
```
1  DELETE FROM `shapeappspring`.`muscle` WHERE (`muscleid` = '10');
```

C. Développer des composants d'accès aux données

1. Design Pattern: Data Access Object (DAO)

Lorsque nous concevons un programme, quel que soit le langage utilisé, nous avons souvent besoin de sauvegarder (sérialiser) nos objets dans une base de données (MySQL, Oracle, MongoDB...). Il existe de nombreux modèles de conception pour implémenter la persistance de la base de données.

Le fonctionnement du Design Pattern DAO, pour Data Access Object.



Buisness logic Layer ou BL : Une BL contient la logique métier de l'application. Elle sert principalement à exposer une API utilisée par la couche de présentation et à prendre à sa charge l'ensemble de l'intelligence fonctionnelle de l'application.



Data Access layer ou DAO : Ce pattern sert à éviter le mélange entre les objets métiers du programme et le code lié à la persistance de ces objets.

Data Storage : C'est la persistance des données, c'est-à-dire, l'archivage des données. Cela fait référence aux supports magnétiques, optiques ou mécaniques qui enregistrent et conservent des informations numériques pour des opérations en cours ou futures.

2. Quel est le but de ce design pattern ?

Le but de cette séparation est de rendre chaque couche indépendante pour faciliter la maintenance et les changements possibles de l'application.

Il offre une plus grande sécurité car l'accès à la base de données n'est autorisé que par cette couche.

La couche DAO contiendra l'implémentation des méthodes CRUD.

CRUD : (**create, read, update, delete**) (créer, lire, mettre à jour, supprimer) est un acronyme pour les façons dont on peut fonctionner sur des données stockées. C'est un moyen mnémotechnique pour les quatre fonctions de base du stockage persistant.

3. Crédit du package DAO

Chaque classe métier possède sa propre classe DAO :

```
✓ com.shape.ShapeAppSpring.dao
  > J AbonnementDao.java
  > J EntrainementDao.java
  > J ExerciceDao.java
  > J GalerieDao.java
  > J MensurationDao.java
  > J MuscleDao.java
  > J UserDao.java
```

Ici nous avons 7 classes :

- AbonnementDao
- EntrainementDao
- ExerciceDao
- GalerieDao
- MensurationDao
- MuscleDao
- UserDao



4. Présentation d'une classDao

Prenons comme exemple la classe EntrainementDao :

```
1 package com.shape.ShapeAppSpring.dao;
2
3 import java.util.List;
4
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.stereotype.Service;
7
8 import com.shape.ShapeAppSpring.domain.Entrainement;
9 import com.shape.ShapeAppSpring.repository.IEntrainementRepository;
10
11 @Service
12 public class EntrainementDao {
13
14     @Autowired
15     IEntrainementRepository entrainementRepository;
16
17
18     // Liste des Entrainements
19     public List<Entrainement> getEntrainements() {
20         return entrainementRepository.findAll();
21     }
22
23
24     // Save un Entrainement
25     public Entrainement saveEntrainement(Entrainement entrainement) {
26         return entrainementRepository.save(entrainement);
27     }
28
29
30     // get un Entrainement by ID
31     public Entrainement getEntrainementByID(Long entrainementId) {
32         return entrainementRepository.findById(entrainementId).get();
33     }
34
35
36     // Delete un Entrainement
37
38     public void deleteEntrainement(Entrainement entrainement) {
39         entrainementRepository.delete(entrainement);
40     }
41
42
43     // Update un Entrainement
44
45     public Entrainement updateEntrainement(Entrainement entrainement) {
46         return entrainementRepository.save(entrainement);
47     }
48
49
50 }
```



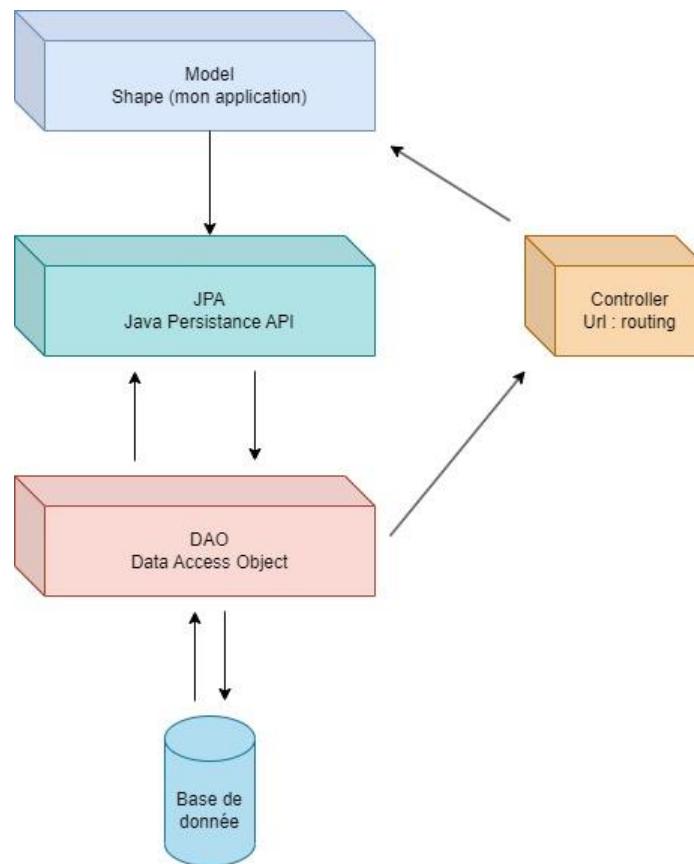
L'annotation `@Service` permet de déclarer un *bean* de service.

L'annotation `@Autowired` permet d'activer l'injection automatique de dépendance. Contrairement au mode autowiring en XML, il n'est pas possible de définir une stratégie à appliquer. Cette annotation peut être placée sur un constructeur, une méthode ou directement sur un attribut.

Nous pouvons voir que l'annotation `@Autowired` est au-dessus de `IEntrainementRepository`.

La classe `IEntrainementRepository` extends `JpaRepository` :

```
1 package com.shape.ShapeAppSpring.repository;
2
3 import org.springframework.data.jpa.repository.JpaRepository;
4
5 import com.shape.ShapeAppSpring.domain.Entrainement;
6
7 public interface IEntrainementRepository extends JpaRepository<Entrainement, Long>{
8
9 }
```





JPA est une spécification d'ORM (Object Relational Mapping) pour Java. Un ORM, comme son nom l'indique, permet de faire le lien entre le monde objet et le monde de la base de données relationnelle. Un ORM doit permettre de produire automatiquement les ordres SQL relatifs aux actions CRUD (Create/Read/Update/Delete) sur les objets et de les engager en base de données.

V. Sécurité et mise en place

A. Pourquoi sécuriser une application ?

La sécurité d'une application est un ensemble de mesures prises pour protéger l'application contre les menaces telles que les attaques de hackers, les fuites de données et les vulnérabilités.

Sécuriser une application est important pour plusieurs raisons :

- Protection des données personnelles :
L'application peut contenir des données personnelles sensibles telles que des informations de carte de crédit, des adresses e-mail et des numéros de téléphone. Si ces données tombent entre de mauvaises mains, elles peuvent être utilisées pour des activités malveillantes.
- Protection contre les attaques de hackers :
Les hackers peuvent utiliser des vulnérabilités dans l'application pour accéder à des données sensibles, perturber le fonctionnement normal de l'application ou compromettre la sécurité du système.
- Conformité aux réglementations en matière de protection des données :
De nombreuses réglementations ont été mises en place pour protéger les données personnelles des utilisateurs, telles que le RGPD en Europe. Les entreprises doivent se conformer à ces réglementations pour éviter les amendes et les poursuites judiciaires.
- Confiance des utilisateurs :
Les utilisateurs sont plus susceptibles d'utiliser une application s'ils ont confiance en sa sécurité. Une application sécurisée renforce la confiance des utilisateurs et améliore l'image de l'entreprise.

En somme, sécuriser une application est essentiel pour protéger les données personnelles des utilisateurs, prévenir les attaques de hackers, respecter les réglementations et gagner la confiance des utilisateurs.



B. Comment sécuriser une application ?

1. Comment faire ?

La sécurisation d'une application peut être réalisée à différents niveaux et il est important de mettre en place plusieurs couches de sécurité pour une protection efficace. Voici quelques mesures de sécurité que vous pouvez prendre pour sécuriser votre application :

- Authentification et autorisation :
Mettre en place un système d'authentification et d'autorisation pour contrôler l'accès à l'application et aux données sensibles. Utiliser des mots de passe forts, des codes PIN et des clés d'authentification à deux facteurs pour renforcer la sécurité.
- Cryptage des données :
Crypter les données sensibles telles que les informations de carte de crédit, les mots de passe et les informations personnelles pour éviter que les hackers ne puissent les lire même s'ils parviennent à les intercepter.
- Validation et désinfection des entrées utilisateur :
Valider et nettoyer les entrées utilisateur pour éviter les attaques d'injection SQL et les failles de sécurité.
- Mise à jour régulière des logiciels :
Mettre à jour régulièrement les logiciels de l'application pour corriger les failles de sécurité et les vulnérabilités.
- Protection contre les attaques de déni de service (DDoS) :
Mettre en place des mesures de protection contre les attaques DDoS pour garantir que l'application reste disponible et fonctionnelle même en cas d'attaques malveillantes.
- Audit des activités :
Surveiller les activités de l'application pour détecter les comportements suspects et les attaques potentielles.
- Éducation des utilisateurs :
Sensibiliser les utilisateurs à la sécurité de l'application en leur fournissant des informations sur les risques de sécurité et les bonnes pratiques à adopter pour protéger leurs données personnelles.

En somme, sécuriser une application nécessite une approche globale et la mise en place de mesures de sécurité à plusieurs niveaux pour protéger les données sensibles et garantir la sécurité de l'application.



2. Construire une application organisée en couches

Structurer une application en couches permet de mieux structurer le code et de faciliter la maintenance et l'évolution de l'application.

En organisant l'application en couches, chaque couche se concentre sur une responsabilité spécifique et offre des interfaces claires pour communiquer avec les autres couches. Cela permet d'isoler les fonctionnalités et de les développer indépendamment les unes des autres, ce qui facilite la maintenance, le débogage et la correction des bugs.

En outre, l'utilisation de couches permet de mieux respecter le principe de responsabilité unique, qui stipule qu'une classe ou une méthode ne doit avoir qu'une seule responsabilité. Cela rend le code plus facile à comprendre et à maintenir, car chaque classe ne s'occupe que d'une tâche spécifique.

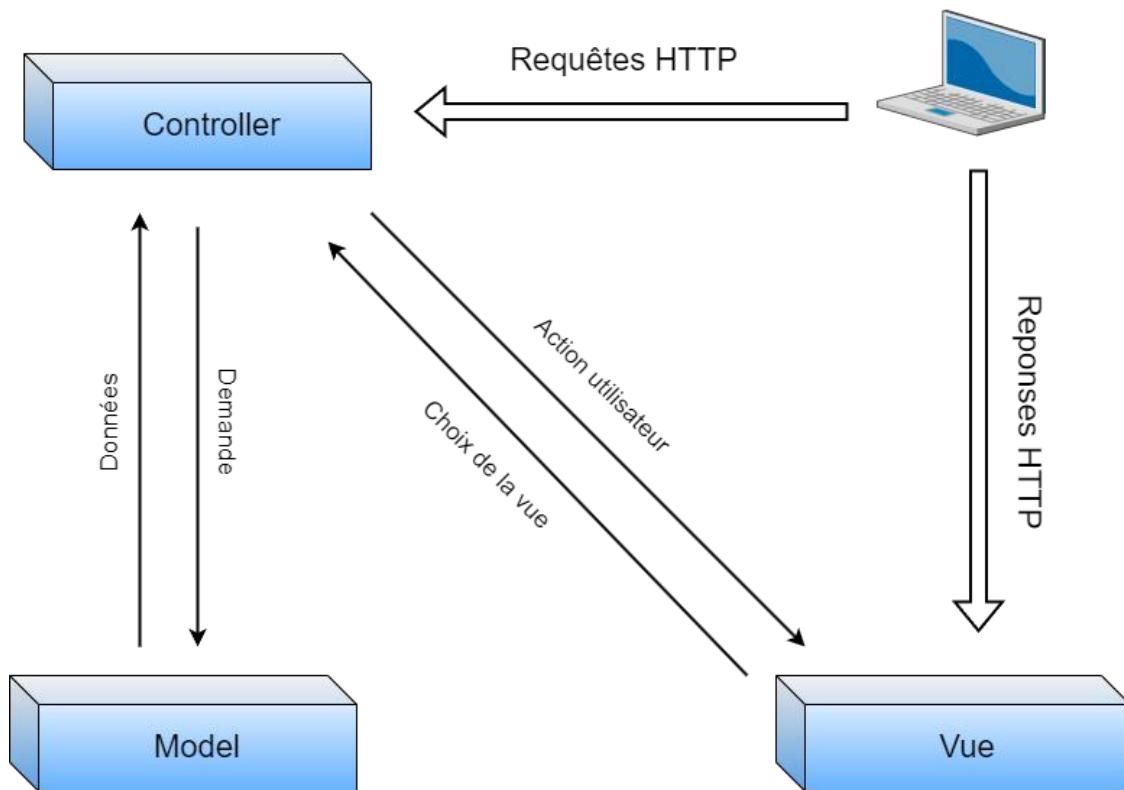
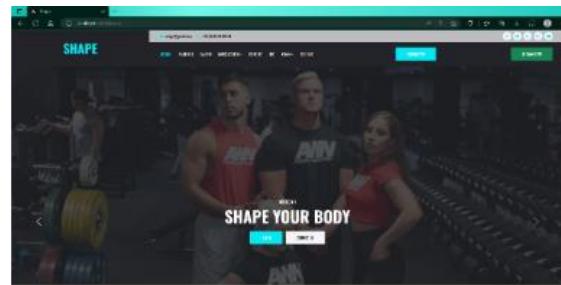
Enfin, une architecture en couches facilite l'ajout de nouvelles fonctionnalités et l'évolution de l'application, car chaque couche peut être développée et testée de manière indépendante. Cela permet d'ajouter de nouvelles fonctionnalités sans affecter le reste de l'application et de faire évoluer l'application de manière progressive et sûre.

En somme, construire une application organisée en couches permet de structurer le code de manière claire et de faciliter la maintenance, l'évolution et l'ajout de nouvelles fonctionnalités.

3. Le MVC

Pour le développement de notre application nous avons choisi le pattern MVC : Model-View-Controller.

Le MVC pattern est utilisé pour développer des applications logicielles. Il permet de séparer la logique de présentation, la logique de traitement et la gestion des données dans trois composants distincts, facilitant ainsi la conception, la maintenance et l'évolution de l'application.



- Le modèle (**Model**) représente les données et la logique métier de l'application. Il traite les données, effectue les calculs et interagit avec la base de données ou le système de stockage de l'application.
- La vue (**View**) est la partie de l'application qui est visible par l'utilisateur final. Elle affiche les données du modèle et fournit une interface utilisateur pour interagir avec l'application.
- Le contrôleur (**Controller**) reçoit les actions de l'utilisateur à partir de la vue et les traite en fonction de la logique métier de l'application. Il communique avec le modèle pour effectuer les opérations de traitement nécessaires et utilise la vue pour afficher les résultats.

La séparation des préoccupations rend le code plus facile à comprendre, à tester et à maintenir. Il est également plus facile d'ajouter de nouvelles fonctionnalités à

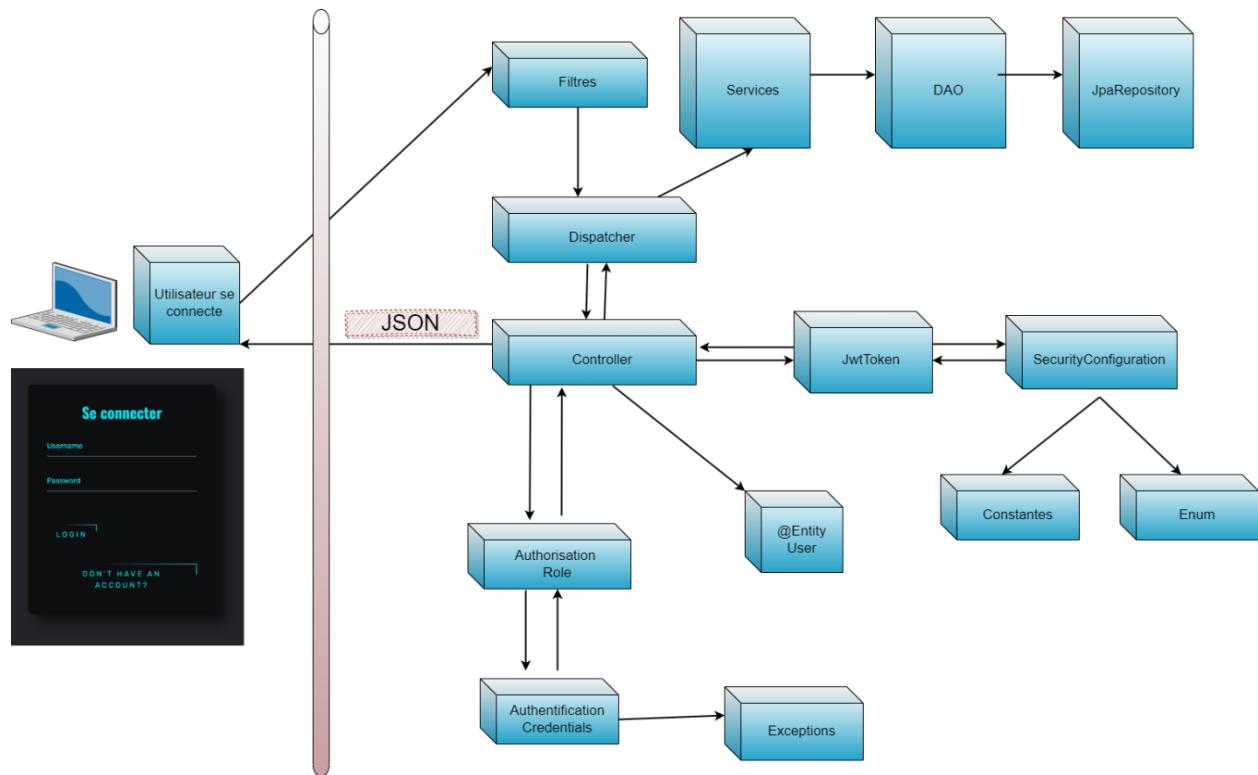


l'application ou de modifier l'interface utilisateur sans affecter la logique métier sous-jacente.

C. Mise en place de la sécurité dans le back-end

1. Schéma de la mise en place

La mise en place de la sécurité a été mise en place selon le schéma suivant :



2. Autorisation et authentification

La sécurité mise en œuvre par Spring Security repose sur deux étapes majeures : L'authentification et l'autorisation.

- L'authentification est le processus par lequel un utilisateur ou un système prouve son identité. Il s'agit généralement d'un processus de vérification de l'identifiant et du mot de passe de l'utilisateur. L'authentification permet donc de s'assurer que l'utilisateur est bien celui qu'il prétend être.



- L'autorisation, quant à elle, est le processus qui consiste à vérifier si un utilisateur ou un système est autorisé à accéder à une ressource ou à effectuer une action particulière. L'autorisation est généralement basée sur les droits et les permissions accordées à l'utilisateur ou au système.

Pour résumer, l'authentification permet de vérifier l'identité d'un utilisateur ou d'un système, tandis que l'autorisation permet de vérifier si cet utilisateur ou ce système a le droit d'accéder à une ressource ou d'effectuer une action particulière.

En d'autres termes, l'authentification répond à la question "Qui êtes-vous?", tandis que l'autorisation répond à la question "Qu'êtes-vous autorisé à faire ?".

3. Le corsFilter

```
@SpringBootApplication
public class ShapeAppSpringApplication {

    public static void main(String[] args) {
        SpringApplication.run(ShapeAppSpringApplication.class, args);
    }

    @Bean
    public BCryptPasswordEncoder bCryptPasswordEncoder() {
        return new BCryptPasswordEncoder();
    }

    @Bean
    public CorsFilter corsFilter() {
        UrlBasedCorsConfigurationSource urlBasedCorsConfigurationSource = new UrlBasedCorsConfigurationSource();
        CorsConfiguration corsConfiguration = new CorsConfiguration();
        corsConfiguration.setAllowCredentials(true);

        corsConfiguration.setAllowedOrigins(Arrays.asList("http://localhost:4200"));

        corsConfiguration.setAllowedHeaders(Arrays.asList(
            "Origin", "Access-Control-Allow-Origin",
            "Content-Type",
            "Accept", "Jwt-Token",
            "Authorization",
            "Origin, Accept",
            "X-Requested-With",
            "Access-Control-Request-Method",
            "Access-Control-Request-Headers"
        ));
        corsConfiguration.setExposedHeaders(Arrays.asList(
            "Origin",
            "Content-Type",
            "Accept",
            "Jwt-Token",
            "Authorization",
            "Access-Control-Allow-Origin",
            "Access-Control-Allow-Origin",
            "Access-Control-Allow-Credentials"
        ));
        corsConfiguration.setAllowedMethods(Arrays.asList("GET", "POST", "PUT", "DELETE", "OPTIONS"));
        urlBasedCorsConfigurationSource.registerCorsConfiguration("/**", corsConfiguration);
        return new CorsFilter(urlBasedCorsConfigurationSource);
    }
}
```

Le corsFilter (Cross-Origin Resource Sharing filter) est un mécanisme de sécurité utilisé dans les applications web pour limiter les requêtes cross-domaines, c'est-à-dire les requêtes provenant de domaines différents de celui sur lequel l'application est hébergée.

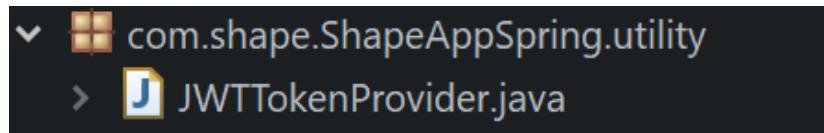


4. L'authentification et JwtToken

L'authentification est le processus de vérification de l'identité d'un utilisateur ou d'un système informatique, pour garantir que seuls les utilisateurs autorisés ont accès à des ressources protégées.

Un JWT est un jeton d'authentification qui contient des informations sur l'utilisateur, telles que son identifiant et ses autorisations. Le jeton est signé cryptographiquement pour garantir son authenticité, de sorte que le serveur peut vérifier que le jeton n'a pas été falsifié ou altéré.

Le générateur de token se situe dans le package utility



Voici la classe JWTTokenProvider.java

```
@Component
public class JWTTokenProvider {
    //Pour recuperer notre TOKEN il nous faut un cle secrete et une cle publique

    // Je fais appel a l'annotation value
    // ${} = je recuperne une valeur
    @Value("${jwt.secret}") // Il provient du fichier application.properties
    private String secret;

    public String generateJWTToken(UserPrincipal userPrincipal) {
        String[] claims = getClaimsFromUser(userPrincipal);

        return JWT.create().withIssuer(GET_ARRAYS LLC).withAudience(GET_ARRAYS ADMINISTRATION).withIssuedAt(new Date())
            .withSubject(userPrincipal.getUsername()).withArrayClaim(AUTHORITIES, claims)
            .withExpiresAt(new Date(System.currentTimeMillis() + EXPIRATION_TIME))
            .sign(HMAC512(secret.getBytes()));
    }
}
```

La méthode `generateJWTToken(UserPrincipal userPrincipal)` génère un JWT pour l'utilisateur en utilisant une clé secrète spécifiée dans le fichier « application.properties » voir page //.. Cette méthode utilise la bibliothèque JWT pour créer le jeton et le signer avec la clé secrète.

```
public List<GrantedAuthority> getAuthorities(String token) {
    String[] claims = getClaimsFromToken(token);
    return stream(claims).map(SimpleGrantedAuthority::new).collect(Collectors.toList()); // :: methode reference to
                                                                 // Java 8 using
                                                                 // Collection
}
```

La méthode `getAuthorities(String token)` extrait les autorisations d'un jeton donné. Cette méthode utilise également la bibliothèque JWT pour extraire les revendications du jeton et les convertir en une liste d'autorisations.



```
private String[] getClaimsFromToken(String token) {  
    JWTVerifier verifier = getJWTVerifier();  
  
    return verifier.verify(token).getClaim(AUTHORITIES).asArray(String.class);  
}
```

La méthode `getClaimsFromToken(String token)` extrait toutes les revendications d'un jeton donné en utilisant un objet `JWTVerifier`. Cette méthode utilise la bibliothèque `JWT` pour valider le jeton en vérifiant sa signature et en vérifiant que le jeton n'a pas expiré.

```
private JWTVerifier getJWTVerifier() {  
    I  
    JWTVerifier verifier;  
    try {  
        Algorithm algorithm = HMAC512(secret);  
        verifier = JWT.require(algorithm).withIssuer(GET_ARRAYS_LLC).build();  
    } catch(JWTVerificationException exception) {  
        throw new JWTVerificationException(TOKEN_CANNOT_BE_VERIFIED);  
    }  
    return verifier;  
}
```

La méthode `getJWTVerifier()` crée un objet `JWTVerifier` à partir de la clé secrète spécifiée dans le fichier `application.properties`.

```
private String[] getClaimsFromUser(UserPrincipal user) {  
    I  
    List<String> authorities = new ArrayList<>();  
    // Je parcours mes autorisations associées à mes utilisateurs  
    for (GrantedAuthority grantedAuthority : user.getAuthorities()) {  
        // et je récupère ces auth et j'ajoute à ma liste  
        authorities.add(grantedAuthority.getAuthority());  
  
    }  
    return authorities.toArray(new String[0]);  
}
```

La méthode `getClaimsFromUser(UserPrincipal user)` extrait les autorisations de l'objet `UserPrincipal` fourni et les retourne sous forme de tableau de chaînes.

```
//Get authentication when we verify the token  
public Authentication getAuthentication(String username, List<GrantedAuthority> authorities, HttpServletRequest request) {  
    UsernamePasswordAuthenticationToken usernamePassworAuthToken = new UsernamePasswordAuthenticationToken(username, null, authorities);  
    usernamePassworAuthToken.setDetails(new WebAuthenticationDetailsSource().buildDetails(request));  
    return usernamePassworAuthToken;  
}
```

La méthode `getAuthentication(String username, List<GrantedAuthority> authorities, HttpServletRequest request)` crée un objet `Authentication` à partir du nom d'utilisateur, des autorisations et de la demande HTTP fournies.



```
public boolean isTokenValid(String username, String token) {  
    JWTVerifier verifier = getJWTVerifier();  
    return StringUtils.isNotEmpty(username) && !isTokenExpired(verifier, token);  
}
```

La méthode `isTokenValid(String username, String token)` vérifie si le jeton fourni est valide en utilisant la méthode `getJWTVerifier()` pour créer un objet `JWTVerifier` et en vérifiant que le jeton n'a pas expiré.

```
private boolean isTokenExpired(JWTVerifier verifier, String token) {  
    Date expiration = verifier.verify(token).getExpiresAt();  
    // new Date() = je crée une nouvelle instance de date  
    return expiration.before(new Date());  
}
```

La méthode `isTokenExpired(JWTVerifier verifier, String token)` vérifie si le jeton a expiré en extrayant la date d'expiration du jeton et en la comparant à la date actuelle.

```
public String getSubject(String token) {  
    JWTVerifier verifier = getJWTVerifier();  
  
    return verifier.verify(token).getSubject();  
}
```

La méthode `getSubject(String token)` extrait le nom d'utilisateur associé à un jeton donné en utilisant un objet `JWTVerifier`. Cette méthode utilise la bibliothèque `JWT` pour valider le jeton et extraire le nom d'utilisateur.

5. Security Configuration

Permet de protéger les ressources contre les accès non autorisés.

```
@Configuration  
@Component  
@RequiredArgsConstructor  
public class SecurityConfiguration extends AbstractUserDetailsAuthenticationProvider {  
    private JwtAuthorizationFilter jwtAuthorizationFilter;  
    private JwtAccessDeniedHandler jwtAccessDeniedHandler;  
    private JwtAuthenticationEntryPoint jwtAuthenticationEntryPoint;  
    private UserDetailsService userDetailsService;  
    private BCryptPasswordEncoder bCryptPasswordEncoder;
```

La classe `SecurityConfiguration` est annotée avec `@Configuration` et `@Component`. `@Configuration` indique que la classe est une configuration Spring et `@Component` signifie que cette classe sera gérée par Spring.



Cette classe hérite de la classe `AbstractUserDetailsAuthenticationProvider` et définit plusieurs champs tels que `JwtAuthorizationFilter`, `JwtAccessDeniedHandler`, `JwtAuthenticationEntryPoint`, `UserDetailsService` et `BCryptPasswordEncoder`.

Dans le constructeur de la classe, ces champs sont initialisés via l'injection de dépendances, qui est gérée par Spring.

```
@Bean
public SecurityFilterChain filterChain (HttpSecurity http) throws Exception {
    http.csrf().disable()
        .authorizeHttpRequests()
        // .requestMatchers(POST, "/user/**")
        .requestMatchers("/**")
        .permitAll()
        .requestMatchers("/**")
        .authenticated()
        .anyRequest()
        .hasAnyRole("USER", "ADMIN")
        .and()
        /*.authenticationProvider(authenticationProvider)*/
        .httpBasic(withDefaults())
        .sessionManagement()
        .sessionCreationPolicy(STATELESS)
        .and()
        .exceptionHandling().accessDeniedHandler(jwtAccessDeniedHandler)
        .authenticationEntryPoint(jwtAuthenticationEntryPoint)
        .and()
        .addFilterBefore(jwtAuthorizationFilter, UsernamePasswordAuthenticationFilter.class);

    return http.build();
}
```

La méthode `filterChain()` est annotée avec `@Bean` et retourne un `SecurityFilterChain`, qui est une configuration de sécurité. Cette méthode utilise les méthodes de configuration de Spring Security pour définir les règles de sécurité, la gestion de la session, les autorisations d'accès pour les différentes URL et les rôles nécessaires pour y accéder. Cette méthode utilise également un filtre `jwtAuthorizationFilter` pour gérer l'authentification des utilisateurs.

```
@Override
protected void additionalAuthenticationChecks(UserDetails userDetails,
                                             UsernamePasswordAuthenticationToken authentication) throws AuthenticationException {
    if(authentication.getCredentials() == null || userDetails.getPassword() == null) {
        throw new BadCredentialsException("Credentials may not be null");
    }
    if(!bcryptPasswordEncoder.matches((String) authentication.getCredentials(), userDetails.getPassword())) {
        throw new BadCredentialsException("Invalid credentials");
    }
}

@Override
protected UserDetails retrieveUser(String username, UsernamePasswordAuthenticationToken authentication)
    throws AuthenticationException {

    return userDetailsService.loadUserByUsername(username);
}
```



Les méthodes `additionalAuthenticationChecks()` et `retrieveUser()` sont des méthodes héritées de la classe `AbstractUserDetailsAuthenticationProvider` qui sont utilisées pour vérifier les informations d'authentification fournies par l'utilisateur.

```
@Bean  
public AuthenticationManager authenticationManager(  
    HttpSecurity http,  
    BCryptPasswordEncoder bCryptPasswordEncoder,  
    UserDetailsService userDetailsService)  
throws Exception {  
    return http.getSharedObject(AuthenticationManagerBuilder.class)  
        .userDetailsService(userDetailsService)  
        .passwordEncoder(bCryptPasswordEncoder)  
        .and()  
        .build();  
}
```

Enfin, la méthode `authenticationManager()` est annotée avec `@Bean` et retourne un gestionnaire d'authentification pour l'application. Cette méthode utilise `AuthenticationManagerBuilder` pour configurer l'utilisateur et le mot de passe, ainsi que le `BCryptPasswordEncoder`.

6. BCryptPasswordEncoder

L'utilisation de `BCryptPasswordEncoder` pour stocker les mots de passe des utilisateurs dans une application est recommandée car cela permet de renforcer la sécurité de l'application.

Lorsqu'un utilisateur crée un compte et entre son mot de passe, le mot de passe est crypté à l'aide de `BCryptPasswordEncoder` avant d'être stocké dans la base de données. Lorsque l'utilisateur se connecte à nouveau, le mot de passe entré est crypté à l'aide de `BCryptPasswordEncoder` et comparé avec le mot de passe stocké dans la base de données pour vérifier s'ils correspondent.



Vue du register :

The screenshot shows the login interface of the Shape application. The left side is a white form with a dark border, and the right side is a dark background featuring a large, friendly cartoon octopus. The form contains fields for Prename, Name, Identifier, Email, Password, and Confirm password. Below the fields are two buttons: 'SENGRISTRER' (in blue) and 'Déjà enregistré ?' (in orange). A green success message at the bottom of the screen indicates account creation.

Login

Heureux de vous revoir!

Entrez votre identifiant et mot de passe pour accéder à l'interface administrateur.

Prenom
Adrien

Nom
Houen

Identifiant
Egon

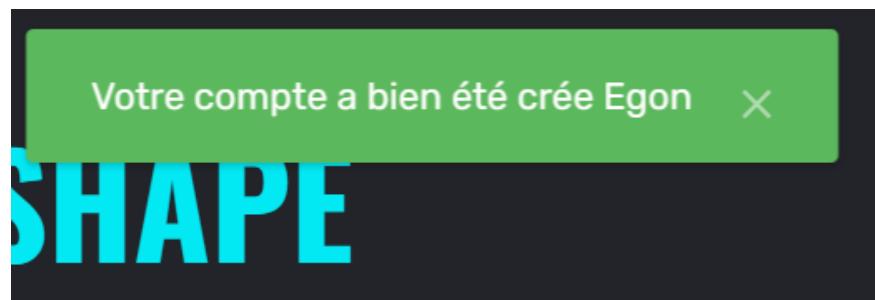
E-mail
adhouen@email.com

Password

Confirm password

Le mot de passe corespond

SENGRISTRER [Déjà enregistré ?](#)





Vue de la base de données avec le mot de passe crypté :

uid	authorities	authorizer	user_email	firstname	is_active	is_not_locked	join_date	last_login_date	last_login_date_display	lastname	user_password	profile_img	role	user_id	username
1	ROLE_USER	NULL	adhouen@yahoo.com	Adrien	1	1	2023-04-07 08:46:05	2023-04-07 08:46:05	2023-04-07 08:46:05	Houen	\$2a\$10\$cBRXfA3B/2zTSI/Vnq9iEOJqNEd5dYBBGtC5zjho43/5ZYwEBmwZe	/shape.png	ROLE_USER	2994064260	Egon
	ROLE_ADMIN	NULL													

user_password

\$2a\$10\$cBRXfA3B/2zTSI/Vnq9iEOJqNEd5dYBBGtC5zjho43/5ZYwEBmwZe
NULL

7. Les constantes - Autorisation suivant les rôles

```
com.shape.ShapeAppSpring.constant
  +-- Authority.java
  +-- EmailConstant.java
  +-- FileConstant.java
  +-- SecurityConstant.java
  +-- UserImplConstant.java
```

Pour les autorisations :

```
public class Authority {
    public static final String[] USER_AUTHORITIES= {"user:read"};
    public static final String[] HR_AUTHORITIES= {"user:read","user:update"};
    public static final String[] MANAGER_AUTHORITIES= {"user:read","user:update"};
    public static final String[] ADMIN_AUTHORITIES= {"user:read","user:create","user:update"};
    public static final String[] SUPER_ADMIN_AUTHORITIES= {"user:read","user:create","user:update","user:delete"};
}
```

Ce code définit une classe nommée "Authority" qui contient des constantes pour les autorités d'utilisateurs pour différents rôles.

Il y a cinq constantes pour les rôles suivants :

- "USER_AUTHORITIES" : autorise la lecture des données utilisateur.
- "HR_AUTHORITIES" : autorise la lecture et la mise à jour des données utilisateur.
- "MANAGER_AUTHORITIES" : autorise la lecture et la mise à jour des données utilisateur.
- "ADMIN_AUTHORITIES" : autorise la lecture, la création et la mise à jour des données utilisateur.



- "SUPER_ADMIN_AUTHORITIES" : autorise la lecture, la création, la mise à jour et la suppression des données utilisateur.

Ces constantes peuvent être utilisées pour définir des autorisations d'accès en fonction des rôles des utilisateurs dans une application. Par exemple, si un utilisateur est un "USER", il peut seulement lire des données, alors que si un utilisateur est un "ADMIN", il peut créer, mettre à jour et lire des données.

```
1 package com.shape.ShapeAppSpring.constant;
2
3 public class SecurityConstant {
4     public static final long EXPIRATION_TIME= 432_000_000; // 5 days expressed in milliseconds
5     public static final String TOKEN_PREFIX = "Bearer "; //No further check for verification when getting a token starting with Bearer
6     public static final String JWT_TOKEN_HEADER = "Jwt-Token";
7     public static final String TOKEN_CANNOT_BE_VERIFIED= "Token cannot be verified ";
8     public static final String GET_ARRAYS_LLC = "Get Arrays, SHAPE"; //SHAPE is the name your compagny
9     public static final String GET_ARRAYS_ADMINISTRATION = "User Management Portal";
10    public static final String AUTHORITIES = "authorities";
11    public static final String FORBIDDEN_MESSAGE = "You need to login to access this page";
12    public static final String ACCESS_DENIED_MESSAGE = "You do not have the permission to access this page";
13    public static final String OPTIONS_HTTP_METHOD = "OPTIONS";
14    //public static final String[] PUBLIC_URLS = {"user/login","user/register","user/resetpassword/**","user/image/**"};
15    public static final String[] PUBLIC_URLS = {"**"};
16
17 }
```

```
public class UserImplConstant {
    public static final String NEW_USER_PASSWORD = "New User password: ";
    public static final String USER_NOT_FOUND_BY_USERNAME = "User not found by username";
    public static final String USERNAME_ALREADY_EXIST = "Username already exist";
    public static final String NO_USER_FOUND_BY_USERNAME = "No user found by username";
    public static final String EMAIL_ALREADY_EXIST = "Email already exist";
    public static final String FOUND_USER_BY_USERNAME ="Returning find username";
    public static final String NO_USER_FOUND_BY_EMAIL = "Aucun utilisateur trouvé avec cet email";
    public static final CopyOption REPLACE_EXISTING = null;
}
```

8. Les énumérations et les rôles

```
public enum Role {
    ROLE_USER(USER_AUTHORITIES),
    ROLE_HR(HR_AUTHORITIES),
    ROLE_MANAGER(MANAGER_AUTHORITIES),
    ROLE_ADMIN(ADMIN_AUTHORITIES),
    ROLE_SUPER_ADMIN(SUPER_ADMIN_AUTHORITIES);

    private String[] authorities;

    Role(String...authorities){
        this.authorities = authorities;
    }
    public String[] getAuthorities() {
        return authorities;
    }
}
```



Ce code définit une énumération "Role" qui représente les différents rôles des utilisateurs dans une application.

Il y a cinq rôles possibles définis en tant que constantes dans l'énumération :

- ROLE_USER
- ROLE_HR
- ROLE_MANAGER
- ROLE_ADMIN
- ROLE_SUPER_ADMIN.

Cette énumération est utilisée pour définir des autorisations d'accès en fonction des rôles des utilisateurs dans une application. Par exemple, si un utilisateur est un "ROLE_USER", il aura seulement accès aux autorités accordées à ce rôle, tandis qu'un utilisateur qui est "ROLE_ADMIN" aura accès aux autorités accordées à ce rôle et celles accordées aux rôles inférieurs.

La méthode `getAuthorities()` est utilisée pour obtenir les autorités accordées à un rôle spécifique.

D. Mise en place de la sécurité dans le front-end

1. AuthenticationGuard

Ce code est une classe qui permet de protéger les routes qui nécessitent une authentification en vérifiant si l'utilisateur est connecté. Si l'utilisateur n'est pas connecté, il est redirigé vers la page de connexion et une notification d'erreur est affichée.

```
export class AuthenticationGuard implements CanActivate {
    constructor(private AuthenticationService:AuthenticationService,
    private router:Router,
    private notifier:NotifierService){}
    canActivate(
        route: ActivatedRouteSnapshot,
        state: RouterStateSnapshot): boolean{
        return this.isUserLoggedIn();
    }

    private isUserLoggedIn():boolean{
        if(this.AuthenticationService.isUserLoggedIn()){
            return true;
        }
        this.router.navigate(['/login']);
        //TODO send notification to user
        this.notifier.notify('error', 'Vous devez vous logger pour accéder à cette page');
        return false;
    }
}
```



2. Le service d'authentification

```
11 export class AuthenticationService {  
12     public host = environment.apiUrl;  
13     declare private token: string;  
14     declare private loggedInUsername: string;  
15     private jwtHelper = new JwtHelperService();  
16     declare private tok : string;  
17  
18     constructor(  
19         private http:HttpClient,  
20     ) {  
21     }  
22  
23     public login(user: User) : Observable<HttpResponse<User>>{  
24         return this.http.post<User>(`${this.host}/user/login`, user, {observe: 'response'});  
25     }  
26  
27     public register(user: User) : Observable<User>{  
28         return this.http.post<User>(`${this.host}/user/register`, user);  
29     }  
30  
31     public logOut() : void {  
32         this.token = '';  
33         this.loggedInUsername= '';  
34         this.tok ='';  
35         localStorage.removeItem('user');  
36         localStorage.removeItem('token');  
37         localStorage.removeItem('users');  
38     }  
39  
40     public saveToken(token: string): void {  
41         this.token = token;  
42         localStorage.setItem('token', token);  
43     }  
44  
45     public addUserToLocalCache(user: User): void {  
46         localStorage.setItem('user', JSON.stringify(user));  
47     }  
48  
49     public getUserFromLocalCache(): User {  
50         return JSON.parse(localStorage.getItem('user')!); /* || '{}' */  
51     }  
52  
53     /* Récupère le token dans le local storage */  
54     public loadToken() : void {  
55         this.token != localStorage.getItem('token'); /* || '{}' */  
56     }  
57  
58     /* Récupère le TOKEN de la méthode loadToken() pour pouvoir l'utiliser */  
59     public getToken() : string {  
60         return this.token;  
61     }  
62  
63     /* Vérifie si l'utilisateur est connecté */  
64     public isUserLoggedIn() : boolean {  
65         const tok =this.token;  
66         if(tok!=null && tok !='') {  
67             if(this.jwtHelper.decodeToken(tok).sub != null || '') {  
68                 if(!this.jwtHelper.isTokenExpired(tok)) {  
69                     this.loggedInUsername = this.jwtHelper.decodeToken(tok).sub;  
70                     console.log("Résultat this.loggedInUsername (decoded token) : [ " +  
71                     + this.loggedInUsername + " ] Authentication > isLoggedIn());  
72                     return true;  
73                 }  
74             }  
75         } else {  
76             this.logOut();  
77             return false ;  
78         }  
79         return false;  
80     }  
81  
82     }  
83  
84 }
```

Le constructeur de la classe injecte le service HttpClient pour effectuer des appels API.

La méthode "login" envoie une demande de connexion à l'API avec les informations d'utilisateur fournies et retourne une observable contenant la réponse de l'API.

La méthode "register" envoie une demande d'enregistrement d'utilisateur à l'API avec les informations d'utilisateur fournies et retourne une observable contenant la réponse de l'API.

La méthode "logOut" efface le token d'authentification, le nom d'utilisateur connecté et les données utilisateurs stockés dans le stockage local.

La méthode "saveToken" stocke le token d'authentification dans le stockage local.

La méthode "addUserToLocalCache" stocke les informations d'utilisateur dans le stockage local.

La méthode "getUserFromLocalCache" récupère les informations d'utilisateur à partir du stockage local.

La méthode "loadToken" récupère le token d'authentification à partir du stockage local.

La méthode "getToken" renvoie le token d'authentification actuellement stocké dans la classe.

La méthode "isUserLoggedIn" vérifie si l'utilisateur est connecté en vérifiant si le token d'authentification est stocké et valide en utilisant le service JwtHelper. Si l'utilisateur est connecté, la méthode renvoie "true" et met à jour le nom d'utilisateur connecté, sinon elle renvoie "false" et efface les données d'authentification stockées dans la classe.



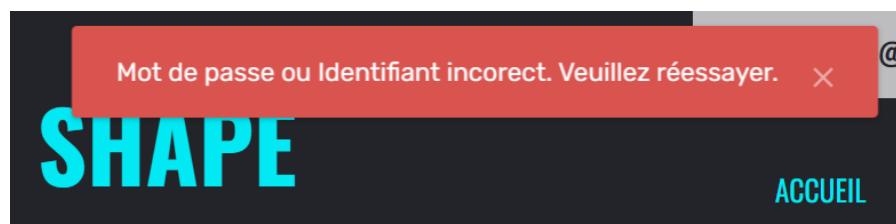
3. Component - Login

La méthode `onLogin()` dans le loginComponent :

```
public onLogin(user: User): void{
  this.showLoading = true;
  // console.log(user);

  this.subscriptions.push(this.AuthenticationService.login(user).subscribe(
    (response: HttpResponse<User>)=> {
      const token = response.headers.get(HeaderType.JWT_TOKEN);
      this.AuthenticationService.saveToken(token!);
      this.AuthenticationService.addUserToLocalCache(response.body!);
      this.router.navigateByUrl('/accueil');
      this.notifier.notify('success', 'Vous êtes connecté');
      this.showLoading = false;
    },
    (errorResponse: HttpErrorResponse) => {
      console.log(errorResponse);
      this.notifier.notify('error', 'Mot de passe ou Identifiant incorrect. Veuillez réessayer.');
      this.showLoading = false;
    }
  )
)}
```

Par exemple voici le résultat sur la page web si l'utilisateur se trompe de mot de passe ou d'identifiant :





VI. Les tests

A. Tester le back-end

1. Logiciel utilisé



JUnit est un framework open-source de test unitaire pour le langage de programmation Java. Il fournit un ensemble de bibliothèques et d'annotations pour écrire et exécuter des tests unitaires dans un environnement de développement intégré (IDE) tel qu'Eclipse, NetBeans ou IntelliJ IDEA.

2. Test unitaire

Les tests JUnit sont souvent exécutés en continu tout au long du cycle de développement, ce qui permet aux développeurs de détecter rapidement les erreurs et les bogues dans leur code. Les tests JUnit peuvent également être intégrés dans des outils de construction et d'intégration continue pour automatiser les tests lors de la compilation et de la livraison du code.

JUnit n'a pas été utilisé pour le développement de cette application mais a été utilisé pour le développement d'autre application.

Test Unitaire de la méthode factorielle, voici les résultats en utilisant JUnit :

En cas d'erreur :



B. Tester de front-end

Il existe 2 types de test pour le tester le front-end :

- Les tests d'intégration : ces tests permettent de vérifier que les différentes parties de l'application fonctionnent correctement ensemble.
- Les tests fonctionnels : ces tests permettent de vérifier que l'application fonctionne correctement du point de vue de l'utilisateur.

De plus, il est important de s'assurer que les tests sont exécutés dans différents navigateurs et sur différentes résolutions d'écran pour garantir une expérience utilisateur cohérente.

C. Tester la base de données

1. Logiciel utilisé



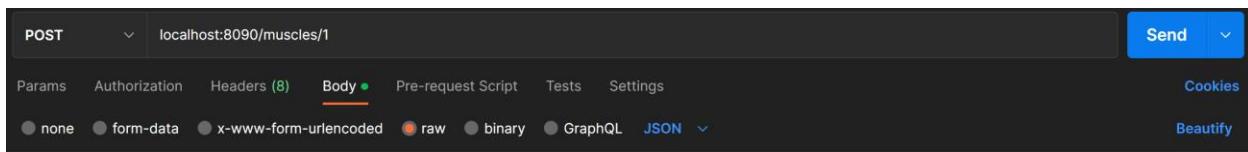
Postman est un logiciel gratuit qui vous permet d'effectuer des requêtes API sans coder. Les requêtes prennent la forme suivante : Verbe HTTP + URI + Version HTTP + Headers + Body facultatif. Les verbes HTTP sont des types d'actions que l'on peut faire lors de la formulation d'une requête.

2. Requête avec Postman

Faire des requêtes avec Postman permet de tester les API et de vérifier si elles fonctionnent correctement. On peut envoyer des requêtes HTTP GET, POST, PUT, DELETE, ..., pour tester différentes fonctionnalités de l'API et voir comment elles répondent. On peut également vérifier les codes de réponse, les en-têtes de réponse et les données de réponse pour s'assurer que l'API fonctionne correctement et retourne les résultats attendus.

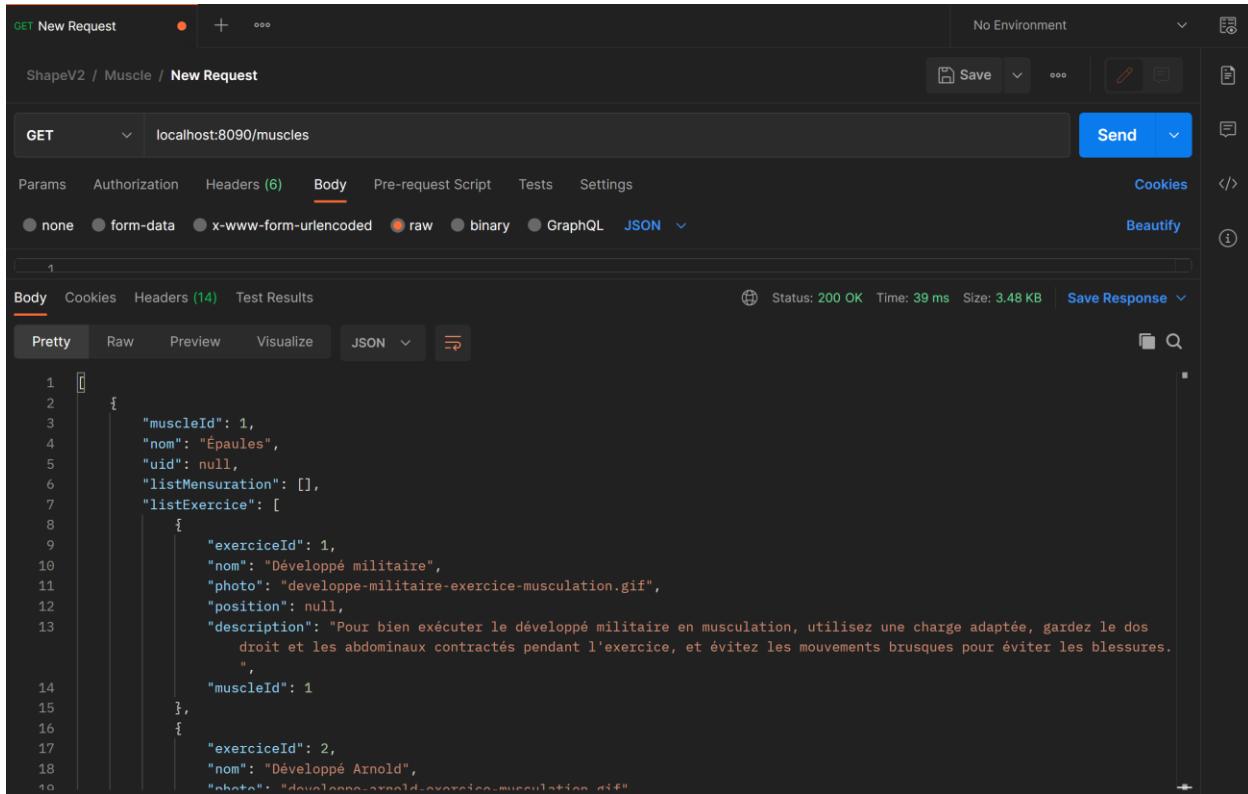


Méthode POST :

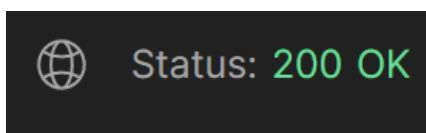


Méthode GET :

Ici nous testons la requête suivante : localhost :8090/muscles



```
1 [ { 2   "muscleId": 1, 3   "nom": "Épaules", 4   "uid": null, 5   "listMensuration": [], 6   "listExercice": [ 7     { 8       "exerciceId": 1, 9       "nom": "Développé militaire", 10      "photo": "developpe-militaire-exercice-musculation.gif", 11      "position": null, 12      "description": "Pour bien exécuter le développé militaire en musculation, utilisez une charge adaptée, gardez le dos droit et les abdominaux contractés pendant l'exercice, et évitez les mouvements brusques pour éviter les blessures.", 13      "muscleId": 1 14     }, 15     { 16       "exerciceId": 2, 17       "nom": "développé Arnold", 18       "photo": "developpe-arnold-exercice-musculation.gif" 19     } 20   ] } ]
```



Ceci est un code de réponse HTTP qui indique que la requête du client vers le serveur a été traitée avec succès. En d'autres termes, cela signifie que la demande a été comprise et que le serveur a renvoyé une réponse valide.

Le code "200" est l'un des codes de réponse HTTP les plus courants, il indique que la requête a été traitée avec succès et que le serveur a renvoyé les informations demandées dans le corps de la réponse. "OK" est simplement un message indiquant que tout s'est bien passé, sans erreur ni problème.



VII. Conclusion et évolution

A. Conclusion

1. Bilan du projet

Les objectifs initiaux du projet ont été respectés dans le temps imparti. N'ayant aucune connaissance en programmation en début de formation, je suis très satisfait du résultat obtenu ainsi que des fonctionnalités développées.

2. Difficultés rencontrées

Comme expliqué précédemment, en commençant cette formation je n'avais aucune connaissance en programmation. De ce fait, la première difficulté a été de me familiariser aux divers langages de programmations que ce soit pour le front-end ou encore pour le back-end.

L'ajout de certaines fonctionnalités ont nécessité plus de recherches que d'autres.

Voici les étapes suivies lors de difficultés rencontrées :

- Consulter la documentation fournie
- Visionner des vidéos explicatives sur internet
- Parcourir les forums d'entraide
- Ne pas hésiter à demander à ses camarades / collègues

B. Evolution

1. Evolution de l'application

L'application présentée à ce jour est fonctionnelle, cependant je souhaite mettre en place d'autres fonctionnalités telles que l'adhésion à des newsletters :



NEWSLETTER

INSCRIVEZ VOUS POUR RECEVOIR LA NEWSLETTER

Inscrivez vous dès maintenant pour recevoir prochainement notre newsletter.

Des recettes adaptées ainsi que des exercices personnalisés

Your Email

S'INSCRIRE

Ou encore la mise en place d'abonnement ou de succès :

OUR PLAN

CHOISIS TON ABONNEMENT ADAPTÉ

Gratuit

0 €

DÉCOUVERTE

Accès aux exercices
Prepare ton planning
Calcul ton BMI

[Enroll now](#)

6 mois

20 €

ACCÈS ILLIMITÉ

Accès aux exercices
Prepare ton planning
Calcul ton BMI
Télécharge tes photos
Suis ton évolution
Suivis personnalisé

[Enroll now](#)

12 Mois unlimited

35 €

ACCÈS ILLIMITÉ

Accès aux exercices
Prepare ton planning
Calcul ton BMI
Télécharge tes photos
Suis ton évolution
Suivis personnalisé

[Enroll now](#)

Bien sûr, plusieurs autres idées sont en train d'être développées. Une application est en perpétuelle évolution.



2. Perspective pour l'avenir

Cette formation de développeur d'application ainsi que la mise en place de ce projet m'ont permis d'acquérir les compétences nécessaires pour le monde du travail.

- Je vais continuer une veille technologique.
- Consolider mes connaissances en programmation.

Ainsi, à l'heure où ces lignes sont écrites, je suis actuellement en processus de recrutement dans une société située à Paris.