



# DOSSIER PROFESSIONNEL (DP)

*Nom de naissance*

HOUEN

*Nom d'usage*

Entrez votre nom d'usage ici.

*Prénom*

Adrien

*Adresse*

83 rue de Paris, 93260 Les Lilas

## Titre professionnel visé

Concepteur développeur d'applications

### MODALITE D'ACCES :

- Parcours de formation
- Validation des Acquis de l'Expérience (VAE)

# DOSSIER PROFESSIONNEL (DP)

## Présentation du dossier

Le dossier professionnel (DP) constitue un élément du système de validation du titre professionnel.  
**Ce titre est délivré par le Ministère chargé de l'emploi.**

Le DP appartient au candidat. Il le conserve, l'actualise durant son parcours et le présente **obligatoirement à chaque session d'examen.**

Pour rédiger le DP, le candidat peut être aidé par un formateur ou par un accompagnateur VAE.  
Il est consulté par le jury au moment de la session d'examen.

### Pour prendre sa décision, le jury dispose :

1. des résultats de la mise en situation professionnelle complétés, éventuellement, du questionnaire professionnel ou de l'entretien professionnel ou de l'entretien technique ou du questionnement à partir de productions.
2. du **Dossier Professionnel** (DP) dans lequel le candidat a consigné les preuves de sa pratique professionnelle.
3. des résultats des évaluations passées en cours de formation lorsque le candidat évalué est issu d'un parcours de formation
4. de l'entretien final (dans le cadre de la session titre).

*[Arrêté du 22 décembre 2015, relatif aux conditions de délivrance des titres professionnels du ministère chargé de l'Emploi]*

### Ce dossier comporte :

- ▶ pour chaque activité-type du titre visé, un à trois exemples de pratique professionnelle ;
- ▶ un tableau à renseigner si le candidat souhaite porter à la connaissance du jury la détention d'un titre, d'un diplôme, d'un certificat de qualification professionnelle (CQP) ou des attestations de formation ;
- ▶ une déclaration sur l'honneur à compléter et à signer ;
- ▶ des documents illustrant la pratique professionnelle du candidat (facultatif)
- ▶ des annexes, si nécessaire.

*Pour compléter ce dossier, le candidat dispose d'un site web en accès libre sur le site.*



<http://travail-emploi.gouv.fr/titres-professionnels>

## Sommaire

### Exemples de pratique professionnelle

#### Concevoir et développer des composants d'interface utilisateur en intégrant les recommandations de sécurité

- |  |       |
|--|-------|
| ▶ Maquetter une application .....                                      | p. 6  |
| ▶ Développer une interface utilisateur de type desktop .....           | p. 12 |
| ▶ Développer des composants d'accès aux données.....                   | p. 16 |
| ▶ Développer la partie front-end d'une interface utilisateur web ..... | p. 19 |
| ▶ Développer la partie back-end d'une interface utilisateur web .....  | p. 27 |

#### Concevoir et développer la persistance des données en intégrant les recommandations de sécurité

- |  |       |
|--|-------|
| ▶ Concevoir une base de données .....                                  | p. 35 |
| ▶ Mettre en place une base de données.....                             | p. 37 |
| ▶ Développer des composants dans le langage d'une base de données..... | p. 42 |

#### Concevoir et développer une application multicouche répartie en intégrant les recommandations de sécurité

- |  |       |
|--|-------|
| ▶ Collaborer à la gestion d'un projet informatique et à l'organisation de l'environnement de développement ..... | p. 44 |
| ▶ Concevoir une application.....   | p. 48 |
| ▶ Développer des composants métier .....   | p. 52 |
| ▶ Construire une application organisée en couches .....  | p. 55 |
| ▶ Développer une application mobile .....  | p. 57 |
| ▶ Préparer et exécuter les plans de tests d'une application.....   | p. 61 |
| ▶ Préparer et exécuter le déploiement d'une application.....   | p. 65 |

# DOSSIER PROFESSIONNEL (DP)

**Titres, diplômes, CQP, attestations de formation** (*facultatif*)

p.

**Déclaration sur l'honneur**

p.

69

**Documents illustrant la pratique professionnelle** (*facultatif*)

p.

**Annexes** (*Si le RC le prévoit*)

p.

# **EXEMPLES DE PRATIQUE PROFESSIONNELLE**

# DOSSIER PROFESSIONNEL (DP)

## Activité 1 Concevoir et développer des composants d'interface utilisateur en intégrant les recommandations de sécurité

*Exemple n°1 ▶ Maquetter une application*

### 1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Tout d'abord, j'ai mis en place le zoning, c'est à dire une schématisation grossière de ce que sera la future page web, on utilise des blocs pour déterminer où se trouveront les contenus et fonctionnalités.

Puis la maquette graphique ou le mockup.

### 2. Précisez les moyens utilisés :



Draw.io : permet de créer des diagrammes et des organigrammes personnalisables.



Figma : pour éditer des graphiques vectoriels et faire du prototypage

### 3. Avec qui avez-vous travaillé ?

J'ai travaillé en autonomie.

### 4. Contexte

Nom de l'entreprise, organisme ou association ▶ *Shape*

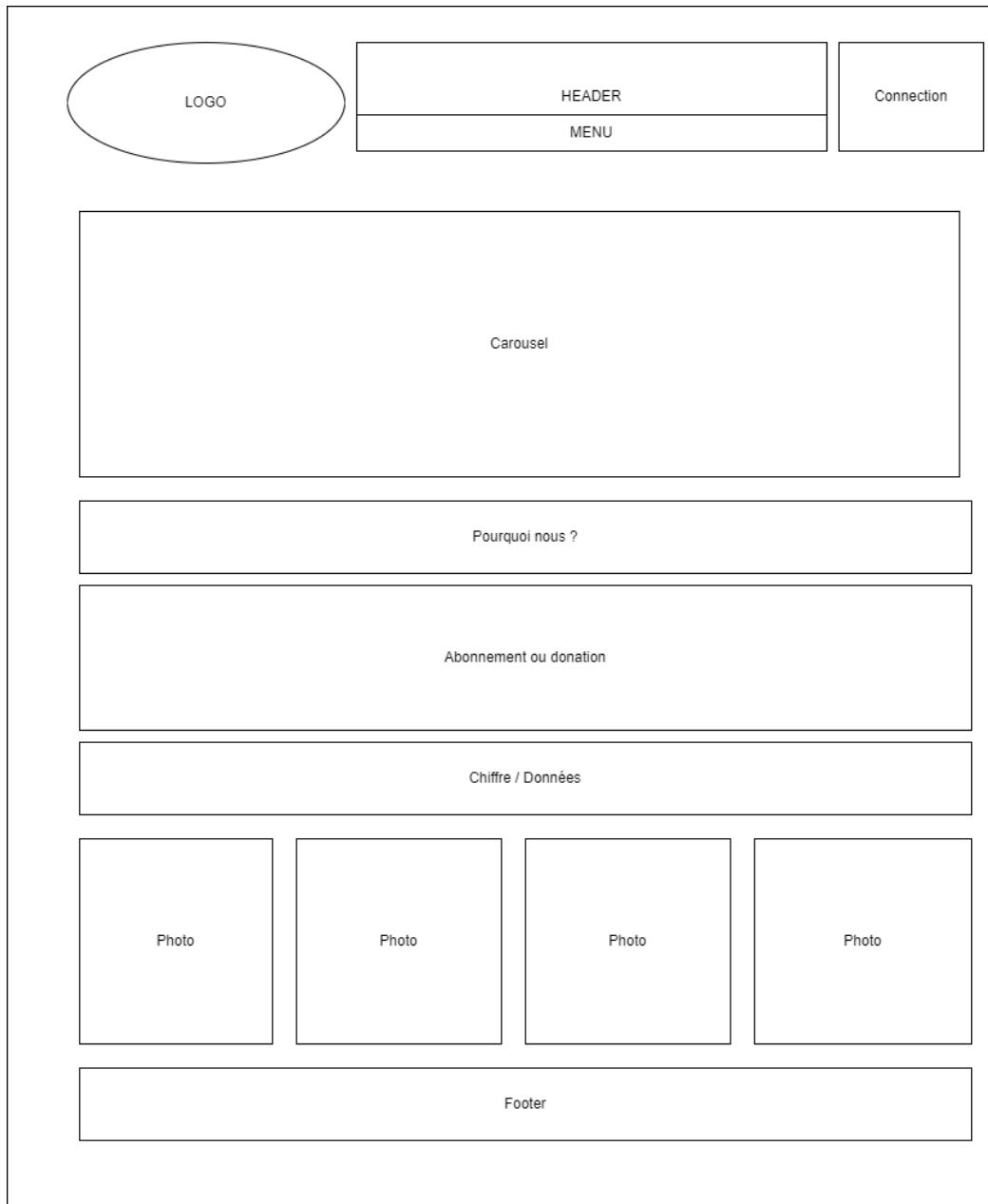
Chantier, atelier, service ▶ *Cliquez ici pour taper du texte.*

Période d'exercice ▶ Du : *01/01/2023* au : *03/03/2023*

# DOSSIER PROFESSIONNEL (DP)

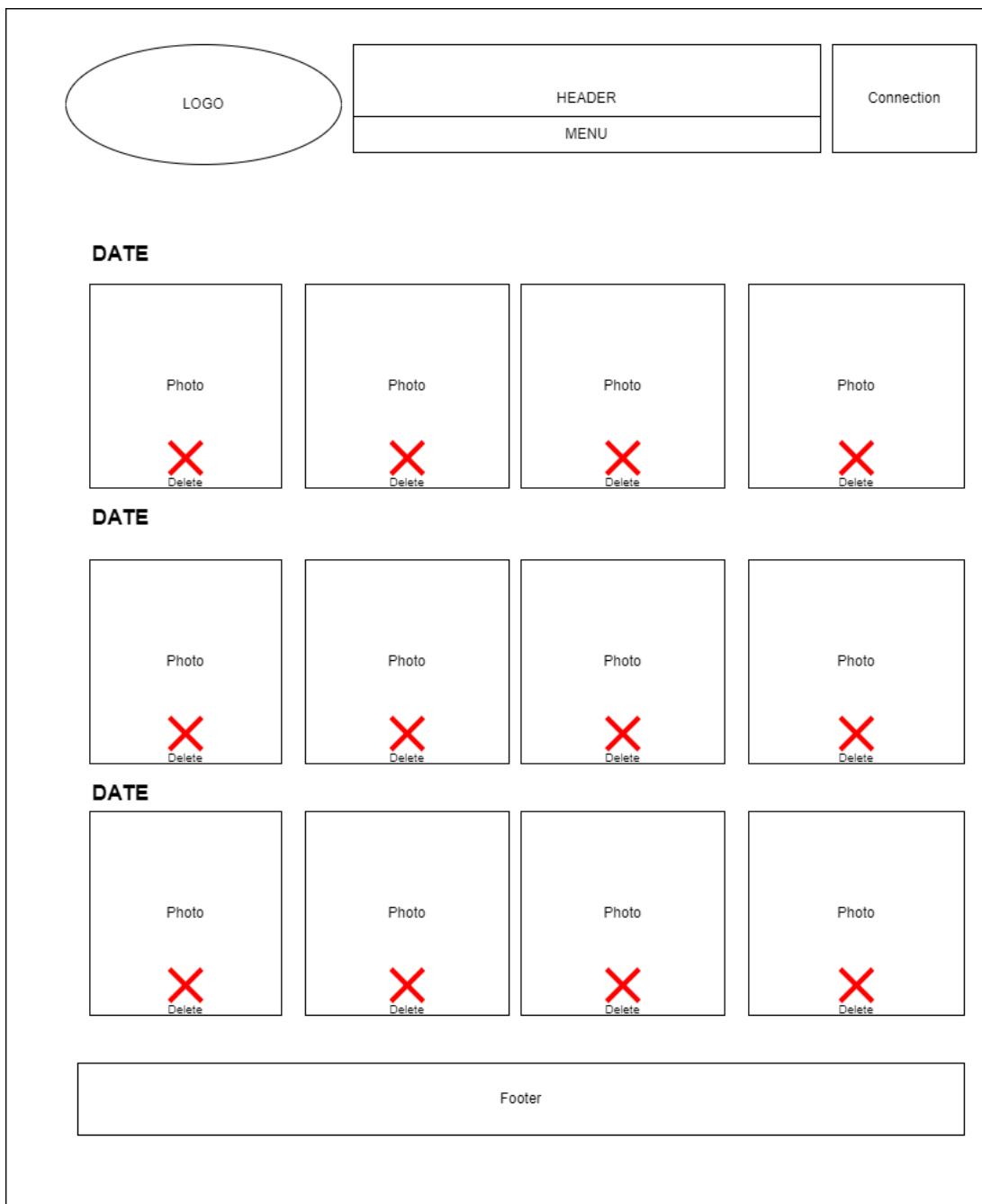
## 5. Informations complémentaires (*facultatif*)

Zoning : Accueil



# DOSSIER PROFESSIONNEL (DP)

Zoning : Galerie

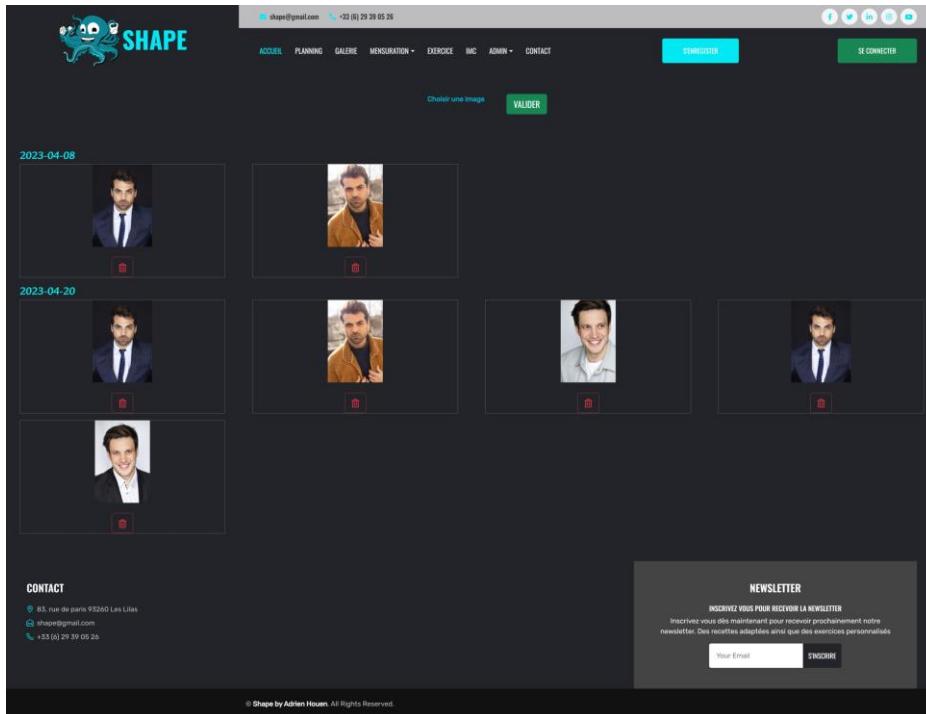


# DOSSIER PROFESSIONNEL (DP)

## Maquette graphique : calcul IMC

La maquette graphique pour le calcul IMC présente un formulaire sur un fond noir. En haut à gauche, il y a un logo d'un octopus bleu avec le mot "SHAPE". En haut à droite, il y a des icônes pour les réseaux sociaux et des boutons "SE CONNECTER" et "SE CONNECTER". Le formulaire est intitulé "Calcul ton IMC". Il contient des champs pour entrer la taille ("Taille en cm") et le poids ("Poids en kg"). Ensuite, il y a un bouton "VALIDER". Une fois validé, l'IMC est affiché ("IMC: 24.42 En bonne santé") avec une barre de couleur correspondante. En dessous, une question est posée : "Qu'est ce que l'indice de masse corporelle ? (IMC)". Une explication de l'IMC est fournie, mentionnant qu'il est calculé à partir de la taille et du poids selon la formule suivante :  $IMC = \text{poids en kg} / \text{taille}^2$ . Ensuite, il y a une section "IMC Résultat" qui indique que l'IMC est de 24.42, ce qui est "En bonne santé". En bas à gauche, il y a une section "CONTACT" avec l'adresse 83, rue de paris 93260 Les Lilas, l'e-mail shape@gmail.com et le numéro +33 (0) 29 39 05 26. En bas à droite, il y a une section "NEWSLETTER" avec un champ "Your Email" et un bouton "S'INSCRIRE".

## Maquette graphique : galerie



# DOSSIER PROFESSIONNEL (DP)

## Maquette graphique : contact

La maquette graphique de la page de contact du site Shape est une version mobile du site. En haut, le logo d'un octopus bleu avec l'inscription "SHAPE" est visible. En haut à droite, il y a des icônes pour les réseaux sociaux et des boutons "SE CONNECTER" et "SE CONNECTER". Le menu principal comprend "ACCUEIL", "PLANNING", "GALERIE", "MENSURATION", "EXERCICE", "INC", "ADMIN" et "CONTACT".

Le formulaire de contact est intitulé "CONTACTEZ-NOUS" et "GET IN TOUCH". Il demande l'adresse (83 rue de Paris, 93260 Les Lilas), le pays (France) et l'e-mail (shape@gmail.com). Des champs pour le nom, l'email, le téléphone et un message sont également présents, avec un bouton "SOUMETTRE" en bas.

En dessous du formulaire, il y a une carte Google Maps montrant l'emplacement de l'entreprise à 83 Rue de Paris, 93260 Les Lilas. La carte affiche diverses rues et bâtiments locaux.

À la base de la page, il y a deux sections : "CONTACT" (avec l'adresse, l'e-mail et le numéro de téléphone) et "NEWSLETTER" (avec un champ pour l'e-mail et un bouton "S'INSCRIRE").

## Maquette graphique : ajouter un entraînement

La maquette graphique de la page d'ajout d'un entraînement du site Shape est une version mobile du site. En haut, le logo d'un octopus bleu avec l'inscription "SHAPE" est visible. En haut à droite, il y a des icônes pour les réseaux sociaux et des boutons "SE CONNECTER" et "SE CONNECTER". Le menu principal comprend "ACCUEIL", "PLANNING", "GALERIE", "MENSURATION", "EXERCICE", "INC", "ADMIN" et "CONTACT".

Le formulaire d'ajout d'un entraînement est intitulé "Ajouter un entraînement". Il permet de sélectionner une journée de la semaine (Lundi, Mardi, Mercredi, Jeudi, Vendredi, Samedi, Dimanche). Une liste déroulante pour le muscle ("Épaules") montre divers exercices : Développé militaire, Développé Arnold, Face pull, Élévations latérales à la poule, etc. Des séries et des répétitions peuvent être définies pour chaque exercice.

À la base de la page, il y a deux sections : "CONTACT" (avec l'adresse, l'e-mail et le numéro de téléphone) et "NEWSLETTER" (avec un champ pour l'e-mail et un bouton "S'INSCRIRE").

# DOSSIER PROFESSIONNEL (DP)

Maquette graphique : éditer un entraînement

La maquette graphique montre une interface web pour la gestion d'un entraînement. En haut à gauche, il y a un logo d'un requin bleu avec le mot "SHAPE". En haut à droite, il y a des boutons pour "SE CONNECTER" et "SE DECONNECTER". Au centre, une fenêtre modale intitulée "Editor : Entrainement" est ouverte, montrant des paramètres tels que "series", "répétition", "seconde", "recup", "temps" et "distance". En bas de cette fenêtre, il y a un bouton "SAVE". En-dessous de la fenêtre, une section "Lundi" liste trois exercices pour les épaules :

Muscle	Exercice	Série	Répétition	Poids	Récupération	Temps	Distance	Edit	Remove
Épaules	Développé militaire	12	12	12 kg	sec				
Épaules	Poche pull	12	12	12 kg	sec				
Épaules	Per deck Inversé	12	12	30 kg	sec				

En bas à gauche, il y a une section "CONTACT" avec des informations telles que l'adresse 63, rue de Paris 93200 Les Lilas, l'e-mail shape@gmail.com et le numéro de téléphone +33 (6) 29 39 05 20. En bas à droite, il y a une section "NEWSLETTER" avec un champ pour entrer l'e-mail et un bouton "S'INSCRIRE".

# DOSSIER PROFESSIONNEL (DP)

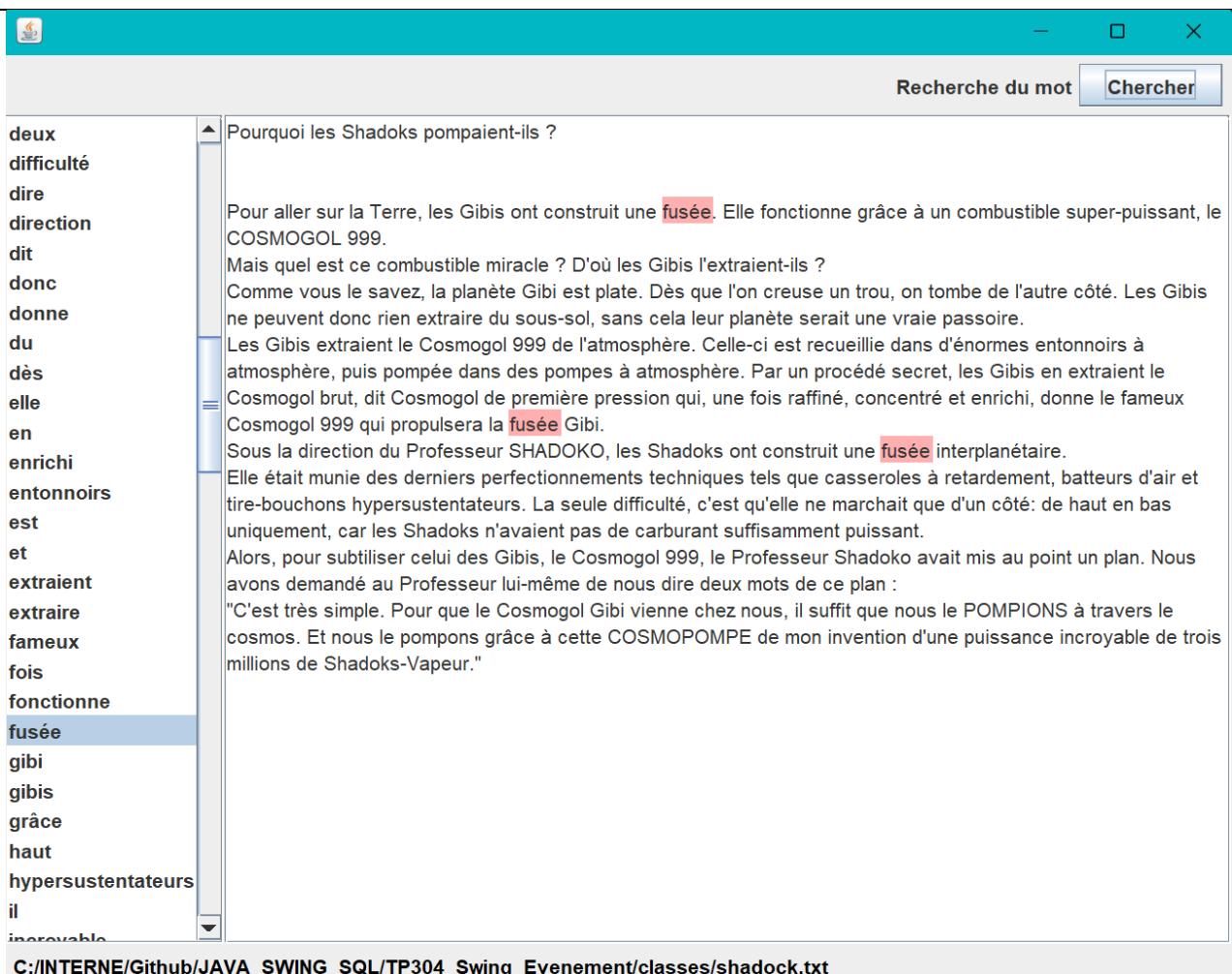
## Activité 1 Concevoir et développer des composants d'interface utilisateur en intégrant les recommandations de sécurité

*Exemple n°2 ▶ Développer une interface utilisateur de type desktop*

### 1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

En premier lieu, j'ai dû me familiariser avec WindowsBuilder qui permet la création d'interface Desktop.  
Puis j'ai implémenté mon code dans la partie source. La partie design m'a servi à la partie graphique de l'application.

#### I. Aperçu de l'application



# DOSSIER PROFESSIONNEL (DP)

## II. Source et Design

Grace à l'installation de Windows Builder j'ai accès à deux parties :

La partie source :

```
○ ○ ○

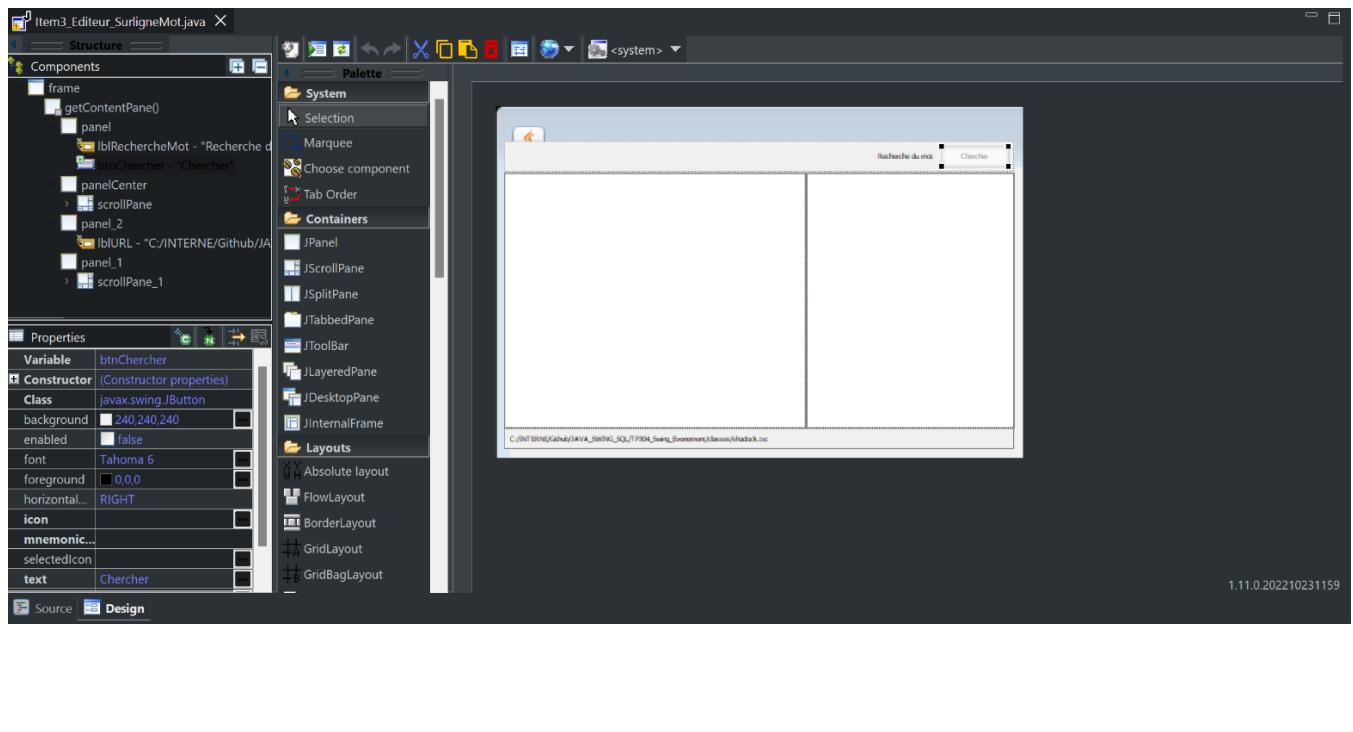
1  public class Item3_Editeur_SurligneMot {
2
3     private JFrame frame;
4
5     private JList list;
6     private JList list_1;
7     private JButton btnChercher;
8
9     private String texteAffiche = "";
10
11    private String texteDroite = "";
12
13    private JTextArea txtrDansLePlancher;
14
15    /**
16     * Launch the application.
17     */
18    public static void main(String[] args) {
19        EventQueue.invokeLater(new Runnable() {
20            public void run() {
21                try {
22                    Item3_Editeur_SurligneMot window = new Item3_Editeur_SurligneMot();
23                    window.frame.setVisible(true);
24                } catch (Exception e) {
25                    e.printStackTrace();
26                }
27            }
28        });
29    }
30
31    /**
32     * Create the application.
33     */
34    public Item3_Editeur_SurligneMot() {
35        initialize();
36    }
37
38    /**
39     * Initialize the contents of the frame.
40     */
41    private void initialize() {
42        frame = new JFrame();
43        frame.setBounds(100, 100, 450, 300);
44        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
45        frame.getContentPane().setLayout(new BorderLayout(0, 0));
46
47        JPanel panel = new JPanel();
48        frame.getContentPane().add(panel, BorderLayout.NORTH);
49        panel.setLayout(new FlowLayout(FlowLayout.RIGHT, 5, 5));
50
51        JLabel lblRechercheMot = new JLabel("Recherche du mot:");
52        lblRechercheMot.setHorizontalTextPosition(SwingConstants.RIGHT);
53        panel.add(lblRechercheMot);
54
55        btnChercher = new JButton("Chercher");
56
57        btnChercher.setEnabled(false);
58
59        btnChercher.setHorizontalTextPosition(SwingConstants.RIGHT);
60        panel.add(btnChercher);
61
62        JPanel panelCenter = new JPanel();
63        frame.getContentPane().add(panelCenter, BorderLayout.CENTER);
64        panelCenter.setLayout(new BoxLayout(panelCenter, BoxLayout.Y_AXIS));
65
66        JScrollPane scrollPane = new JScrollPane();
67        panelCenter.add(scrollPane);
68
69        JPanel panel_2 = new JPanel();
70        FlowLayout flowLayout = (FlowLayout) panel_2.getLayout();
71        flowLayout.setAlignment(FlowLayout.LEFT);
72        frame.getContentPane().add(panel_2, BorderLayout.SOUTH);
73
74        JLabel lblURL = new JLabel("New label");
75        panel_2.add(lblURL);
76
77        txtrDansLePlancher = new JTextArea();
78        String ficher = "shaddock.txt";
79        String way = getClass().getClassLoader().getResource(ficher).getPath().replaceAll("/", "\\");
80        Path path = Paths.get(way);
81
82        try (BufferedReader r = Files.newBufferedReader(path, StandardCharsets.UTF_8)) {
83            String line = null;
84            try {
85                while ((line = r.readLine()) != null) {
86                    texteDroite += line;
87                    texteAffiche += line + "\n";
88                }
89            } finally {
90                r.close();
91            }
92            /* Texte du label URL (si OK) */
93            lblURL.setForeground(Color.black);
94            lblURL.setText(way);
95        } catch (NoSuchFileException e) {
96            /* Texte du label URL (si NOK) */
97            lblURL.setForeground(Color.red);
98            lblURL.setText("Fichier inconnu");
99        } catch (IOException e) {
100            System.out.println(e.getMessage());
101        }
102    }
103}
```

# DOSSIER PROFESSIONNEL (DP)

```
215     private void surlignerBouton(ActionEvent e) {
216         Highlighter highlighter = txtrDansLePlancher.getHighlighter();
217         highlighter.removeAllHighlights();
218         getHighlight((String) list_1.getSelectedValue());
219     }
220 }
221
222 private void surlignerSouris(MouseEvent e) {
223     Highlighter highlighter = txtrDansLePlancher.getHighlighter();
224     highlighter.removeAllHighlights();
225     getHighlight((String) list_1.getSelectedValue());
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 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 */
247
```

Source Design

Et à la partie design :



# DOSSIER PROFESSIONNEL (DP)

## 2. Précisez les moyens utilisés :



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, bouton 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. Avec qui avez-vous travaillé ?

J'ai travaillé en autonomie.

## 4. Contexte

Nom de l'entreprise, organisme ou association ► *Shape*

Chantier, atelier, service ► *Cliquez ici pour taper du texte.*

Période d'exercice ► Du : *01/01/2023* au : *03/03/2023*

## 5. Informations complémentaires (facultatif)

# DOSSIER PROFESSIONNEL (DP)

## Activité 1

Concevoir et développer des composants d'interface utilisateur en intégrant les recommandations de sécurité

*Exemple n°3 ▶ Développer des composants d'accès aux données*

### 1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Création de la couche DAO :

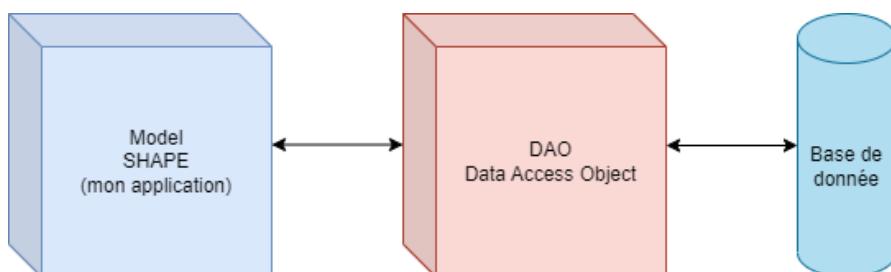
Data Access layer ou DAO : **Ce pattern sert à éviter le mélange entre les objets métiers du programme et le code lié à la persistance de ces objets.**

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

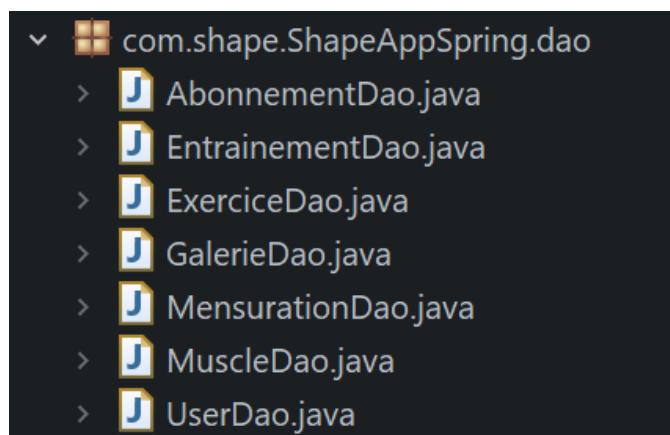
Il offre une plus grande sécurité car l'accès à la base de données n'est autorisé que par cette couche.

Cette 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.



Chaque classe métier possède sa propre classe DAO :



# DOSSIER PROFESSIONNEL (DP)

Prenons par exemple la classe EntrainementDao :

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

## 2. Précisez les moyens utilisés :



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.

## 3. Avec qui avez-vous travaillé ?

J'ai travaillé en autonomie.

# DOSSIER PROFESSIONNEL (DP)

## 4. Contexte

Nom de l'entreprise, organisme ou association ► **Shape**

Chantier, atelier, service ► *Cliquez ici pour taper du texte.*

Période d'exercice ► Du : **01/01/2023** au : **03/03/2023**

## 5. Informations complémentaires (*facultatif*)

# DOSSIER PROFESSIONNEL (DP)

## Activité-type 1

Concevoir et développer des composants d'interface utilisateur en intégrant les recommandations de sécurité

Partie n°4 ► Développer la partie front-end d'une interface utilisateur web

### 1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Création du dossier Shape puis dans ce dossier, lancer le CMD 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.
```

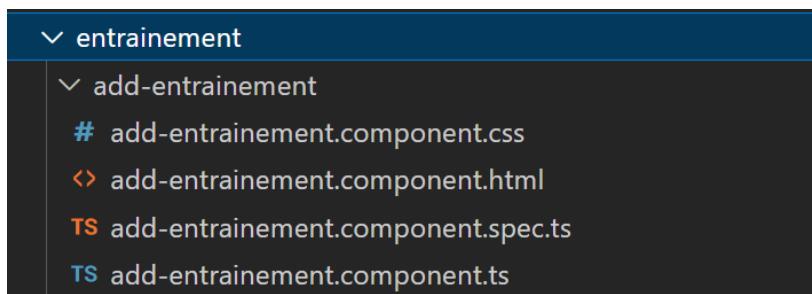
# DOSSIER PROFESSIONNEL (DP)

## I. Création de component

Nous devons commencer par créer un dossier components dans src. Pour chaque création d'un nouveau component, nous devons saisir la commande « `ng g c xxxxx` ». Par exemple, pour la création du component Entrainement nous devons saisir la commande « `ng g c Entrainement` ».

```
1 C:\INTERNE\IDK\shape\shapeApp\src\app\components>ng g c entrainement
CREATE src/app/components/entrainement/entrainement.component.html (27 bytes)
CREATE src/app/components/entrainement/entrainement.component.spec.ts (641 bytes)
CREATE src/app/components/entrainement/entrainement.component.ts (226 bytes)
CREATE src/app/components/entrainement/entrainement.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.



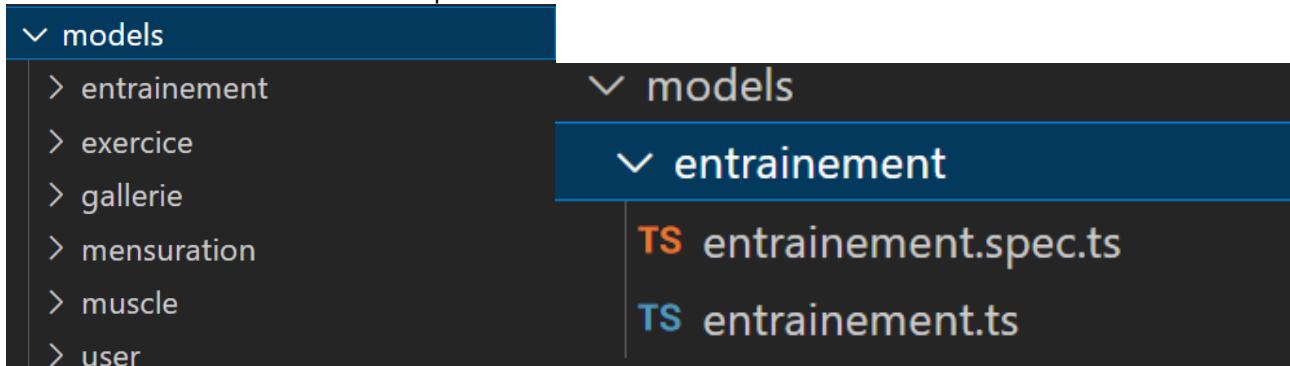
La création de ce nouveau component met à jour directement app.module.ts, EntrainementComponent est ajouté automatiquement.

```
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 { }
```

# DOSSIER PROFESSIONNEL (DP)

## II. Les models

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



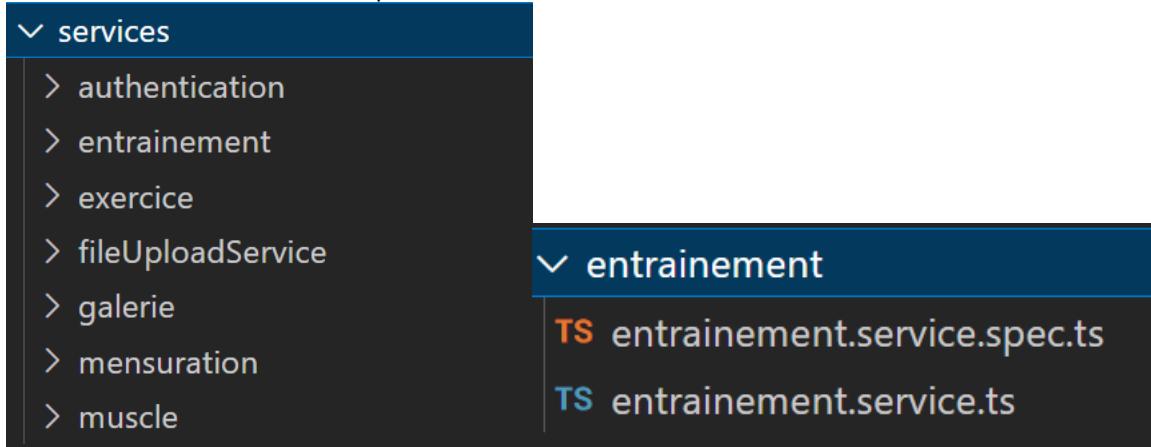
Par exemple pour le fichier entraînement.ts, nous avons :

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

# DOSSIER PROFESSIONNEL (DP)

## III. Les services

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



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

```
1 import { AppSettings } from '../../../../../settings/app.setting';
2 import { Observable } from 'rxjs';
3 import { Entrainement } from '../../../../../models/entraînement/entraînement';
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 /entraînements */
22   public getAllEntrainements() : Observable<Entrainement[]>{
23     return this.http.get<Entrainement[]>(`${this.host}/entraînements`)
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}/entraînements` , 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}/entraînements` , formData)
35   }
36
37   /* I delete un entraînement */
38   public deleteEntrainement(entrainementId: number) : Observable< Entrainement >{
39     return this.http.delete<Entrainement>(` ${this.host}/entraînements/${{entrainementId}}`);
40   }
41
42   public editEntrainement(id:number) {
43     return this.http.get(AppSettings.APP_URL+'/entraînements/'+id)
44   }
45
46   public updateEntrainement2(entrainement:Entrainement){
47     return this.http.put(AppSettings.APP_URL+'/entraînements/'+ entrainement.entrainementId, JSON.stringify(entrainement),this.httpOptions);
48   }
49
50 }
```

# DOSSIER PROFESSIONNEL (DP)

## IV. Le HTML

```
1 <br>
2 <br>
3 <h2 class="text-center">Ajouter un entraînement</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="radio">
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="exerciceEpaule_{{item.exerciceId}}" value="{{item.nom}}" formControlName="exercice">
45            <label for="exerciceEpaule_{{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 </form>
94 </div>
95 {{form.value | json}}
```

## **DOSSIER PROFESSIONNEL (DP)**

## V. 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
76 /* CSS les deux */
77 .container .choixExos{
78   display: flex;
79   align-items: center;
80 }
81
82
83 /* Button submit */
84 .container .buttonSubmit{
85   display: flex;
86   justify-content: center;
87 }
88
89
90
91
```

## VI. Tableau de routes

Ensuite, nous devons 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 nommées routes.

# DOSSIER PROFESSIONNEL (DP)

```
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/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}, // Contact
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 { }
```

## VII. Démarrage du serveur

Pour démarrer le serveur, je dois taper dans la console « ng serve -o », le « -o » cela permet d'ouvrir automatiquement 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.
```

# DOSSIER PROFESSIONNEL (DP)

## 2. Précisez les moyens utilisés :



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.



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



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.

## 3. Avec qui avez-vous travaillé ?

J'ai travaillé en autonomie.

## 4. Contexte

Nom de l'entreprise, organisme ou association ► *Shape*

Chantier, atelier, service ► *Cliquez ici pour taper du texte.*

Période d'exercice ► Du : *01/01/2023* au : *03/03/2023*

## 5. Informations complémentaires (*facultatif*)

# DOSSIER PROFESSIONNEL (DP)

## Activité-type 1

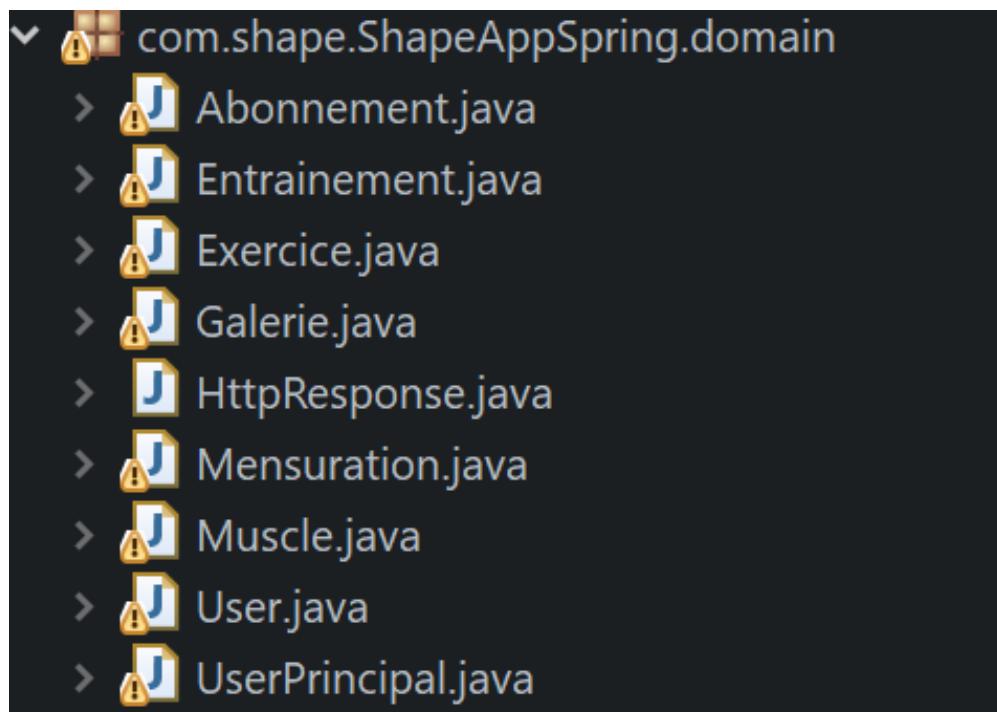
Concevoir et développer des composants d'interface utilisateur en intégrant les recommandations de sécurité

Partie n°5 ▶ Développer la partie back-end d'une interface utilisateur web

### 1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

#### I. Domain

Création du package : domain



# DOSSIER PROFESSIONNEL (DP)

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 appelés 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

# DOSSIER PROFESSIONNEL (DP)

## II. 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 :



# DOSSIER PROFESSIONNEL (DP)

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.

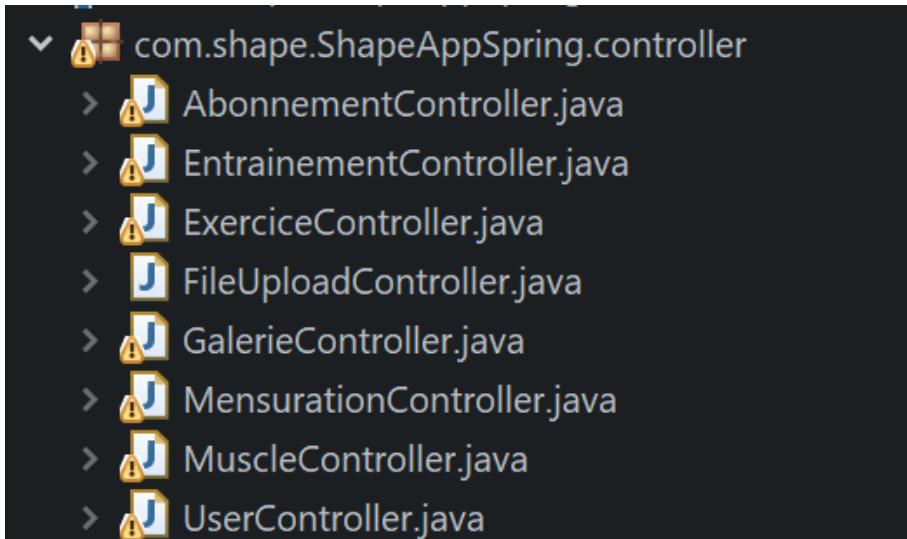
JPA est une spécification d'ORM (Object Relational Mapping) pour Java.

## III. 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 :



# DOSSIER PROFESSIONNEL (DP)

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.

# DOSSIER PROFESSIONNEL (DP)

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.

## IV. 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, vous pouvez 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 :



# DOSSIER PROFESSIONNEL (DP)

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 Entrainements
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    // get un Entrainement by ID
26    public Entrainement getEntrainementByID(Long entrainementId) {
27        return entrainementRepository.findById(entrainementId).get();
28    }
29
30
31    // Delete un Entrainement
32    public void deleteEntrainement(Entrainement entrainement) {
33        entrainementRepository.delete(entrainement);
34    }
35
36
37    // Update un Entrainement
38
39    public Entrainement updateEntrainement(Entrainement entrainement) {
40        return entrainementRepository.save(entrainement);
41    }
42
43
44
45
46}
```

## 2. Précisez les moyens utilisés :



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.

## 3. Avec qui avez-vous travaillé ?

J'ai travaillé en autonomie.

# DOSSIER PROFESSIONNEL (DP)

## 4. Contexte

Nom de l'entreprise, organisme ou association ► **Shape**

Chantier, atelier, service ► **Cliquez ici pour taper du texte.**

Période d'exercice ► Du : **01/01/2023** au : **03/03/2023**

## 5. Informations complémentaires (*facultatif*)

# DOSSIER PROFESSIONNEL (DP)

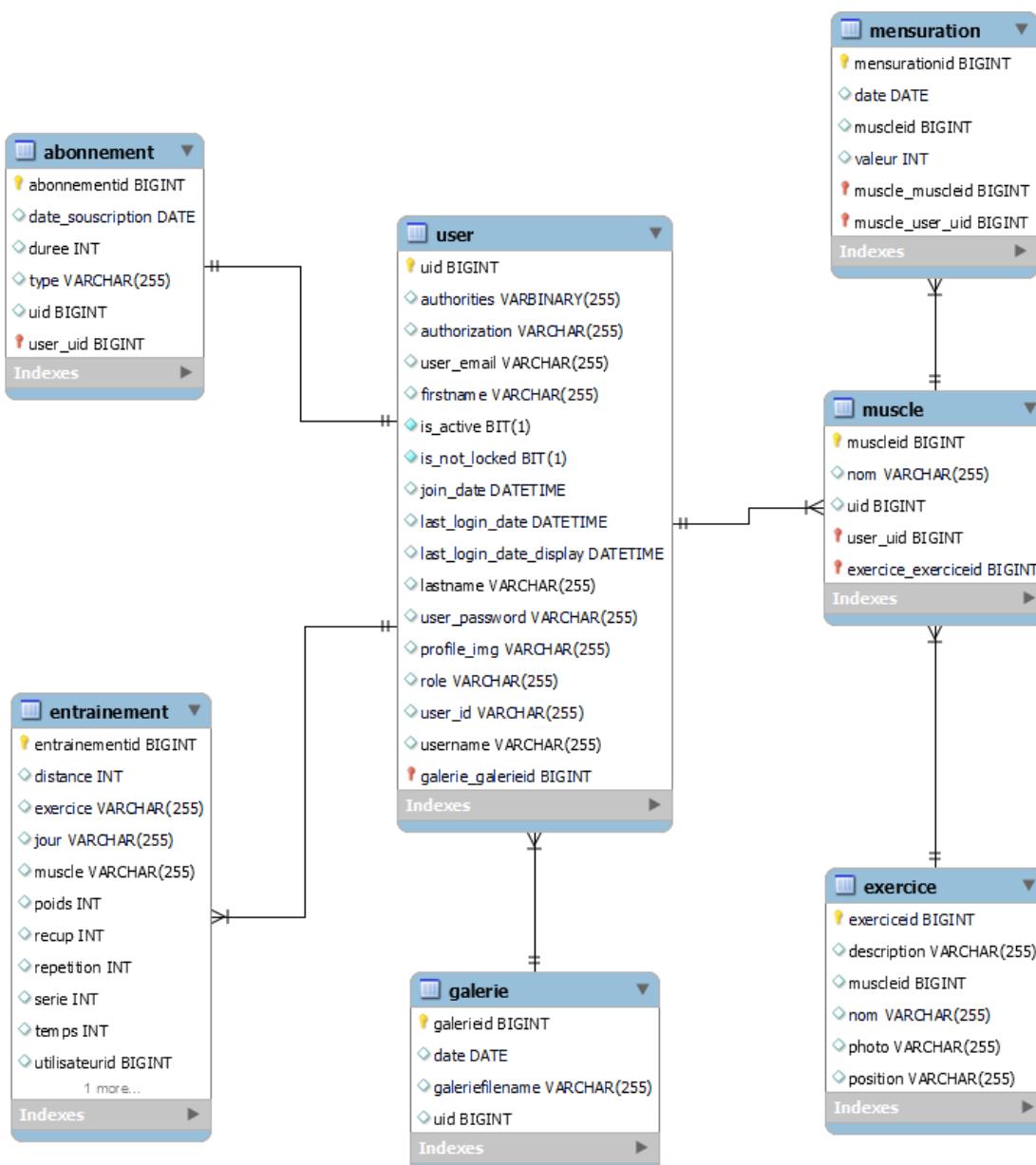
## Activité 2

Concevoir et développer la persistance des données en intégrant les recommandations de sécurité

Partie n°1 ► Concevoir une base de données

### 1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Concevoir un diagramme de base de données qui est une représentation visuelle des relations entre les entités dans une base de données.



# DOSSIER PROFESSIONNEL (DP)

## 2. Précisez les moyens utilisés :



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 diagrammes.



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.



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.

## 3. Avec qui avez-vous travaillé ?

J'ai travaillé en autonomie.

## 4. Contexte

Nom de l'entreprise, organisme ou association ► *Shape*

Chantier, atelier, service ► *Cliquez ici pour taper du texte.*

Période d'exercice ► Du : *01/01/2023* au : *03/03/2023*

## 5. Informations complémentaires (*facultatif*)

# DOSSIER PROFESSIONNEL (DP)

## Activité-type 2

**Concevoir et développer la persistance des données en intégrant les recommandations de sécurité**

**Partie n°2 ▶ Mettre en place une base de données**

### 1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

#### I. 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
```

#### II. 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.

# DOSSIER PROFESSIONNEL (DP)

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 @Entity
6 @Table(name="MUSCLE")
7 public class Muscle implements Serializable{
8
9     @Id
10    @GeneratedValue(strategy = GenerationType.IDENTITY)
11    @Column(name = "MUSCLEID")
12    private Long muscleId;
13
14    @Column(name = "NOM")
15    private String nom;
16    @Column(name = "UID")
17    private Long uid;
18
19    // ASSOCIATION
20    //Avec Mensuration
21    @OneToMany(fetch = FetchType.LAZY, mappedBy = "muscleId")
22    private List<Mensuration> listMensuration= new ArrayList<>();
23
24    //Avec Exercice
25    @OneToMany(fetch = FetchType.LAZY, mappedBy = "muscleId")
26    private List<Exercice> listExercice= new ArrayList<>();
27
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

# DOSSIER PROFESSIONNEL (DP)

## III. La magie de JPA

Grace 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         entrainementid 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 (entrainementid)
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
```

# DOSSIER PROFESSIONNEL (DP)

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



## IV. Insertion des données

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

The 'entrainement' table grid shows the following data:

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

The 'Apply SQL Script to Database' dialog box shows the following details:

- Review SQL Script
- Apply SQL Script
- Applying SQL script to the database
- The following tasks will now be executed. Please monitor the execution.  
Press Show Logs to see the execution logs.
- Execute SQL Statements
- SQL script was successfully applied to the database.

# DOSSIER PROFESSIONNEL (DP)

## 2. Précisez les moyens utilisés :



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.

## 3. Avec qui avez-vous travaillé ?

J'ai travaillé en autonomie.

## 4. Contexte

Nom de l'entreprise, organisme ou association ► *Shape*

Chantier, atelier, service ► *Cliquez ici pour taper du texte.*

Période d'exercice ► Du : *01/01/2023* au : *03/03/2023*

## 5. Informations complémentaires (*facultatif*)

# DOSSIER PROFESSIONNEL (DP)

## Activité 2

Concevoir et développer la persistance des données en intégrant les recommandations de sécurité

Partie n°3 ▶ Développer des composants dans le langage d'une base de données

### 1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

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      `entrainementid` 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 (`entrainementid`)
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');
```

# DOSSIER PROFESSIONNEL (DP)

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');
```

## 2. Précisez les moyens utilisés :



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.

Langage : **SQL**

## 3. Avec qui avez-vous travaillé ?

J'ai travaillé en autonomie

## 4. Contexte

Nom de l'entreprise, organisme ou association ► *Shape*

Chantier, atelier, service ► *Cliquez ici pour taper du texte.*

Période d'exercice ► Du : *01/01/2023* au : *03/03/2023*

## 5. Informations complémentaires (facultatif)

# DOSSIER PROFESSIONNEL (DP)

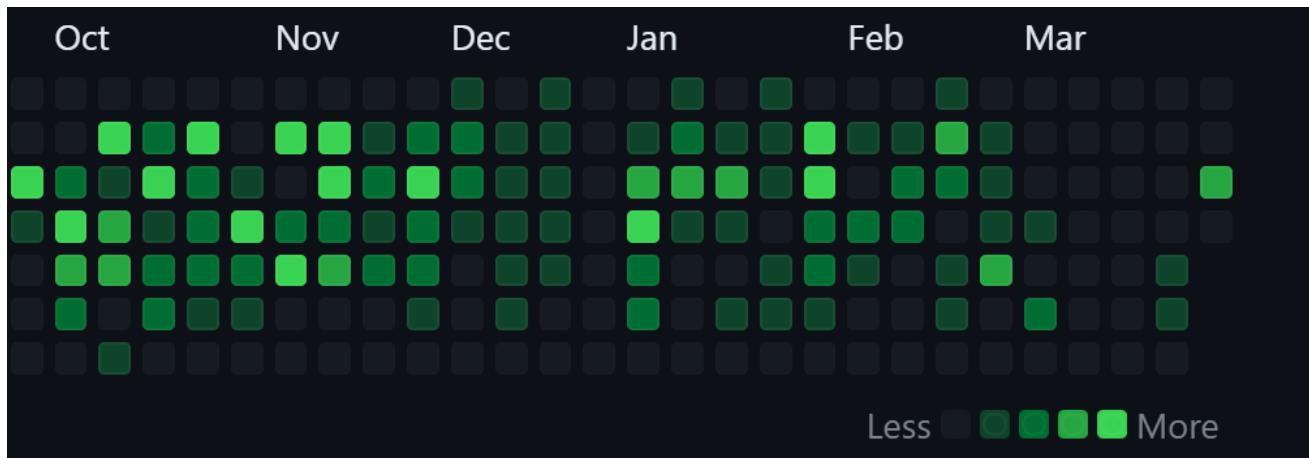
## Activité 3 Concevoir et développer une application multicouche répartie en intégrant les recommandations de sécurité

**Partie n°1 ▶** Collaborer à la gestion d'un projet informatique et à l'organisation de l'environnement de développement

### 1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

#### I. Utilisation de GitHub

- GitHub permet télécharger les modifications locales enregistrées dans votre 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.

# DOSSIER PROFESSIONNEL (DP)

## 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).

# DOSSIER PROFESSIONNEL (DP)

**git commit -m « commentaire »** : La commande est utilisée pour enregistrer les modifications apportées à votre 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 interface. At the top, it says "main" and "ShapeWithSecu / ShapeAppSpringWithSecu / src / main / java / com / shape / ShapeAppSpring /". Below this is a list of files and their commit history:

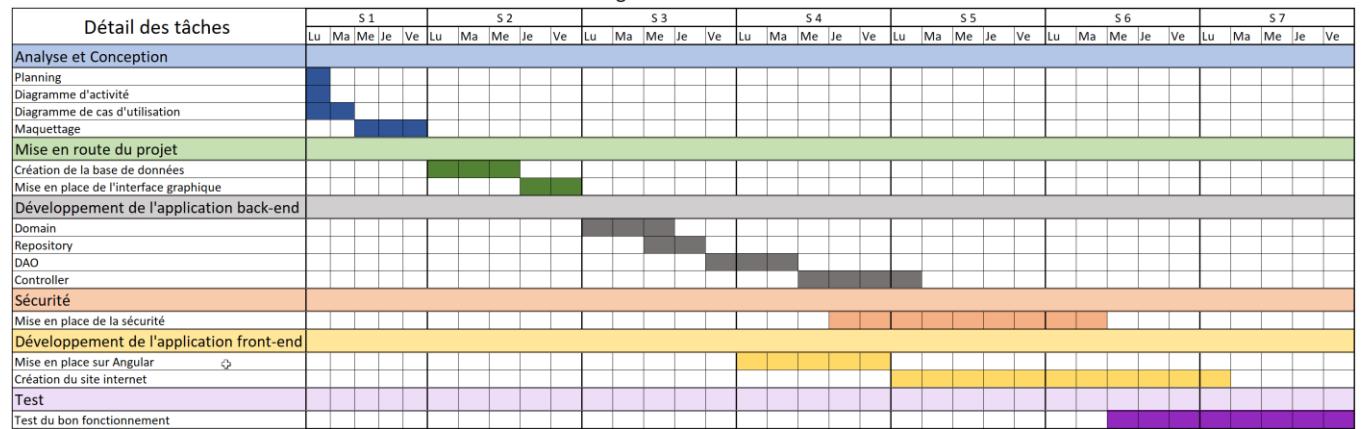
File	Commit Message	Date
configuration	secu	yesterday
constant	secu	yesterday
controller	secu	yesterday
dao	secu	yesterday
domain	secu	yesterday
enumeration	secu	yesterday
exception/domain	secu	yesterday
listener	secu	yesterday
repository	secu	yesterday
resource	secu	yesterday
service	secu	yesterday
utility	secu	yesterday
ShapeAppSpringApplication.java	secu	yesterday

At the bottom left, there is a "Give feedback" button.

## II. 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.

Diagramme de GANTT



# DOSSIER PROFESSIONNEL (DP)

## 2. Précisez les moyens 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.

## 3. Avec qui avez-vous travaillé ?

J'ai travaillé en autonomie.

## 4. Contexte

Nom de l'entreprise, organisme ou association ► *Shape*

Chantier, atelier, service ► *Cliquez ici pour taper du texte.*

Période d'exercice ► Du : *01/01/2023* au : *03/03/2023*

## 5. Informations complémentaires (*facultatif*)

# DOSSIER PROFESSIONNEL (DP)

## Activité 3

Concevoir et développer une application multicouche répartie en intégrant les recommandations de sécurité

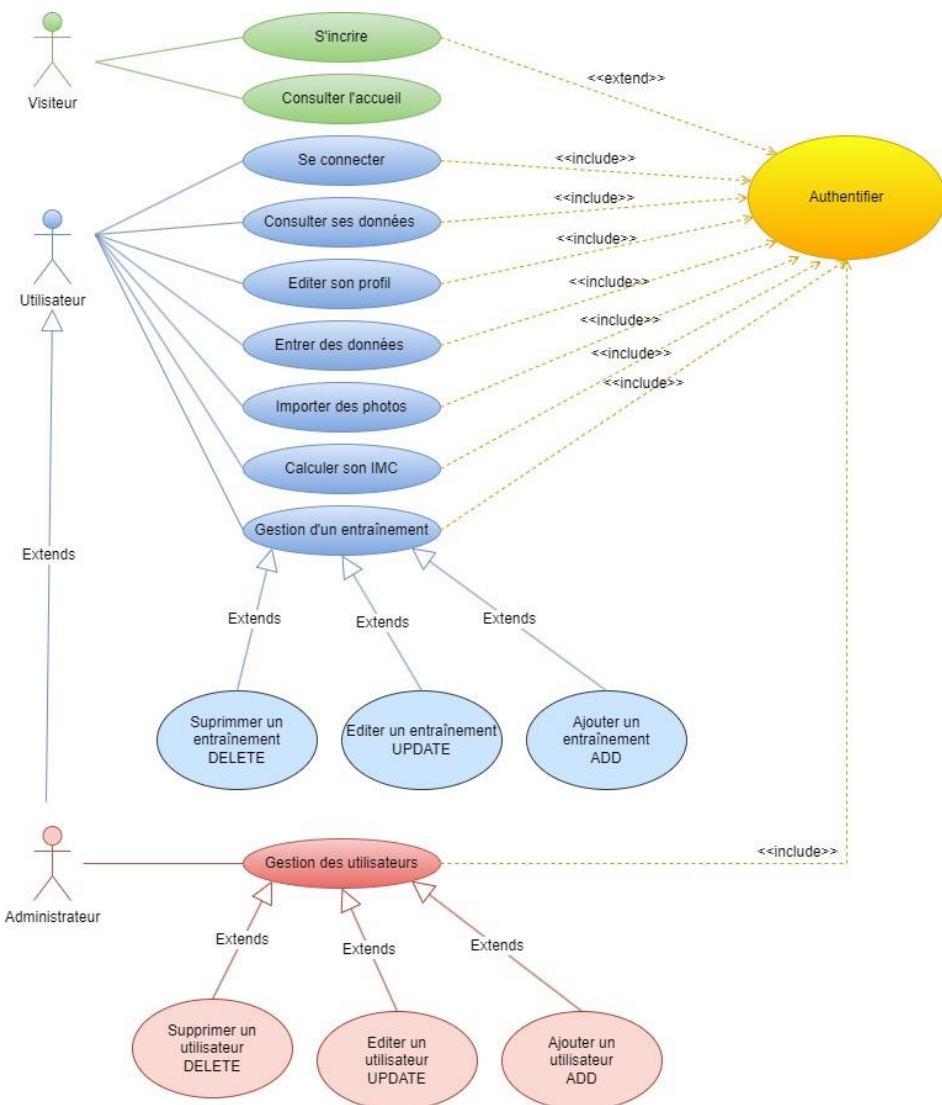
Partie n°2 ▶ Concevoir une application

### 1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

#### I. 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.

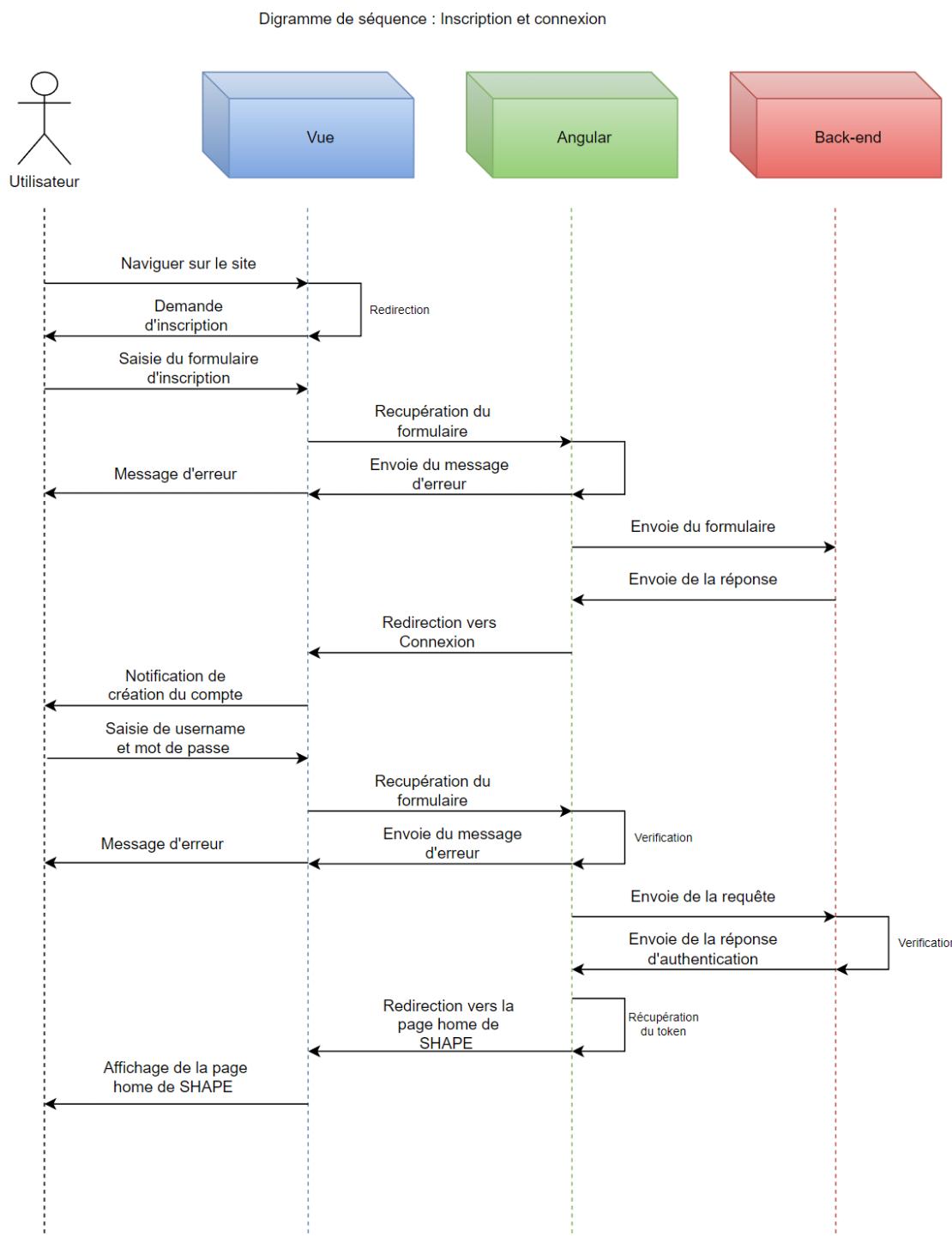
Voici le diagramme d'utilisation de l'application SHAPE sur le web :



# DOSSIER PROFESSIONNEL (DP)

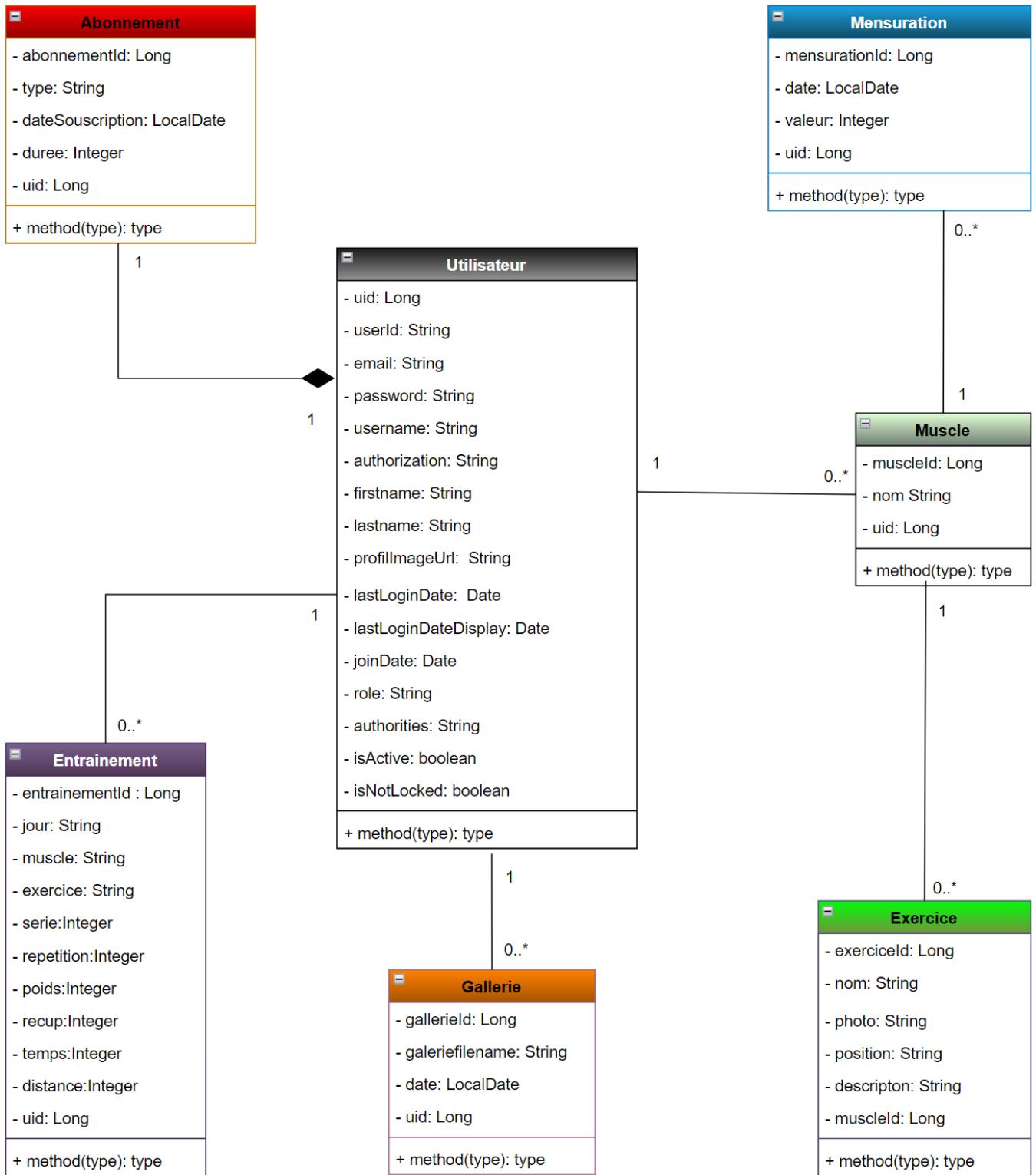
## II. Diagramme de séquence

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.



# DOSSIER PROFESSIONNEL (DP)

## III. Diagramme de classe



# DOSSIER PROFESSIONNEL (DP)

## 2. Précisez les moyens utilisés :



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

## 3. Avec qui avez-vous travaillé ?

J'ai travaillé en autonomie.

## 4. Contexte

Nom de l'entreprise, organisme ou association ► *Shape*

Chantier, atelier, service ► *Cliquez ici pour taper du texte.*

Période d'exercice ► Du : *01/01/2023* au : *03/03/2023*

## 5. Informations complémentaires (*facultatif*)

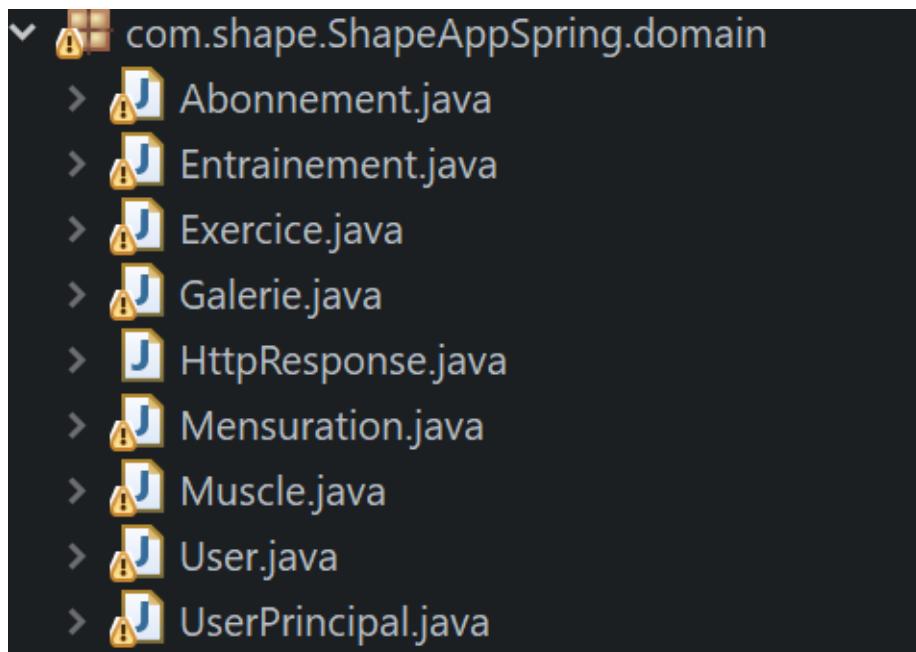
# DOSSIER PROFESSIONNEL (DP)

## Activité 3 Concevoir et développer une application multicouche répartie en intégrant les recommandations de sécurité

Partie n°3 ▶ Développer des composants métier

### 1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Création du package : domain



# DOSSIER PROFESSIONNEL (DP)

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 entraînementId;
    private String jour;
    private String muscle;
    private String exercice;
    private Integer série;
    private Integer répétition;
    private Integer poids;
    private Integer récup;
    private Integer temps;
    private Integer distance;
    private Long uid;

    // GETTER
    public Long getEntraînementId() {
        return entraînementId;
    }
    public String getJour() {
        return jour;
    }
    public String getMuscle() {
        return muscle;
    }
    public String getExercice() {
        return exercice;
    }
    public Integer getSérie() {
        return série;
    }
    public Integer getRépétition() {
        return répétition;
    }
    public Integer getPoids() {
        return poids;
    }
    public Integer getRécup() {
        return récup;
    }
    public Integer getTemps() {
        return temps;
    }
    public Integer getDistance() {
        return distance;
    }
    public Long getUid() {
        return uid;
    }

    // SETTER
    public void setEntraînementId(Long entraînementId) {
        this.entraînementId = entraînementId;
    }
    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 setSérie(Integer série) {
        this.série = série;
    }
    public void setRépétition(Integer répétition) {
        this.répétition = répétition;
    }
    public void setPoids(Integer poids) {
        this.poids = poids;
    }
    public void setRécup(Integer récup) {
        this.récup = récup;
    }
    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 entraînementId, String jour, String muscle,
String exercice, Integer série, Integer répétition, Integer poids,
Integer récup, Integer temps, Integer distance, Long uid) {
        super();
        this.entraînementId = entraînementId;
        this.jour = jour;
        this.muscle = muscle;
        this.exercice = exercice;
        this.série = série;
        this.répétition = répétition;
        this.poids = poids;
        this.récup = récup;
        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 appelés 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

# DOSSIER PROFESSIONNEL (DP)

## 2. Précisez les moyens utilisés :



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.

## 3. Avec qui avez-vous travaillé ?

J'ai travaillé en autonomie.

## 4. Contexte

Nom de l'entreprise, organisme ou association ► *Shape*

Chantier, atelier, service ► *Cliquez ici pour taper du texte.*

Période d'exercice ► Du : *01/01/2023* au : *03/03/2023*

## 5. Informations complémentaires (facultatif)

# DOSSIER PROFESSIONNEL (DP)

## Activité 3

Concevoir et développer une application multicouche répartie en intégrant les recommandations de sécurité

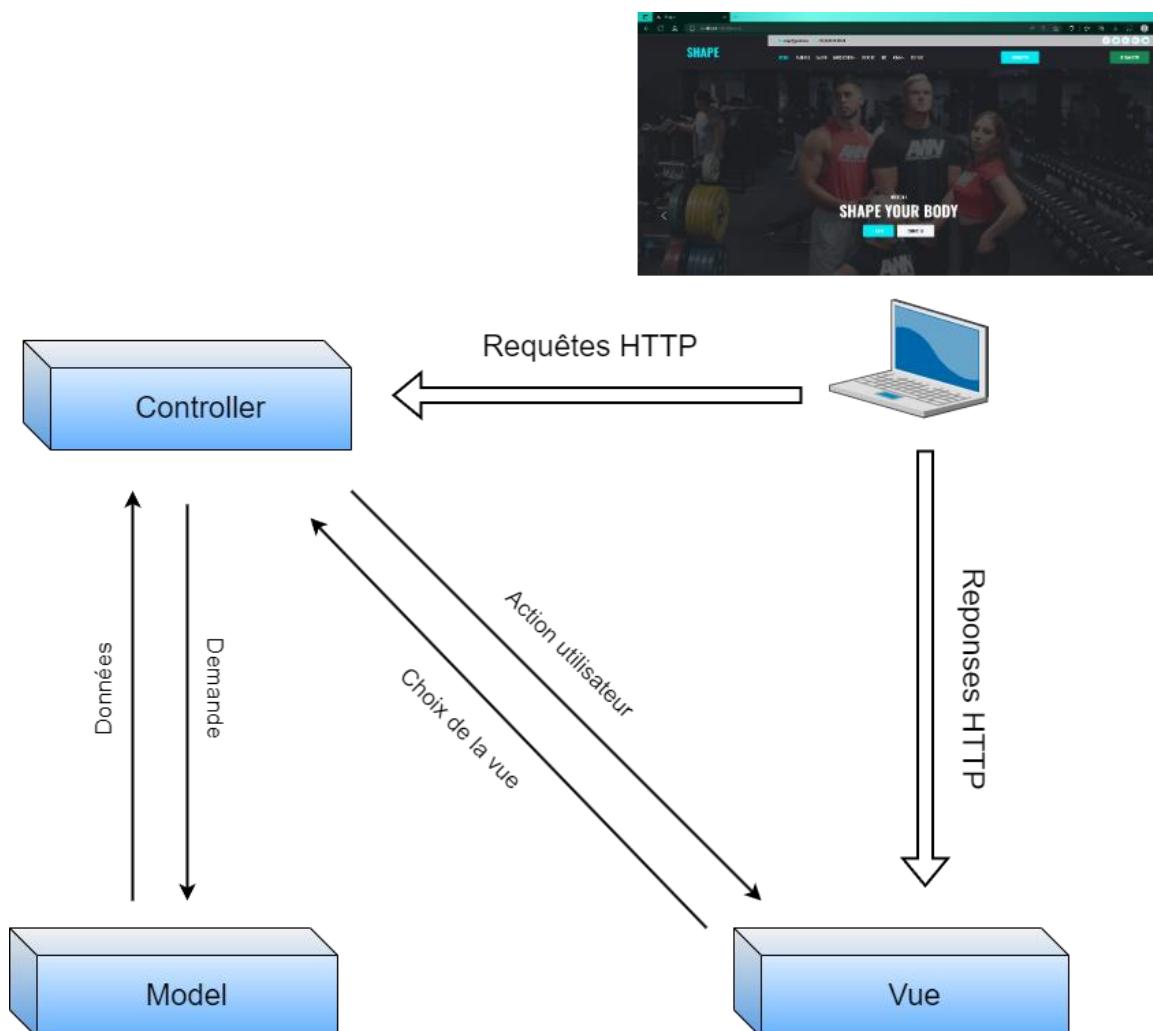
Partie n°4 ► Construire une application organisée en couches

### 1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

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.

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.



# DOSSIER PROFESSIONNEL (DP)

- 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.

## 2. Précisez les moyens utilisés :



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

## 3. Avec qui avez-vous travaillé ?

J'ai travaillé en autonomie.

## 4. Contexte

Nom de l'entreprise, organisme ou association ► *Shape*

Chantier, atelier, service ► *Cliquez ici pour taper du texte.*

Période d'exercice ► Du : *01/01/2023* au : *03/03/2023*

## 5. Informations complémentaires (facultatif)

# DOSSIER PROFESSIONNEL (DP)

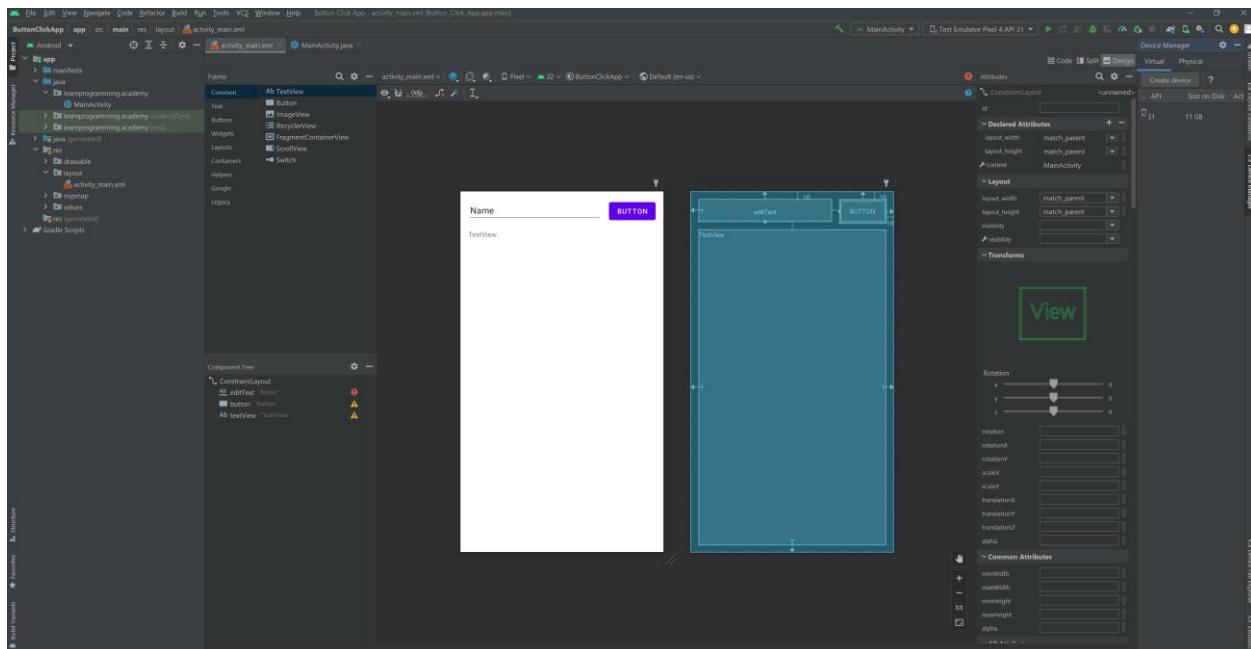
## Activité 3

Concevoir et développer une application multicouche répartie en intégrant les recommandations de sécurité

Partie n°5 ► Développer une application mobile

### 1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

#### I. Interface graphique Android Studio



# DOSSIER PROFESSIONNEL (DP)

## II. Maquette graphique sur téléphone

The mobile phone screen displays the homepage of the "SHAPE" application. At the top, there is a navigation bar with the word "SHAPE" in blue and a three-line menu icon. Below the header, a dark banner features the text "WELCOME" and "SHAPE YOUR BODY" in large white letters, along with "JOIN US" and "CONTACT US" buttons. The main content area has a dark background with a photograph of people working out in a gym. Below the photo, the text "POURQUOI UTILISER SHAPE ?" is followed by two sections: "Suis ton évolution" (with an icon of a person running) and "Calcul ton BMI" (with an icon of a scale). A small note under each section provides placeholder text.

ACCUEIL

PLANNING

GALERIE

MENSURATION ▾

EXERCICE

IMC

ADMIN ▾

CONTACT

WELCOME

SHAPE YOUR BODY

JOIN US CONTACT US

POURQUOI UTILISER SHAPE ?

Suis ton évolution

Calcul ton BMI

Quis ipsum suspendisse ultrices gravida. Risus commodo viverra maecenas accumsan lacus vel facilisis.

Quis ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore facilisis.

The mobile phone screen shows the "LISTE DES EXERCICES" (List of Exercises) section of the "SHAPE" app. The title "CHOISIS TON MUSCLE" is prominently displayed. Below the title, various muscle groups are listed: ÉPAULES, BICEPS (highlighted in a blue circle), TRICEPS, PECTORAUX, DOS, ABDOMINAUX, FESSIERS, QUADRICEPS, ISCHIO-JAMBIERS, and MOLLETS. At the bottom of the screen is a table showing a history of measurements:

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

# DOSSIER PROFESSIONNEL (DP)

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 = poids en kg/taille<sup>2</sup>. 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 = poids en kg/taille<sup>2</sup>. 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

### CONTACTER-NOUS

### GET IN TOUCH

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

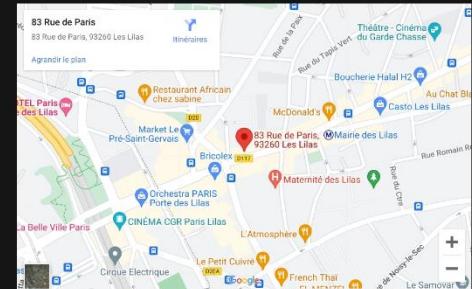
Name:

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.

# DOSSIER PROFESSIONNEL (DP)

## 2. Précisez les moyens 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.

## 3. Avec qui avez-vous travaillé ?

J'ai travaillé en autonomie.

## 4. Contexte

Nom de l'entreprise, organisme ou association ► *Shape*

Chantier, atelier, service ► *Cliquez ici pour taper du texte.*

Période d'exercice ► Du : *01/01/2023* au : *03/03/2023*

## 5. Informations complémentaires (facultatif)

# DOSSIER PROFESSIONNEL (DP)

## Activité 3

**Concevoir et développer une application multicouche répartie en intégrant les recommandations de sécurité**

**Partie n°6 ▶ Préparer et exécuter les plans de tests d'une application**

### 1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

#### I. Tester le 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.

#### II. Tester le back-end

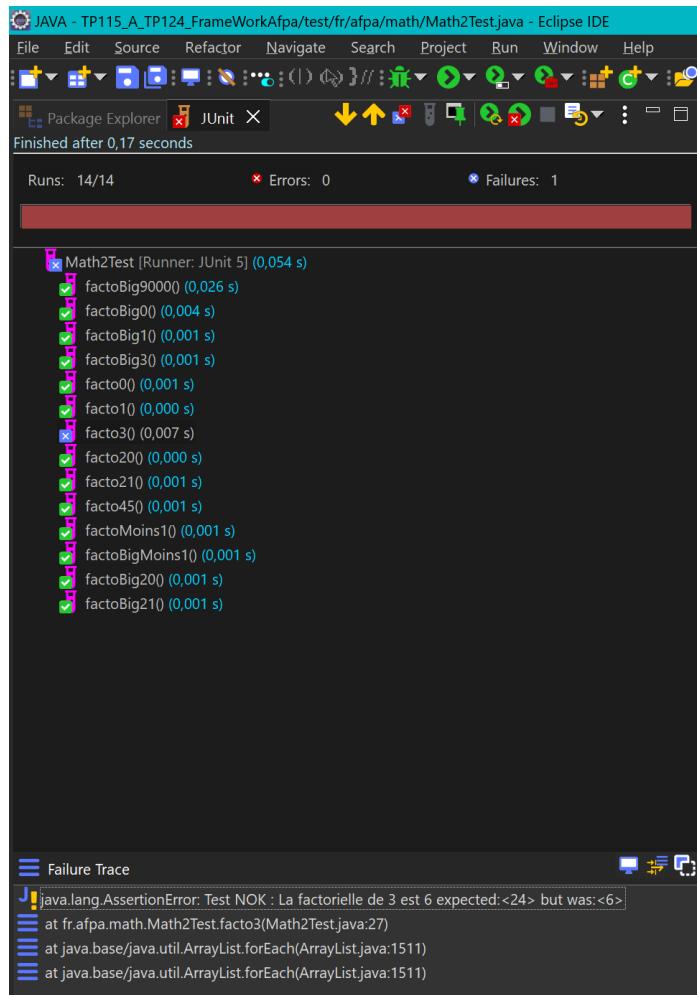
Test Unitaire de la méthode factorielle, voici les résultats en utilisant JUnit :

The screenshot shows the Eclipse IDE interface with the title bar "JAVA - TP115\_A\_TP124\_FrameWorkAfpa/test/fr/afpa/math/Math2Test.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help. The toolbar has various icons for file operations. The "Package Explorer" view shows a project named "Math2Test". The "JUnit" view shows the test results: "Runs: 14/14", "Errors: 0", "Failures: 0". Below this, a list of test methods is shown, all of which have passed (indicated by green checkmarks):

- Math2Test [Runner: JUnit 5] (0,077 s)
- factoBig9000() (0,050 s)
- factoBig0() (0,005 s)
- factoBig1() (0,001 s)
- factoBig3() (0,001 s)
- facto0() (0,001 s)
- facto1() (0,001 s)
- facto3() (0,001 s)
- facto20() (0,001 s)
- facto21() (0,001 s)
- facto45() (0,002 s)
- factoMoins1() (0,002 s)
- factoBigMoins1() (0,001 s)
- factoBig20() (0,001 s)
- factoBig21() (0,001 s)

# DOSSIER PROFESSIONNEL (DP)

En cas d'erreur :



### III. Tester la base de données

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, etc. 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** :

A screenshot of the Postman application. The top bar shows "POST" and the URL "localhost:8090/muscles/1". The "Send" button is on the right. Below the URL, there are tabs for "Params", "Authorization", "Headers (8)", "Body" (which is currently selected and highlighted in green), "Pre-request Script", "Tests", and "Settings". Under "Body", there are options for "none", "form-data", "x-www-form-urlencoded", "raw", "binary", and "GraphQL", with "JSON" selected. To the right of the body tab, there are buttons for "Cookies" and "Beautify". The main area shows the response: "Muscle successfully created!".

Méthode **GET** :

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

# DOSSIER PROFESSIONNEL (DP)

The screenshot shows a Postman interface with a 'New Request' tab selected. The URL is set to 'localhost:8090/muscles'. The 'Body' tab is active, showing a JSON response. The response body is:

```
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    "listExercice": [
17        {
18            "exerciceId": 2,
19            "nom": "Développé Arnold",
20            "photo": "developpe Arnold-exercice-musculation.gif"
21        }
22    ]
}
```

The status bar at the bottom indicates: Status: 200 OK Time: 39 ms Size: 3.48 KB Save Response.



Status: 200 OK

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.

# DOSSIER PROFESSIONNEL (DP)

## 2. Précisez les moyens utilisés :



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.



Postman est un logiciel gratuit qui me 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.

## 3. Avec qui avez-vous travaillé ?

J'ai travaillé en autonomie.

## 4. Contexte

Nom de l'entreprise, organisme ou association ► *Shape*

Chantier, atelier, service ► *Cliquez ici pour taper du texte.*

Période d'exercice ► Du : *01/01/2023* au : *03/03/2023*

## 5. Informations complémentaires (*facultatif*)

# DOSSIER PROFESSIONNEL (DP)

## Activité 3

**Concevoir et développer une application multicouche répartie en intégrant les recommandations de sécurité**

**Partie n°7 ▶ Préparer et exécuter le déploiement d'une application**

### 1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Déploiement de mon site internet.

- J'ai recherché un hébergeur gratuit pouvant accueillir mon site internet en statique.
- J'ai créé un nouveau repository : adhouen (ce sera le nom de domaine de mon site)
- Puis j'ai téléchargé les fichiers de mon site dans mon dépôt GitHub en étant sur d'inclure un fichier index.html à la racine du dépôt.

The screenshot shows a GitHub repository named 'AdHouen Ajout Tp Grid'. It has 1 branch and 0 tags. The commit history shows 9 commits from Sep 28, 2022. The files listed are: Annexe, Photo, pageweb, script, style, .gitignore, LICENSE, and index.html. All files were committed 7 months ago. A message at the bottom encourages adding a README.

File	Description	Time Ago
Annexe	test site internet	7 months ago
Photo	test site internet	7 months ago
pageweb	Ajout Tp Grid	7 months ago
script	test site internet	7 months ago
style	test site internet	7 months ago
.gitignore	creation de git ignore	7 months ago
LICENSE	Initial commit	7 months ago
index.html	Rectification carousel	7 months ago

Help people interested in this repository understand your project by adding a README. Add a README

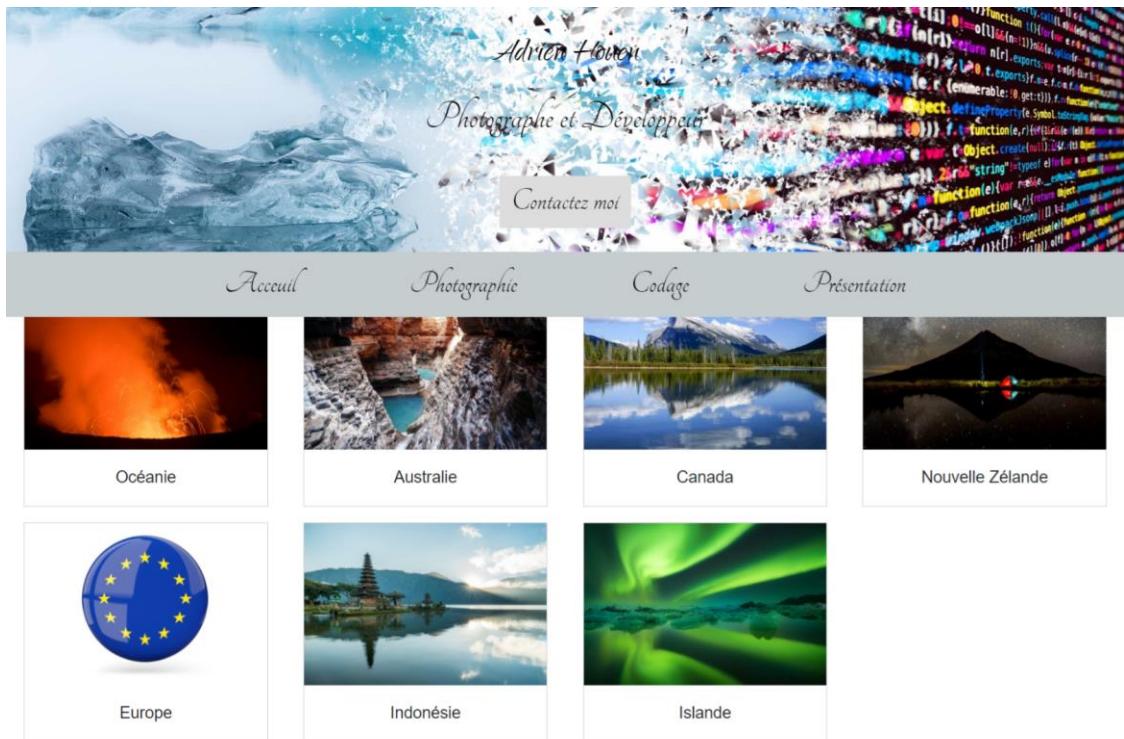
- Après quelques minutes, le site web statique est déployé et il sera accessible à l'adresse suivant :  
[https://votre\\_nom\\_utilisateur.github.io/nom\\_de\\_votre\\_depot](https://votre_nom_utilisateur.github.io/nom_de_votre_depot)
- Pour ma part : <https://adhouen.github.io/>

# DOSSIER PROFESSIONNEL (DP)

Partie Codage :



Partie photographie :



# DOSSIER PROFESSIONNEL (DP)

## 2. Précisez les moyens utilisés :



Google est un moteur de recherche en ligne qui permet aux utilisateurs de trouver des informations sur internet en utilisant des mots-clés.



Github : github est mon hébergeur

## 3. Avec qui avez-vous travaillé ?

J'ai travaillé en autonomie.

## 4. Contexte

Nom de l'entreprise, organisme ou association ► *Shape*

Chantier, atelier, service ► *Cliquez ici pour taper du texte.*

Période d'exercice ► Du : *01/10/2023* au : *05/10/2023*

## 5. Informations complémentaires (facultatif)

# **DOSSIER PROFESSIONNEL (DP)**

## **Titres, diplômes, CQP, attestations de formation**

(facultatif)

Intitulé	Autorité ou organisme	Date
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.

# DOSSIER PROFESSIONNEL (DP)

## Déclaration sur l'honneur

Je soussigné(e) [prénom et nom] **Adrien Houen**,

déclare sur l'honneur que les renseignements fournis dans ce dossier sont exacts et que je suis l'auteur(e) des réalisations jointes.

Fait à **Les Lilas** le **17/04/2023**

pour faire valoir ce que de droit.

Signature :



# DOSSIER PROFESSIONNEL (DP)

## Documents illustrant la pratique professionnelle

(*facultatif*)

### Intitulé

Cliquez ici pour taper du texte.

# DOSSIER PROFESSIONNEL (DP)

## ANNEXES

*(Si le RC le prévoit)*