

# TP Blog Recette

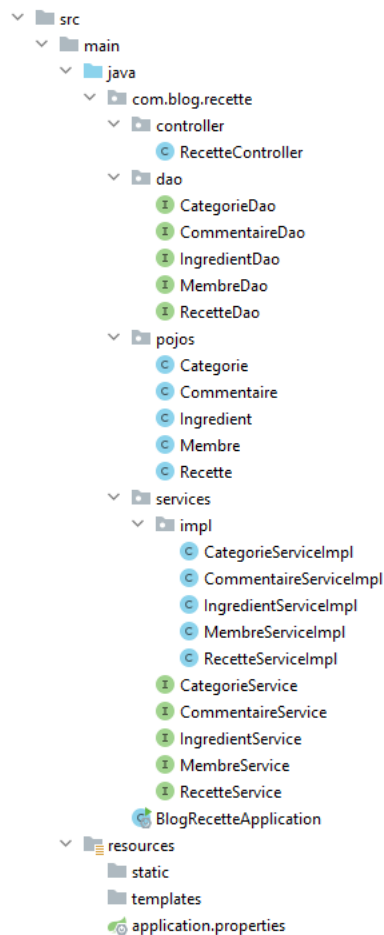
L'objectif du TP est de créer un site internet de recettes à la manière de Marmiton en utilisant Spring, Spring Boot, Spring Data et une base de données sous MySQL.

## Première Partie :

La mise en place du projet Spring Boot BlogRecette consiste à effectuer les étapes suivantes :

- Création du projet
- Création des models
- Création des DAO
- Création des Services
- Création d'un contrôleur pour initialiser les données dans la base de données.

## Structure du projet :



## Création du projet avec Initializr :



### Project

☐ Gradle Project ☒ Maven Project

### Language

☒ Java ☐ Kotlin ☐ Groovy

### Spring Boot

☐ 3.0.0 (SNAPSHOT) ☐ 3.0.0 (RC2) ☐ 2.7.6 (SNAPSHOT) ☒ 2.7.5  
☐ 2.6.14 (SNAPSHOT) ☐ 2.6.13

### Project Metadata

Group com.blog

Artifact recette

Name Recette

Description Project Recette for Spring Boot

Package name com.blog.recette

Packaging ☒ Jar ☐ War

Java ☐ 19 ☐ 17 ☒ 11 ☐ 8

### Dependencies

ADD DEPENDENCIES... CTRL + B

### Spring Data JPA

SQL

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

### Spring Web

WEB

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

### Validation

I/O


Bean Validation with Hibernate validator.

### MySQL Driver

SQL

MySQL JDBC and R2DBC driver.

## Modification du fichier application.properties :

BlogRecette > src > main > resources >  application.properties

Le fichier application.properties permet de configurer l'application :

```
spring.datasource.url=jdbc:mysql://localhost:3306/blog_recette_spring?useSSL=false&serverTimezone=UTC&createDatabaseIfNotExist=true
spring.datasource.username=root
spring.datasource.password=
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
spring.jpa.hibernate.ddl-auto=create
spring.jpa.show-sql=true
server.port=8090
```

la propriété `spring.jpa.hibernate.ddl-auto` permet de spécifier comment le schéma de la base de données va être initialisée via Hibernate <https://springhow.com/spring-boot-database-initialization/>.

## Création des models :

La base de données utilisée est simple. Elle se compose de cinq tables :

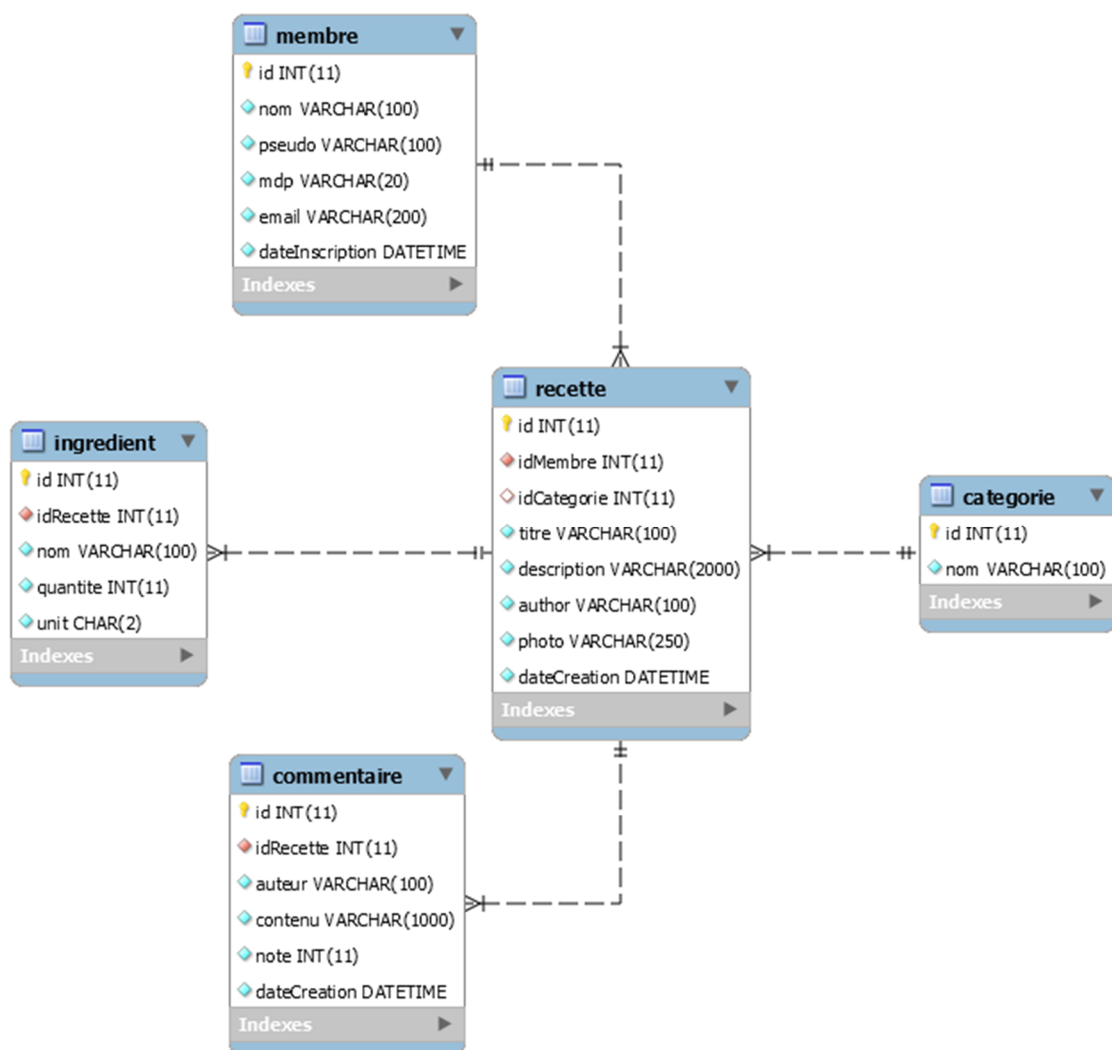
Table categorie : contient la liste des catégories de recettes

Table recette : contient les recettes du blog

Table ingredients : contient la liste des ingrédients pour une recette

Table commentaire : contient les commentaires associés aux recettes.

Table membre : contient les membres du blog (ceux qui créent les recettes).



Commencer par créer la base de données « blog\_recette\_spring » dans MySQL.

Créer ensuite les POJO :

Classe Membre :

```
@Entity
public class Membre {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String nom;
    private String pseudo;
    private String mdp;
    private String email;
    private Date dateInscription;

    @OneToMany(mappedBy="membre")
    private List<Recette> recettes;

    // Getter et Setter

    // toString
}
```

Classe Recette :

```
@Entity
public class Recette {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String titre;
    private String description;
    private String author;
    private String photo;
    private Date dateCreation;

    @ManyToOne
    private Membre membre;

    @ManyToOne
    private Categorie categorie;

    @OneToMany(mappedBy="recette")
    private List<Ingredient> ingredients;

    @OneToMany(mappedBy="recette")
    private List<Commentaire> commentaires;

    // Getter et Setter

    // toString
}
```

Classe Ingredient :

```
@Entity
public class Ingredient {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String nom;
    private int quantite;
    private String unit;

    @ManyToOne
    private Recette recette;

    // Getter et Setter

    // toString
}
```

Classe Categorie :

```
@Entity
public class Categorie {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String nom;

    @OneToMany(mappedBy = "categorie")
    private List<Recette> recettes;

    // Getter et Setter

    // toString
}
```

Classe Commentaire :

```
-
@Entity
public class Commentaire {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String auteur;
    private String contenu;
    private int note;
    private Date dateCreation;

    @ManyToOne
    private Recette recette;

    // Getter et Setter

    // toString
}
```

## Création des DAO :

Membre Dao :

```
@Repository
public interface MembreDao extends JpaRepository<Membre, Long> {
    // Methode par dérivation
    Membre findByPseudo(String pseudo);
}
```

CategorieDao :

```
@Repository
public interface CategorieDao extends JpaRepository<Categorie, Long> {
}
```

RecetteDao :

```
@Repository
public interface RecetteDao extends JpaRepository<Recette, Long> {
    // Par dérivation
    1 usage
    List<Recette> findByMembre(Membre membre);
    List<Recette> findByCategorie(Categorie categorie);
};
```

IngredientDao :

```
@Repository
public interface IngredientDao extends JpaRepository<Ingredient, Long> {
    List<Ingredient> findByRecette(Recette recette);
}
```

CommentaireDao :

```
@Repository
public interface CommentaireDao extends JpaRepository<Commentaire, Long> {
    List<Commentaire> findByRecette(Recette recette);
}
```



## Création des interfaces pour les Services :

MembreService :

```
public interface MembreService {  
  
    1 implementation  
    Membre ajouterMembre(Membre membre);  
  
    1 implementation  
    Membre recupererMembre(Long idMembre);  
  
    1 implementation  
    List<Membre> recupererMembres();  
  
    1 implementation  
    List<Recette> recupererRecettes(Membre membre);  
}
```

CategorieService :

```
public interface CategorieService {  
  
    1 implementation  
    Categorie ajouterCategorie(Categorie categorie);  
  
    1 implementation  
    Categorie recupererCategorie(Long idCategorie);  
  
    1 implementation  
    List<Categorie> recupererCategories();  
  
    1 implementation  
    List<Recette> recupererRecettes(Categorie categorie);  
}
```

RecetteService :

```
public interface RecetteService {  
    Recette ajouterRecette(Recette recette);  
  
    Recette recupererRecette(Long idRecette);  
  
    List<Recette> recupererRecettes();  
  
    List<Ingredient> recupererIngredients(Recette recette);  
  
    List<Commentaire> recupererCommentaires(Recette recette);  
}
```

IngredientService :

```
public interface IngredientService {  
    Ingredient ajouterIngredient(Ingredient ingredient);  
  
    Ingredient recupererIngredient(Long idIngredient);  
  
    List<Ingredient> recupererIngredients();  
}
```

CommentaireService :

```
public interface CommentaireService {  
    Commentaire ajouterCommentaire(Commentaire commentaire);  
  
    Commentaire recupererCommentaire(Long idCommentaire);  
  
    List<Commentaire> recupererCommentaires();  
}
```

## Implémentation des Services :

MembreServiceImpl :

```
@Service
public class MembreServiceImpl implements MembreService {

    3 usages
    @Autowired
    private MembreDao membreDao;

    1 usage
    @Autowired
    private RecetteDao recetteDao;

    @Override
    public Membre ajouterMembre(Membre membre) {
        return membreDao.save(membre);
    }

    @Override
    public Membre recupererMembre(Long idMembre) {
        return membreDao.findById(idMembre).orElse( other: null);
    }

    @Override
    public List<Membre> recupererMembres() {
        return membreDao.findAll();
    }

    @Override
    public List<Recette> recupererRecettes(Membre membre) {
        return recetteDao.findByMembre(membre);
    }
}
```

---

RecetteServiceImpl :

```
@Service
public class RecetteServiceImpl implements RecetteService {

    3 usages
    @Autowired
    private RecetteDao recetteDao;

    1 usage
    @Autowired
    private IngredientDao ingredientDao;

    1 usage
    @Autowired
    private CommentaireDao commentaireDao;

    @Override
    public Recette ajouterRecette(Recette recette) {
        return recetteDao.save(recette);
    }

    @Override
    public Recette recupererRecette(Long idRecette) {
        return recetteDao.findById(idRecette).orElse( other: null);
    }

    @Override
    public List<Recette> recupererRecettes() {
        return recetteDao.findAll();
    }

    @Override
    public List<Ingredient> recupererIngredients(Recette recette) {
        return ingredientDao.findByRecette(recette);
    }

    @Override
    public List<Commentaire> recupererCommentaires(Recette recette) {
        return commentaireDao.findByRecette(recette);
    }
}
```

IngredientServiceImpl :

```
@Service
public class IngredientServiceImpl implements IngredientService {

    3 usages
    @Autowired
    private IngredientDao ingredientDao;

    @Override
    public Ingredient ajouterIngredient(Ingredient ingredient) {
        return ingredientDao.save(ingredient);
    }

    @Override
    public Ingredient recupererIngredient(Long idIngredient) {
        return ingredientDao.findById(idIngredient).orElse( other: null);
    }

    @Override
    public List<Ingredient> recupererIngredients() {
        return ingredientDao.findAll();
    }
}
```

CategorieServiceImpl :

```
@Service
public class CategorieServiceImpl implements CategorieService {

    3 usages
    @Autowired
    private CategorieDao categorieDao;

    1 usage
    @Autowired
    private RecetteDao recetteDao;

    @Override
    public Categorie ajouterCategorie(Categorie categorie) {
        return categorieDao.save(categorie);
    }

    @Override
    public Categorie recupererCategorie(Long idCategorie) {
        return categorieDao.findById(idCategorie).orElse( other: null);
    }

    @Override
    public List<Categorie> recupererCategories() {
        return categorieDao.findAll();
    }

    @Override
    public List<Recette> recupererRecettes(Categorie categorie) {
        return recetteDao.findByCategorie(categorie);
    }
}
```

CommentaireServiceImpl :

```
@Service
public class CommentaireServiceImpl implements CommentaireService {

    3 usages
    @Autowired
    private CommentaireDao commentaireDao;

    @Override
    public Commentaire ajouterCommentaire(Commentaire commentaire) {
        return commentaireDao.save(commentaire);
    }

    @Override
    public Commentaire recupererCommentaire(Long idCommentaire) {
        return commentaireDao.findById(idCommentaire).orElse( other: null);
    }

    @Override
    public List<Commentaire> recupererCommentaires() {
        return commentaireDao.findAll();
    }
}
```

## Création du Controller pour mettre des données :

InitDataController :

```
@Controller
public class InitDataController {

    // Le contrôleur a besoin de services
    // autrement dit il délègue des traitements à un ou plusieurs services
    1 usage
    private MembreService membreService;
    1 usage
    private CategorieService categorieService;
    1 usage
    private RecetteService recetteService;
    1 usage
    private IngredientService ingredientService;
    1 usage
    private CommentaireService commentaireService;

    // Ce constructeur va provoquer l'injection de dépendances
    public InitDataController(MembreService membreService, CategorieService categorieService, RecetteService recetteService,
                             IngredientService ingredientService, CommentaireService commentaireService) {

        super();
        this.membreService = membreService;
        this.categorieService = categorieService;
        this.recetteService = recetteService;
        this.ingredientService = ingredientService;
        this.commentaireService = commentaireService;
    }

    // Cette méthode sera invoquée dès que Spring a injecté tous les objets
    @PostConstruct
    private void init() {

        // TODO
        // crée 2 membres
        // crée 2 catégories
        // crée 2 recettes
        // crée les ingrédients des 2 recettes
        // crée 2 commentaires
    }
}
```



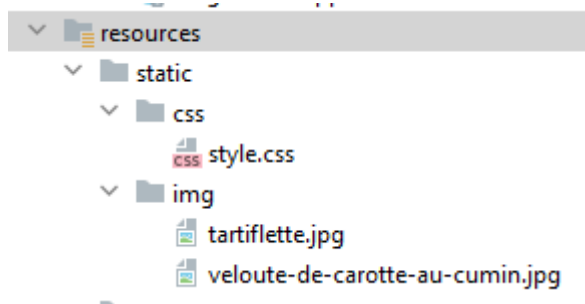
## Deuxième Partie :

La mise en place des pages du BlogRecette consiste à effectuer les étapes suivantes :

- Création des controleurs
- Création des vues

### Design du site internet

Créer un dossier img et css dans le dossier static de ressources



Copier les images des recettes dans le dossier img

Créer un fichier style.css et copier le style suivant :

```
html, body {  
    height: 100%;  
}  
  
body {  
    color: #bfbfbf;  
    background: black;  
    font-family: 'Futura-Medium', 'Futura', 'Trebuchet MS', sans-serif;  
}  
  
h1 {  
    color: white;  
}  
  
.imgRecette {  
    margin-top: 10px;  
    margin-bottom: 10px;  
}  
  
.titreRecette {  
    margin-bottom: 0px;  
}  
  
.titreIngredient {
```

```
    margin-top: 10px;
    margin-bottom: 10px;
}

.titreCommentaire {
    margin-top: 10px;
    margin-bottom: 10px;
}

.checked {
    color: orange;
}

.primaryBtn {
    justify-content: center;
    text-transform: uppercase;
    font-weight: bold;
    padding: 0 1rem;
    color: white;
    text-decoration: none;
    background-color: #f47321;
    border: 2px solid #f47321;
    line-height: 2rem;
    border-radius: 2rem;
    font-size: 1rem;
    text-overflow: ellipsis;
    cursor: pointer;
    transition: all 0.2s ease-in;
}

.submitBtn {
    color: #fff !important;
    background-color: #f47321;
    cursor: pointer;
    margin-top: 10px;
    border: none;
    border-radius: 10px;
    height: 32px;
    font-size: 14px;
    outline: medium none;
    padding: 0 10px !important;
    margin-bottom: 10px !important;
    box-sizing: border-box;
    -webkit-appearance: none;
}

.inputChamp {
    background-color: #f6f6f6;
    border: 1px solid #E4E3E3;
    border-radius: 10px;
    height: 32px;
    width: 100%;
    color: #69676a;
```

```

    font-size: 14px;
    outline: medium none;
    padding: 0 10px !important;
    margin-bottom: 10px !important;
    box-sizing: border-box;
    -webkit-appearance: none;
    width:30%;
}

.inputTextArea {
    font-size: 1rem;
    border-radius: 4px;
    padding: 12px;
    border: 1px solid #ddd;
    box-sizing: border-box;
}

.select {
    font-size: 1rem;
    padding: 2px 10px;
    border-radius: 4px;
    border: 1px solid #ddd;
    box-sizing: border-box;
}

#header {
    display: block;
}

#loginBar {
    display: block;
    padding: 5px 20px;
    width: 70%;
    margin: auto;
}

#loginBar .login {
    display: block;
    float: right;
    margin-bottom: 10px;
}

#global {
    display: block;
    clear: both;
    min-height: 100%; /* Voir commentaire sur html et body plus haut */
    background: #333534;
    width: 70%;
    margin: auto; /* Permet de centrer la div */
    text-align: justify;
    padding: 5px 20px;
}

```

```
#contenu {
    margin-bottom : 30px;
}

#titreBlog, #piedBlog {
    text-align: center;
}

#categorie {
    margin: 1rem 0;
    text-align: center;
    box-shadow: none;
    border-top: 1px solid #ddd;
}

#categorie ul li {
    display: inline-block;
    margin-bottom: 0.5rem;
}

#categorie ul li a {
    display: inline-block;
    color: #bfbfbf;
    line-height: 48px;
    cursor: pointer;
    text-decoration: none;
    margin: 0 12px;
}

#categorie li.selected a {
    border-bottom: 4px solid #f47321;
}

#txtCommentaire {
    width: 50%;
}
```

## Les pages à réaliser

Page d'inscription :

url : [inscription](#)



Cette page doit afficher un formulaire permettant d'effectuer une inscription.

Si le nom, le pseudo, l'email ou le mot de passe ne sont pas renseigné vous devez afficher un message d'erreur à la fin de la page.

Code HTML de la page :

```
<!doctype html>
<html lang="fr">
  <head>
    <meta charset="UTF-8" />
    <link rel="stylesheet" href="css/style.css" />
    <title>Mon Blog de Recettes</title>
  </head>
  <body>

    <header id="header">
      <a href="index"><h1 id="titreBlog">Mon Blog de Recettes</h1></a>
      <div style="width:300px;margin:20px auto;">Bienvenue sur mon blog de
recettes</div>
      <div id="loginBar">
        <div class="login">
```

```

        <a class="primaryBtn login" href="inscription">Inscription</a>
    </div>
</div>
</header>

<div id="global">
    <h1>Inscription</h1>
    <div id="inscription">
        <form method="post" action="inscription" >
            <input id="nom" name="nom" type="text" class="inputChamp"
placeholder="Votre nom *" /><br />
            <input id="pseudo" name="pseudo" type="text"
class="inputChamp" placeholder="Votre pseudo *" /><br />
            <input id="email" name="email" type="text"
class="inputChamp" placeholder="Votre email *" /><br />
            <input id="mdp" name="mdp" type="password"
class="inputChamp" placeholder="Votre mot de passe *" /><br />
            <br />
            <input type="submit" value="Je 'm'inscris"
class="submitBtn" />
        </form>
    </div>
    <div id="erreur">
        <p> Erreurs </p>
    </div>
</div>

<footer id="piedBlog">
    Blog réalisé par
</footer>
</body>
</html>

```

Page d'accueil :

url : /



Cette page doit afficher la liste des recettes. Pour cela vous allez devoir appeler la BDD pour récupérer les recettes, puis rendre dynamique les parties du HTML :

```
<!doctype html>
<html lang="fr">
  <head>
    <meta charset="UTF-8" />
    <link rel="stylesheet" href="css/style.css" />
    <title>Mon Blog de Recettes</title>
  </head>
  <body>

    <header id="header">
      <a href="index"><h1 id="titreBlog">Mon Blog de Recettes</h1></a>
      <div style="width:300px;margin:20px auto;">Bienvenue sur mon blog de
recettes</div>
      <div id="loginBar">
        <div class="login">
          <a class="primaryBtn login" href="inscription">Inscription</a>
        </div>
      </div>

    </body>
```

```

</header>

<div id="global">
  <div id="categorie">
    <ul>
      <li><a href="categorie?idCategorie=">Entrée</a></li>
      <li><a href="categorie?idCategorie=">Plat principal</a></li>
    </ul>
  </div>
  <article>
    <header>
      
      <a href="recette?id=">
        <h1 class="titreRecette">
          Tartiflette
        </h1>
      </a>
      <time>
        07/01/2019
      </time>
    </header>
    <p>
      La tartiflette savoyarde est un gratin de pommes de terre avec du
      Reblochon fondu dessus
    </p>
  </article>
  <hr />
</div>

<footer id="piedBlog">
  Blog réalisé par
</footer>

</body>
</html>

```



Page Catégorie :

url : [categorie](#)



Cette page doit afficher la liste des recettes suivant la catégorie sélectionné. Pour cela vous aller devoir appeler la BDD pour la catégorie dont l'id est indiqué dans l'url, puis récupérer les recettes de correspondant à cette catégorie pour rendre dynamique les parties du HTML.

Code HTML de la page :

```
<!doctype html>
<html lang="fr">
  <head>
    <meta charset="UTF-8" />
    <link rel="stylesheet" href="css/style.css" />
    <title>Mon Blog de Recettes</title>
  </head>
  <body>

    <header id="header">
      <a href="index"><h1 id="titreBlog">Mon Blog de Recettes</h1></a>
      <div style="width:300px;margin:20px auto;">Bienvenue sur mon blog de
recettes</div>
```

```

    <div id="loginBar">
        <div class="login">
            <a class="primaryBtn login" href="inscription">Inscription</a>
        </div>
    </div>
</header>

    <div id="global">
        <div id="categorie">
            <ul>
                <li class="selected"><a
href="categorie?idCategorie=">Entrée</a></li>
                <li><a href="categorie?idCategorie=">Plat principal</a></li>
            </ul>
        </div>
        <article>
            <header>
                
                <a href="recette?id=">
                    <h1 class="titreRecette">
                        Velouté de carottes au cumin
                    </h1>
                </a>
                <time>
                    08/01/2019
                </time>
            </header>
            <p>
                Un velouté de carotte au cumin
            </p>
        </article>
        <hr />
    </div>

    <footer id="piedBlog">
        Blog réalisé par
    </footer>

</body>
</html>

```


Page Recette :

url : [recette](#)

## Mon Blog de Recettes

Bienvenue sur mon blog de recettes

[INSCRIPTION](#)



### Tartiflette

★★★★★  
07/01/2019

La tartiflette savoyarde est un gratin de pommes de terre avec du Reblochon fondu dessus

### Ingrédients

- 750 g Pommes de terre
- 1 u Reblochon
- 200 g Lardons
- 3 cs Crème fraîche épaisse
- 2 u Oignons
- 20 g Beurre
- 1 cc Sel
- 1 p Poivre

### Commentaires

Nicolas : Super recette

Note : 5/5

09/01/2019

Votre commentaire \*

Note

1

Commenter

Cette page doit afficher les informations d'une recette et la liste des ingrédients. Pour cela vous aller devoir appeler la BDD pour la recette dont l'id est indiqué dans l'url, puis récupérer les ingrédients de la recette, pour rendre dynamique les parties du HTML.

Elle affiche aussi la liste des commentaires et doit permettre de saisir un nouveau commentaire avec le formulaire.

Si l'auteur ou le commentaire n'est pas renseigné vous devez afficher un message d'erreur à la fin de la page.

Code HTML de la page :

```
<!doctype html>
<html lang="fr">
<head>
  <meta charset="UTF-8" />
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css">
  <link rel="stylesheet" href="css/style.css" />
  <title>Mon Blog de Recette - Tartiflette</title>
</head>
<body>

  <header id="header">
    <a href="index"><h1 id="titreBlog">Mon Blog de Recettes</h1></a>
    <div style="width:300px;margin:20px auto;">Bienvenue sur mon blog de
recettes</div>
    <div id="loginBar">
      <div class="login">
        <a class="primaryBtn login" href="inscription">Inscription</a>
      </div>
    </div>
  </header>

  <div id="global">
    <article>
      <header>
        
        <h1 class="titreRecette">
          Tartiflette
        </h1>
        <span class="fa fa-star checked"></span>
        <span class="fa fa-star checked"></span>
        <span class="fa fa-star checked"></span>
        <span class="fa fa-star checked"></span>
        <span class="fa fa-star"></span>
        <br>
        <time>
          07/01/2019
        </time>
      </header>
      <p>
        La tartiflette savoyarde est un gratin de pommes de terre avec du
Reblochon fondu dessus
      </p>
    </article>
    <hr />
    <header>
      <h2 id="titreIngredient">
        Ingrédients
      </h2>
      <ul>
        <li>750 g Pommes de terre</li>
        <li>1 u Reblochon</li>
        <li>200 g Lardons</li>
      </ul>
    </header>
  </div>
</body>
</html>
```

```

        <li>3 cs Crème fraîche épaisse</li>
        <li>2 u Oignons</li>
        <li>20 g Beurre</li>
        <li>1 cc Sel</li>
        <li>1 p Poivre</li>
    </ul>
</header>

<h2 id="titreCommentaire">
    Commentaires
</h2>

<div class="divCommentaire">
    <p>Nicolas : Super recette </p>
    <p>Note : 5/5 </p>
    <p>
        09/01/2019
    </p>
    <hr>
</div>

<form method="post" action="recette?id=" >
    <input id="auteur" name="auteur" type="text" placeholder="Votre
nom *" class="inputChamp" /><br />
    <textarea id="txtCommentaire" name="contenu" rows="4"
placeholder="Votre commentaire *" class="inputTextArea" ></textarea><br/>
    <label for="note">Note</label><br />
    <select name="note" id="note" class="select">
        <option value="1">1</option>
        <option value="2">2</option>
        <option value="3">3</option>
        <option value="4">4</option>
        <option value="5">5</option>
    </select>
    <br />
    <input type="submit" value="Commenter" class="submitBtn" />
</form>
<div id="erreur">
    <p> Erreurs </p>
</div>

</div>

<footer id="piedBlog">
    Blog réalisé par
</footer>

</body>
</html>

```

### Fonctionnalités supplémentaires :

- Ajouter un bouton Déconnexion dans la barre de login. Ce bouton doit apparaître que si le membre est connecté.
- Ajouter un bouton Connexion dans la barre de login. Ce bouton doit apparaître que si le membre n'est pas connecté.
- Créer la page de connexion : connexion.html
- Créer les méthodes de contrôleur permettant de mettre en place les fonctionnalités de connexion.