

Les n° de pages  
en lien dans le projet  
s'afficheront ici.

# Dossier de Projets

# Rapport de Stage

## Développeur Web & Web Mobile

Stage réalisé

avec

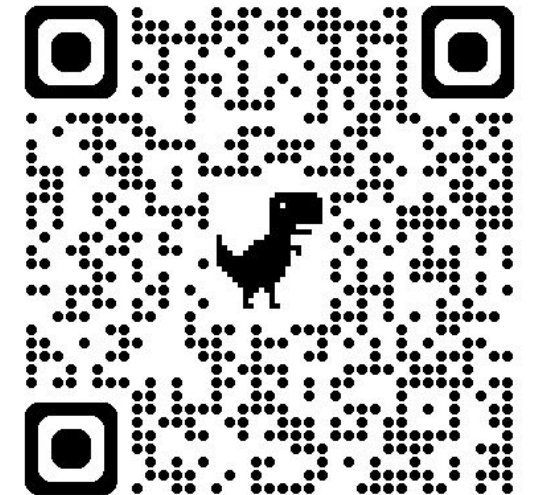
**Gemeny Software**

du 17 février au 02 mai 2025

Durée : 10 semaines

par

Yoann Le Goff





# Présentation de l'entreprise

## Gemeny Software :

### Spécialiste en Cybersécurité

Entreprise française innovante fondée en 2023

Solution logiciel d'obfuscation de données

Équipe:

- **Yohann Serpault**, fondateur et créateur du produit logiciel Gemeny
- **Tim Molivier**, stagiaire marketing digitale
- **Yoann Le Goff**, stagiaire développeur web & web mobile

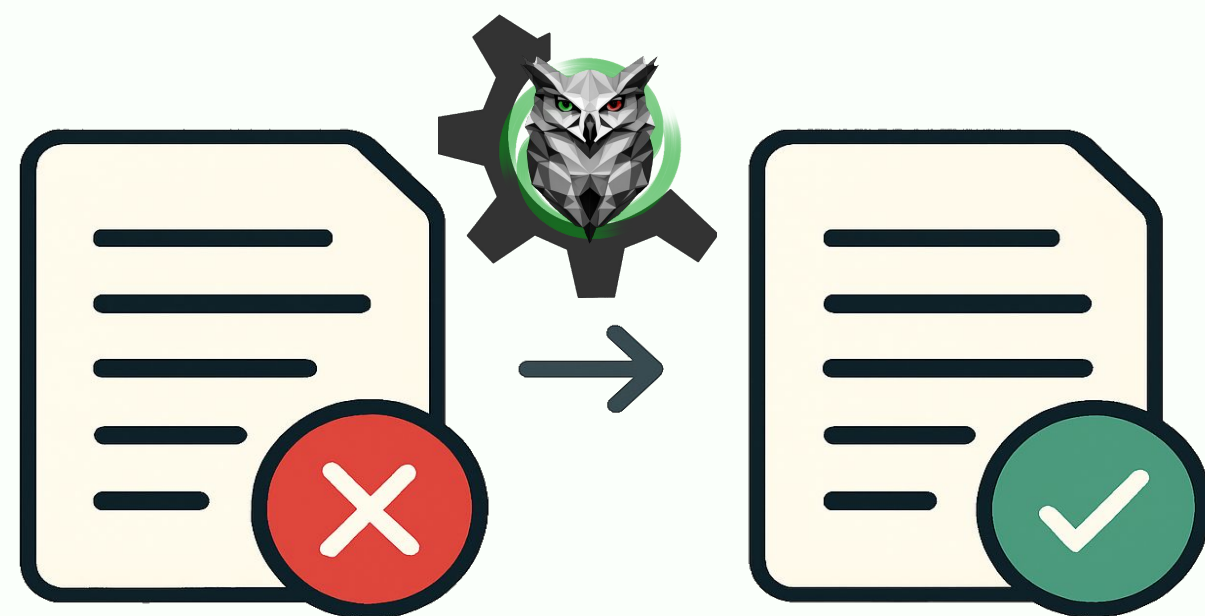




# Présentation du produit Gemeny

[page 6]

[annexe 1]



Données techniques  
et **sensibles**

Données techniques  
**obfusquées**



## Données entrantes

Jeux de données techniques, possiblement sensibles



## Sélection des valeurs

Modifier sans changer la structure des données



## Règles d'obfuscation

Paramètres des règles Gemeny



## Application logique Gemeny

Selon les règles établies, sans altérer la structure



## Résultat de l'opération

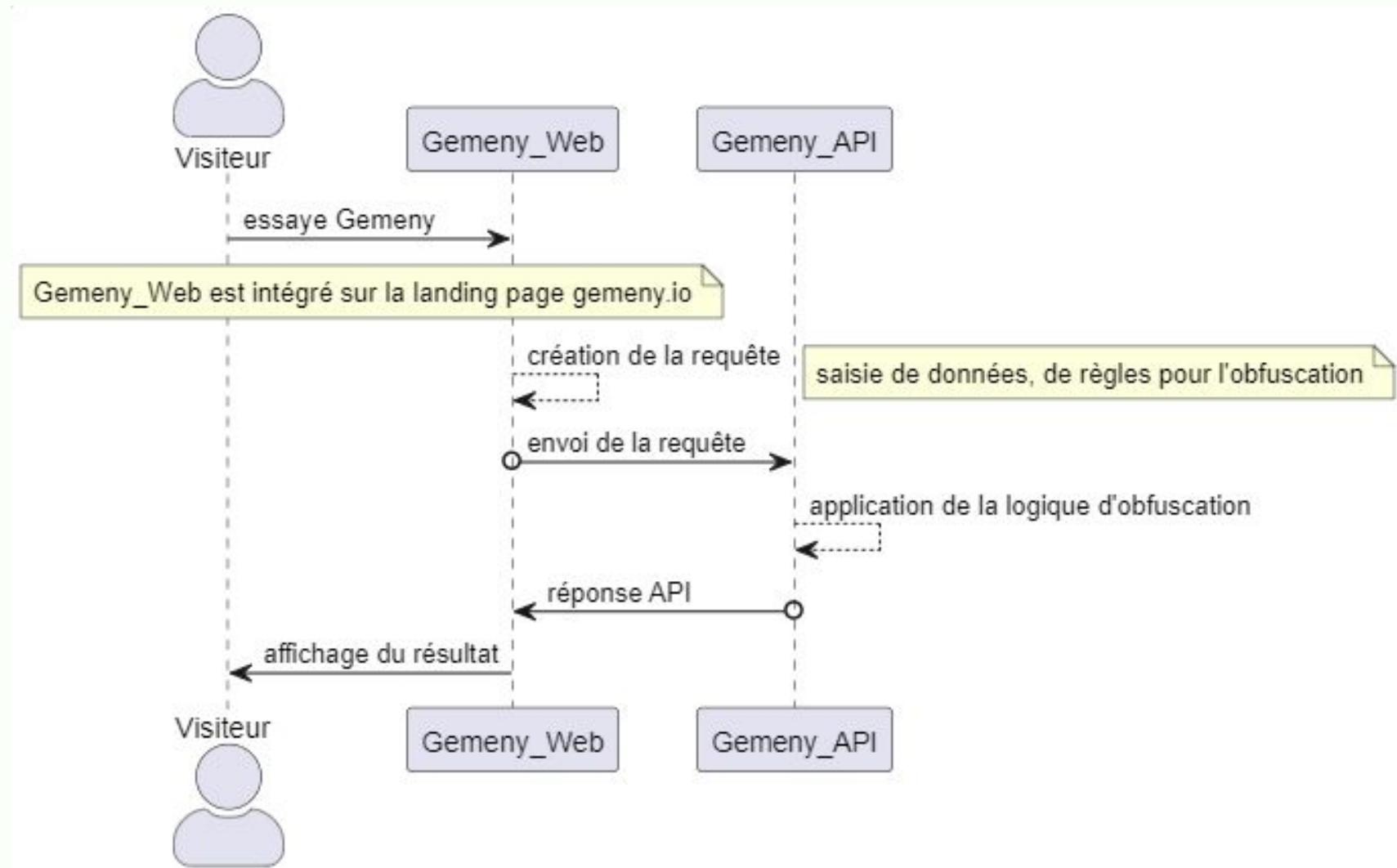
Récupération des données obfusquées

# Objectif du projet

[page 7]

Développement commercial par la démonstration du produit au public / potentiels clients.

Permettre à tout utilisateur de tester **Gemeny** en ligne, simplement et rapidement, sans installation.



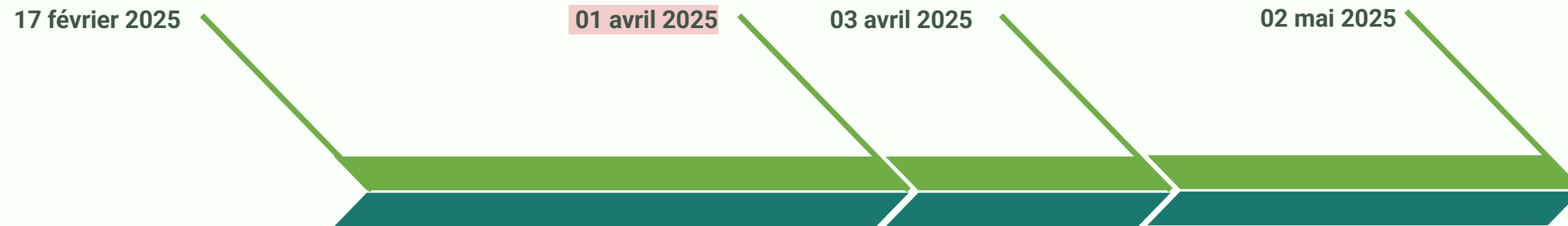
*Séquence de démo du produit Gemeny par un visiteur du site de l'entreprise*



# Chronologie du projet

[page 9-10]

[annexe 23]



## Début du stage

- Découverte du produit Gemeny
- Intégration à l'entreprise
- Conception puis développement de l'interface Gemeny\_web.  
( **ma mission principale** )
- Conception et développement de l'API métier Gemeny en parallèle  
(par **Yohann Serpault**)

## Salon SOFINS

- Présentation du produit Gemeny au salon **SOFINS**, date de livraison de **Gemeny\_Web v1**

**Enjeu principal du projet**

**Date de livraison impérative**

## Fin de la période de stage

- Ajout de fonctionnalités, résolution de bugs sur Gemeny\_web.
- Conception et développement de l'API Gemeny\_Auth



# Maquettage de l'application

[pages 11-12]

Try Gemeny here with 3 easy steps

1. insert your data

```
[ {
  "latitude": 48.856614,
  "longitude": 2.552222,
  "timestamp": "2024-02-15T11:16:11.813424",
  "IMEI": "3211762721091010",
  "MAC_Wifi": "52:83:f1:17:e1:dc",
  "SSID_Wifi": "Clayton LLC",
  "MAC_Bluetooth": "fc:b8:88:b5:71:4d",
  "SSID_Bluetooth": "Robert Austin"
}]
```

2. set some rules for obfuscation

Data Source ?	Custom Regex ?	Randomize options	Match ?	Delete
<input checked="" type="checkbox"/> <div>Line Key</div> écriture / liste	écriture	<input type="radio"/> Auto random characters <input checked="" type="radio"/> Realistic data set	liste	<input checked="" type="checkbox"/>
<input type="checkbox"/> <div>Line Key</div> écriture / liste	écriture	<input checked="" type="radio"/> Auto random characters <input type="radio"/> Realistic data set	liste	<input checked="" type="checkbox"/>

3. all set ? Click here to run Gemeny

+ Add new rule

- respect charte graphique Gemeny
- cohérence avec l'application existante
- amélioration expérience utilisateur (UX/UI)
- approche mobile-first

Maquette desktop réalisée sur **Figma**





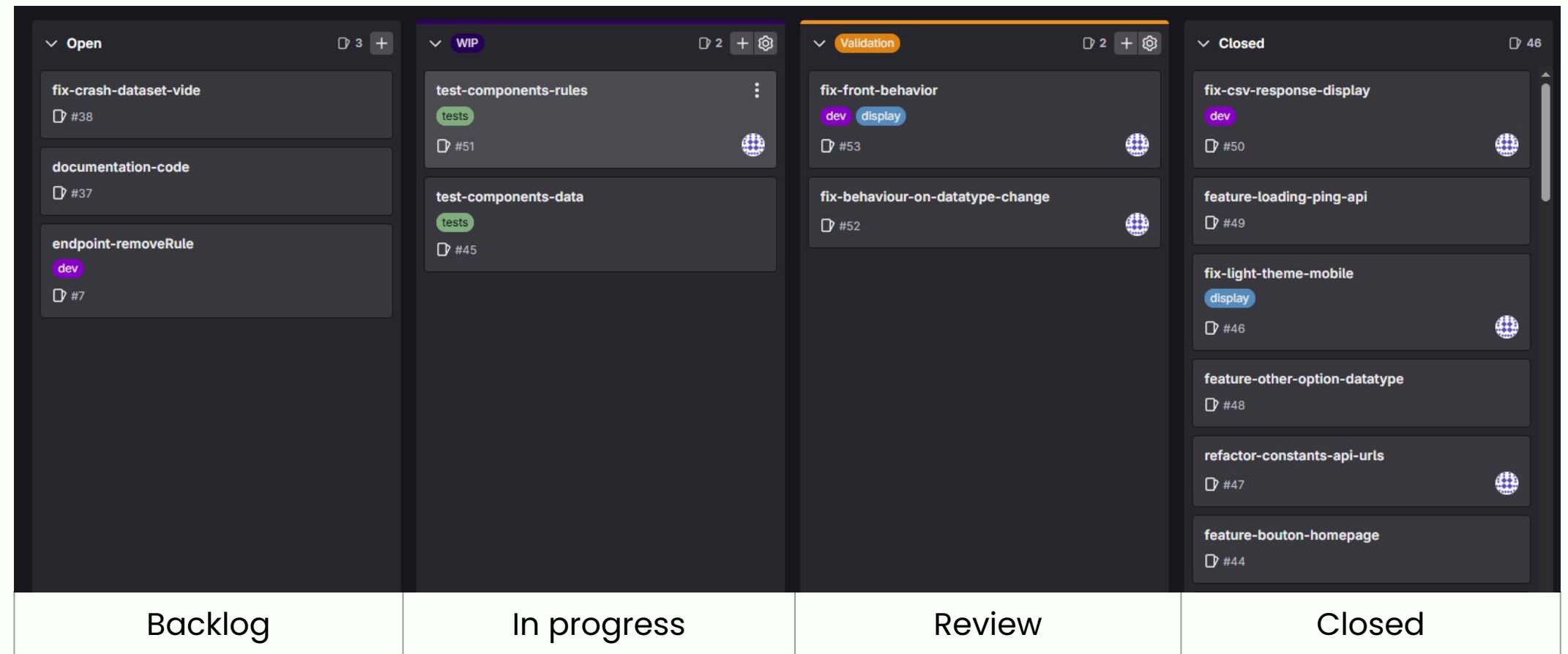
# Organisation de travail

[page 13]

[annexe 2]

Projet **GitLab** avec  
mon maître de stage :

- méthode Agile  
(daily meetings)
- suivi des tâches  
(issues, Kanban)
- chaque tâche  
= 1 branche Git  
dédiée

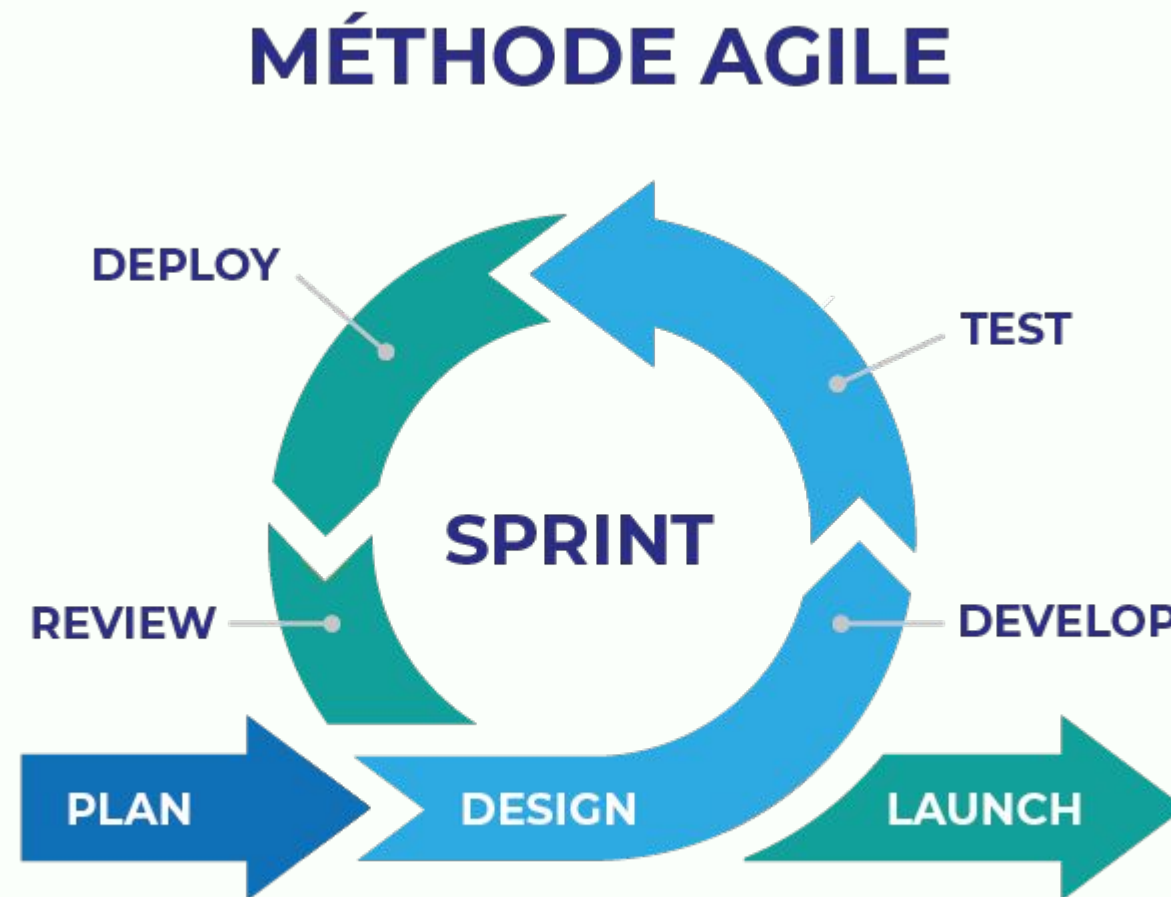




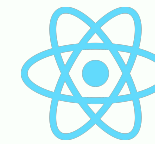
# Environnement de travail

[pages 13-14]

[annexes 3-4]



**Git** (v2.49)



**React** (v.19)



**TypeScript** (v5.7)



**Vite** (builder)



**NPM**  
(gestion des dépendances)



**Axios** (v1.9)  
(requêtes HTTP)



**Visual Studio Code** (IDE)



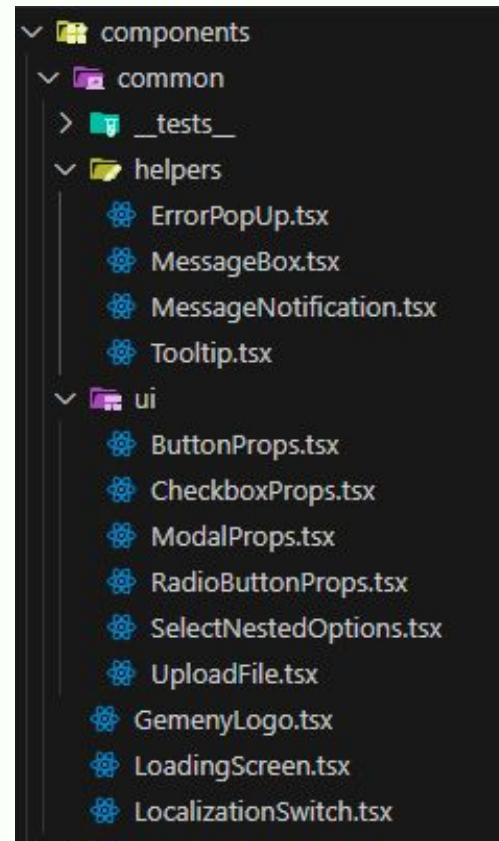
# Structure du projet

[page 15]

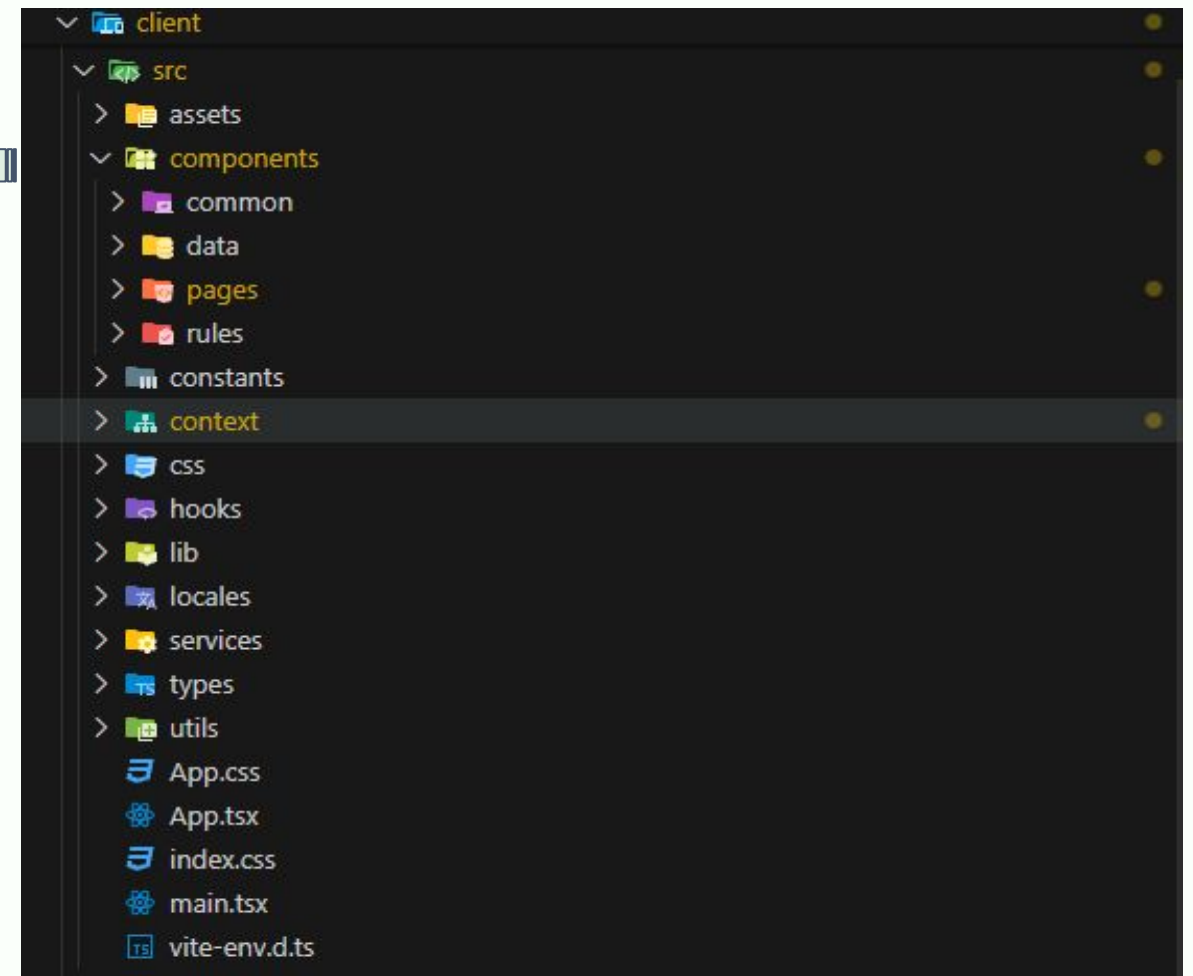
[annexe 5]

Nommage explicite,  
cohérent, conforme aux  
bonnes pratiques.

Regroupement des  
composants par  
thématique, usage.



*Séparation des  
composants*



*Extrait de l'arborescence du projet Gemeny\_Web*



# Réalisation de l'interface statique

[pages 17-18]

[annexe 6]

GENERAL

Essayer Gemeny

Dashboard Logout

Mon Profil

Mes jeux de données

Mes règles

ADMINISTRATION

Gestion utilisateurs

Ajouter utilisateur

Team Settings

Manage Your Team

Members

## Ajouter un nouvel utilisateur

Nom

e.g. alex@example.com

Email

e.g. alex@example.com

Role

USER

Mot de passe

\*\*\*\*\*

Ajouter utilisateur

```
<div className='column dashboard-content'>
<p className='title'>Ajouter un nouvel utilisateur</p>
<form className="box" onSubmit={handleAddNewUserClick}
  aria-label="Formulaire d'ajout d'utilisateur">
  <div className="field">
    <label className="label" htmlFor="nom-input">Nom</label>
    <div className="control">
      <input className="input" id="nom-input" type="text"
        placeholder="e.g. alex@example.com" required
        aria-label="Nom" value={identifiant}
        onChange={(e) => setIdentifiant(e.target.value)}
      />
    </div>
  </div>
</div>
```

Extrait de code du composant **RegisterUser.tsx**

- Validation HTML5 (**required**)
- Accessibilité (**aria-label**, **htmlFor**)

Formulaire d'enregistrement d'un utilisateur (dashboard Admin)



# Interface dynamique : édition de règle

[pages 19-21]

[annexe 7]

```
// Fonction pour mettre à jour la clé choisie
const handleKeyChange = (e: React.ChangeEvent<HTMLSelectElement>) => {
  const newKey = e.target.value;
  setSelectedKey(newKey);
  updateRule(ruleID, { keyArchetype: newKey });
};
```

```
const {t} = useTranslation();
return (
  <form><div className="control" style={{maxWidth: '350px'}}>
    <div className="select is-fullwidth">
      <select value={selectedType} onChange={handleTypeChange}>
        <option value="key">{t('rulesTable:SELECT_KEY_LINE')}</option>
        <option value="line">{t('rulesTable:CUSTOM_LINE')}</option>
      </select>
    </div>
    <div> { /* ternaire selon selectedType pour afficher une liste ou un input type text */ }
    {selectedType === 'key'
      ? (
        <div className="select is-fullwidth">
          <select value={selectedKey} onChange={handleKeyChange}>
            <option value="" disabled>{t('rulesTable:CHOOSE_KEY')}</option>
            <NestedSelectProps keys={groupedKeys} />
          </select>
        </div>
      )
      : (
        <input className="input" id="input-custom-line"
          type="text" placeholder={t('rulesTable:CL_PLACEHOLDER')}
          onChange={handleLineChange}
          value={customLine}
        />
      )
    }
  </div></form>
);
```

Choose dataset Reset

```
{
  "id": 1,
  "name": "John Doe",
  "email": "john.doe@example.com",
  "phone": "+1234567890"
}
```

Characters : 96/2000 ☒ JSON ☐ Other ☐ CSV

	Data source	Custom regex
	<div>Select key </div> <div>email </div>	<input type="text" value="Enter custom regex"/>
	<div>Select key </div> <div>name </div>	<input type="text" value="Enter custom regex"/>
	<div>Select key </div> <div>phone </div>	<input type="text" value="xx??x?x?xx"/>





# Configuration des appels API

[pages 23-24]

[annexes 9-10]

Instances AXIOS

Services API

Utilisation du service

Création des instances Axios

Par API

Gemeny (métier)

Auth (gestion users)

```
// Configuration de base pour l'API métier
const businessApi: AxiosInstance = axios.create({
  baseURL: GEM_API_URL,
  headers: {
    'Content-Type': 'application/json',
  }
});

// Configuration pour l'API utilisateur
const authApi: AxiosInstance = axios.create({
  baseURL: AUTH_API_URL || 'http://localhost:8080/api/users',
  headers: {
    'Content-Type': 'application/json',
  }
});
```

Création des services pour effectuer les requêtes API

Paramètres

Corps de requête

Token (si besoin)

```
export const authAPI = {
  /**
   * Authentifie l'utilisateur et retourne les tokens JWT
   * @param credentials - Email et mot de passe
   * @returns Le token de session utilisateur
   */
  login: async (credentials: LoginRequest): Promise<AuthResponse> => {
    const response = await authApi.post<AuthResponse>
      ('/auth/login', credentials);
    return response.data;
  },
};
```

Intégration des services dans les composants

Gère dynamiquement

Création de la requête

Affichage de la réponse

```
// Fonction de connexion
const login = async (email: string, password: string) => {
  const { user_info, access_token } = await authAPI.login({
    login: email,
    password
  }).catch(error => {
    // Transformer les erreurs Axios en erreurs métier
    const message = axios.isAxiosError(error)
      ? error.response?.data?.message ?? error.message
      : 'Erreur technique';
    throw new Error(message); // Recréer une Error propre
  });
};
```



# Tests & gestion des erreurs

[page 25]

[annexe 11]

		Preprod / dev	Prod / déployé	Cas d'utilisation
1	Utilisation de la console navigateur	✓	✗	<ul style="list-style-type: none"><li>débogage en mode développement</li><li>tests des interactions front/API en local</li></ul>
2	Affichage de messages d'erreur personnalisés	✗	✓	<ul style="list-style-type: none"><li>feedback intuitive pour l'utilisateur</li><li>affichage d'erreurs prévues pour l'utilisateur final de l'interface</li></ul>
3	Mise en place de tests unitaires, intégration	✓	✗	<ul style="list-style-type: none"><li>assurer la robustesse de l'application</li><li>en vue de la mise en place de CI/CD (pipeline de déploiement ...)</li></ul>

```
try {
  await login(identifiant, password);
} catch (error) {
  const message = error instanceof Error ? error.message : 'Erreur technique';
  // console.log("🚀 ~ handleLoginClick ~ identifiant:", identifiant)
  // console.log("🚀 ~ handleLoginClick ~ message:", message)
  setError(message);
}
```

Utilisation de log sur la console du navigateur

```
PASS src/components/common/__tests__/ui/ButtonProps.test.tsx (6.329 s)
PASS src/components/common/__tests__/ui/RadioButtonProps.test.tsx (6.348 s)

Test Suites: 2 passed, 2 total
Tests:       9 passed, 9 total
Snapshots:   0 total
Time:        18.309 s
Ran all test suites matching /Button/i.
```

Extrait et exécution de test unitaire sur mon composant 'bouton'

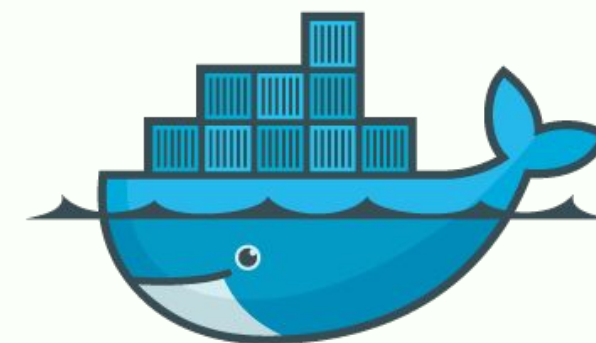


# Déploiement de l'application

- Déploiement régulier de l'interface **Gemeny\_Web** sur le domaine **Surge.sh** : accessible en ligne via une URL dédiée.
- L'API métier de Gemeny est géré par mon maître de stage, en utilisant **Docker**.
- Intégration : l'interface web est intégrée sur le site de l'entreprise via une iframe pour une expérience fluide.



**surge**



**docker**



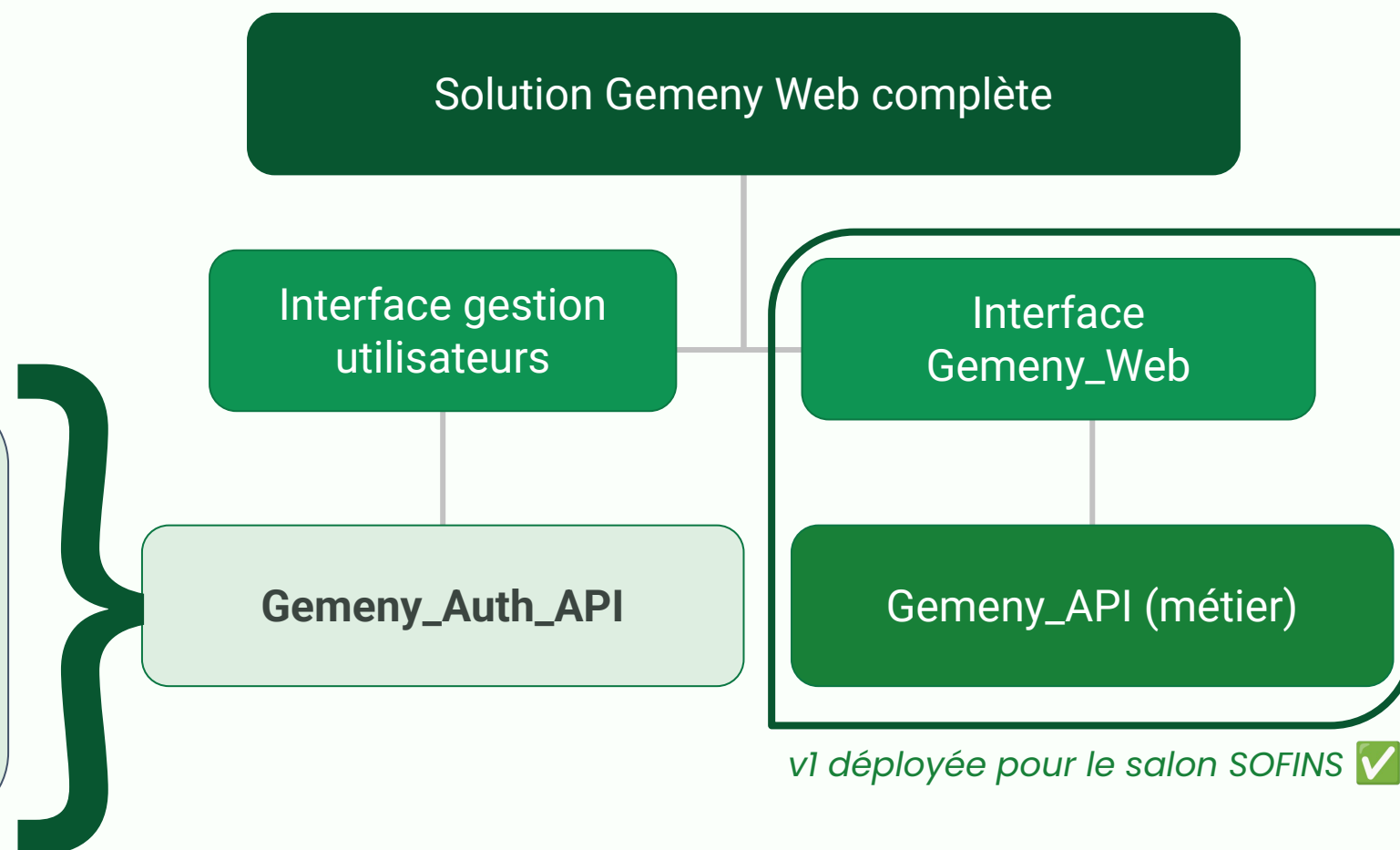
# Mes réalisations Back-end

[page 26]

[annexe 12]

## Mise en place d'un micro-service API\* pour la gestion d'utilisateurs

- Gérer l'inscription, la connexion, et les droits des utilisateurs
- Séparer le front-end du back-end pour plus de sécurité et de modularité
- Préparer l'évolution future de la plateforme (gestion des rôles, permissions, etc.)



*Architecture des services Gemeny Web*

**API\*** : Interface de Programmation d'Application






# Environnement de travail

[page 27]

[annexe 1]

## Mise en place de la base de données

- SGBD\* : **SQL** avec  MariaDB
- Administration : HeidiSQL

## Conception du micro-service

- Diagrammes : PlantUML
- Application de la méthode Merise
- Conception et visualisation :

*Looping*

\***SGBD** : Système de Gestion de Base de Données

## Développement de l'API



- IDE : IntelliJ IDEA Community



- Serveur local : Laragon



- Langage : Java 21



- Maven (gestion des dépendances)



- Framework Java :  
Spring Boot 3.4.3  
(idéal pour le web)



# Conception de l'API

[page 28]

[annexe 12]

## Points clés :

- Inscription, connexion, gestion des utilisateurs
- **API REST Stateless\***  
( pas de stockage de session côté serveur )
- Authentification sécurisée par **token (JWT)**
- Architecture évolutive, maintenable

**Stateless\*** : pas de stockage de session côté serveur, voir annexe 12

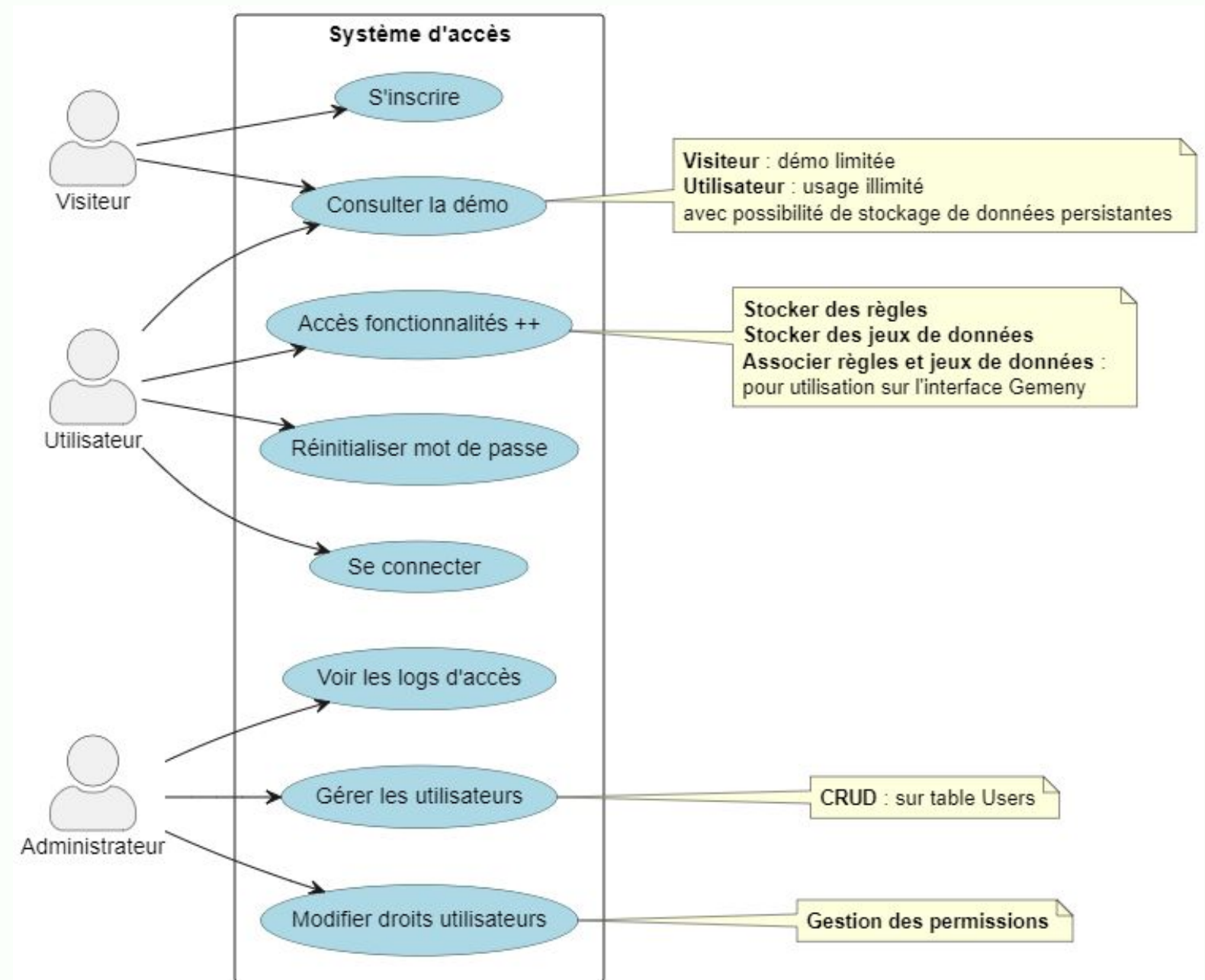


Diagramme cas d'utilisation du micro-service



# Dictionnaire de données

[pages 28-29]

[annexes 13-15]

## Dictionnaire de données pour la table Users

Colonne	Type	Description / Contrainte
id_user	INT (PK, AI)	Clé primaire auto-incrémentée
name	VARCHAR(50)	Nom de l'utilisateur
email	VARCHAR(100)	Adresse e-mail, unique
password	VARCHAR(255)	Mot de passe haché
id_role	INT (FK)	Clé étrangère vers GEM_ROLES(id_role), non nul

### Légende

- (PK) : Clé primaire
- (AI) : Auto-incrémentée
- (FK) : Clé étrangère
- Unique : Valeur unique dans la table

### Diagrammes UMLs :

- *diagramme de classe projet page 29*
- *diagramme de séquence **annexe** page 15*

## Objectifs du dictionnaire de données

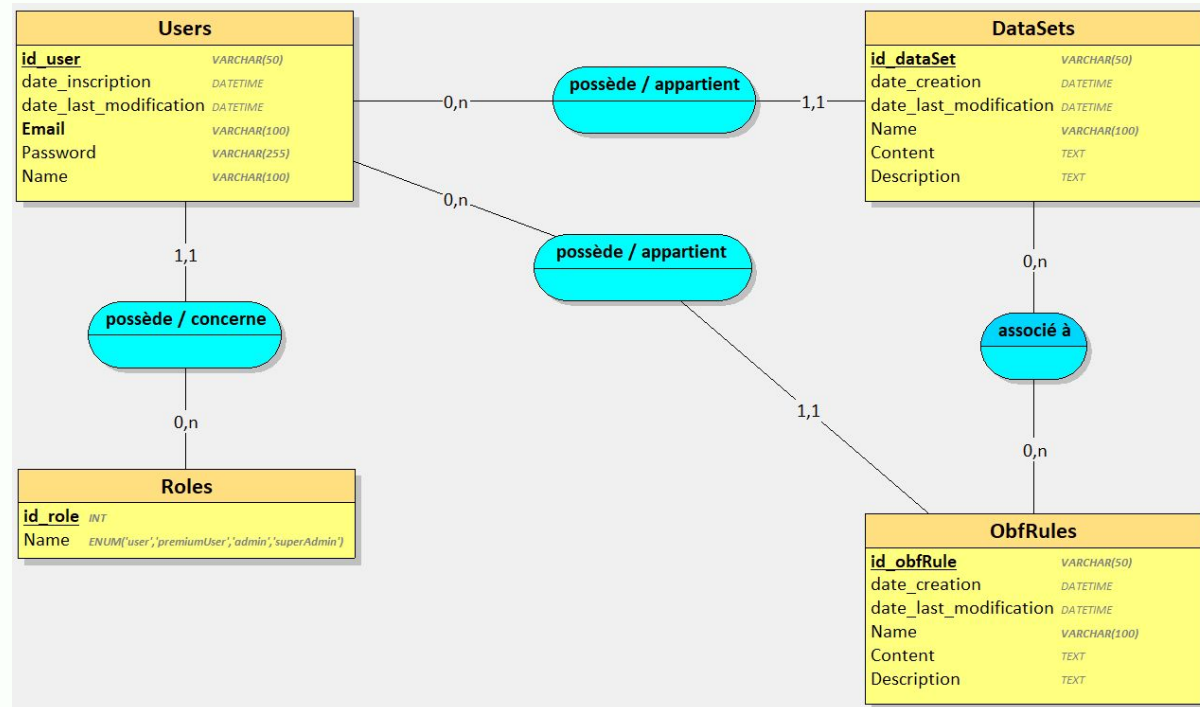
- Harmonisation des échanges front/back
- **Référence** unique pour la conception de la base de données et du serveur API
- Facilite la maintenance et l'évolution



# Conception de la base de données

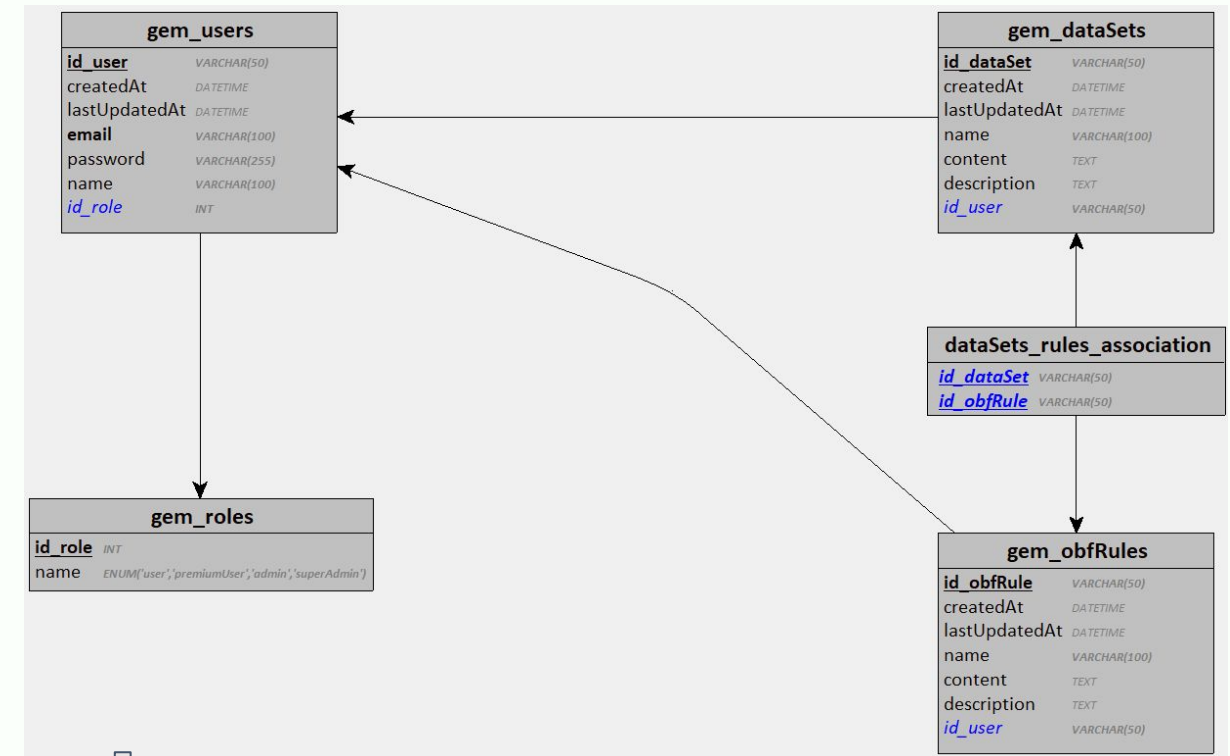
[pages 30-31]

Modèle Conceptuel de Données (MCD)



Entités principales et leurs relations

Modèle Logique de Données (MLD)



Tables SQL, clés étrangères, associations et table pivot

- Visualisation claire des entités et de leurs relations
- Passage du conceptuel (MCD) au logique (MLD) pour préparer la base SQL
- Mise en évidence des clés étrangères et des tables d'association



# Mise en place de la base de données

[pages 32-35]

- Initialisation de la base sous MariaDB avec Laragon
- Utilisation de HeidiSQL pour la gestion et l'administration (droits, sécurité)
- Scripts SQL pour la création des tables et des relations

```
CREATE DATABASE IF NOT EXISTS gemeny_db CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
USE gemeny_db;

CREATE TABLE gem_roles(
  id_role INT AUTO_INCREMENT,
  name ENUM('user','premiumUser','admin','superAdmin') NOT NULL DEFAULT 'user',
  PRIMARY KEY(id_role)
);

CREATE TABLE gem_users(
  id_user VARCHAR(50),
  createdAt DATETIME DEFAULT CURRENT_TIMESTAMP,
  lastUpdatedAt DATETIME ON UPDATE CURRENT_TIMESTAMP,
  email VARCHAR(100) NOT NULL,
  password VARCHAR(255) NOT NULL,
  name VARCHAR(100) NOT NULL,
  id_role INT NOT NULL,
  PRIMARY KEY(id_user),
  UNIQUE(email),
  FOREIGN KEY(id_role) REFERENCES gem_roles(id_role)
);
```

Extrait SQL pour la création de la base de données, tables users et roles

## Accès aux données, création de requêtes préparées

- Sécurité accrue ( - injections SQL)
- Performance (requête réutilisable)
- Gestion des paramètres

```
SELECT
  u.id_user,
  u.email,
  u.name,
  r.name AS role,
  u.createdAt,
FROM
  gem_users u
JOIN
  gem_roles r ON u.id_role = r.id_role
WHERE
  r.name = 'admin';
```

Exemple de requête préparée

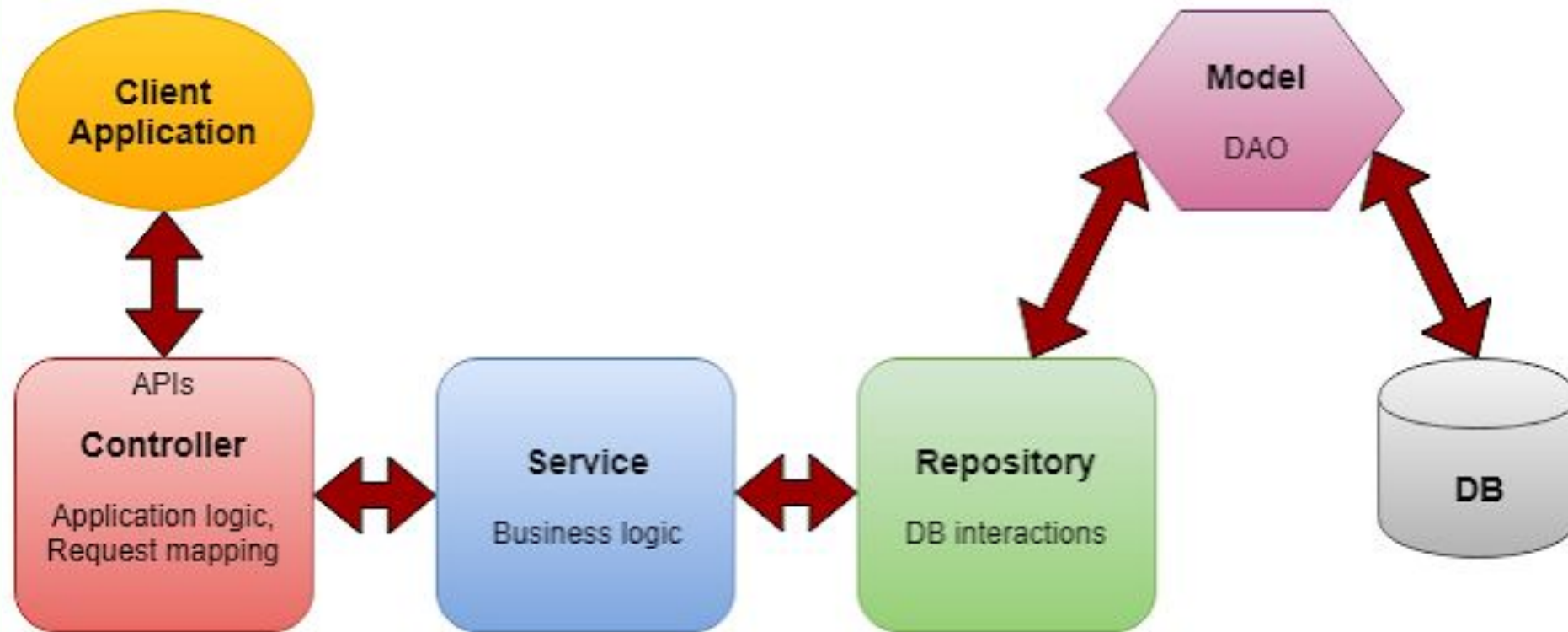




# Réalisation de la structure MVC

[page 36]

[annexe 16]



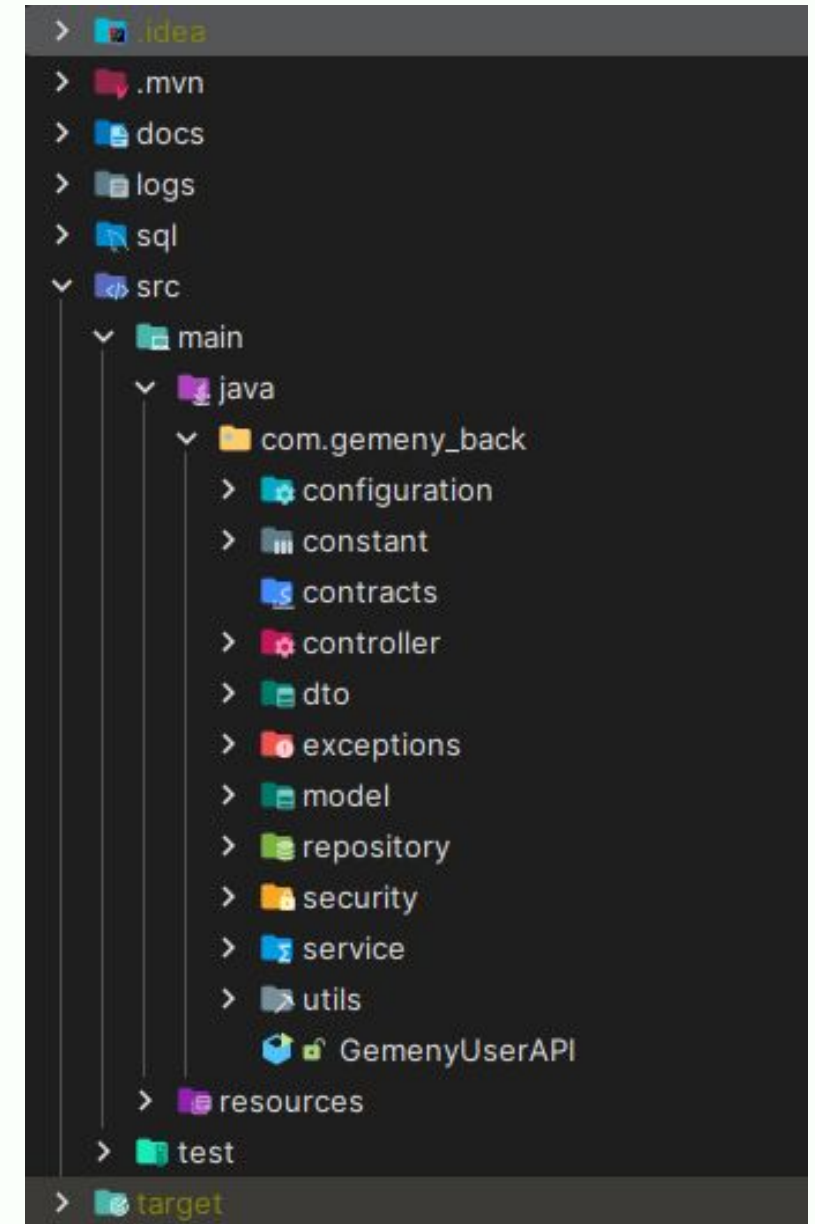
## Objectifs du design pattern **Model Vue Controller :**

- Séparation des responsabilités
- Modularité
- Maintenabilité

Utilisation de **DTO\*** entre les couches de l'application :

- robustesse
- sécurité
- cohérence

**DTO\*** = Data Transfer Object



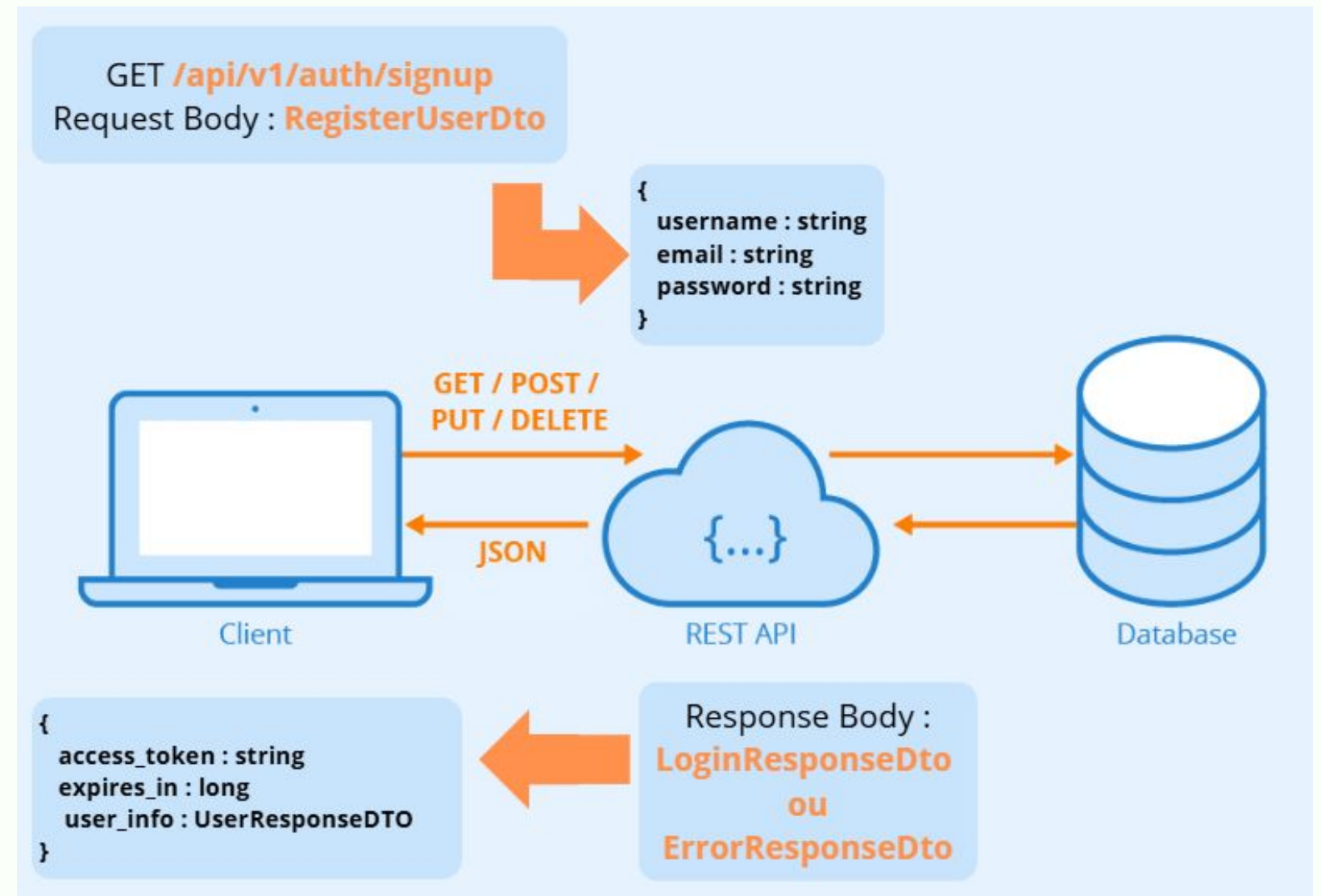
Structure du projet back-end



# Communication API → Interface utilisateur

[page 37]

- L'interface utilisateur (front-end) gère l'affichage, l'expérience et la validation côté client
- Les deux projets échangent des données structurées (JSON) via des requêtes **HTTP**
- Sécurité, maintenabilité et évolutivité.
- Approche **Stateless**, utilisation de **token JWT** pour les requêtes d'accès aux ressources



*Illustration de la séparation forte entre front (Vue) et back (Controller, Model, Service)*





# Développement des composants back

[page 38-39]

[annexe 17-20]

## User

## UserRepository

## UserService

## AuthController

Entité servant de modèle à la base de données, aux DTOs :

- Représente l'entité utilisateur
- Attributs, champs essentiels
- Annotations JPA pour le mapping base de données

Composants d'accès à la base de données (table Users) :

- Méthodes CRUD (*Create, Read, Update, Delete*)
- Requêtes personnalisées (*ex: findByEmail, findByRole*)
- Utilisation de requêtes préparées pour la sécurité

Logique métier liée aux utilisateurs :

- Validation des données (*unicité email, format ...*)
- Gestion du hashage de mot de passe
- Attribution des rôles
- Gestion des exceptions métiers

Gère les endpoints d'authentification :

- Routes : */signup, /login, /profile ...*
- Récupère et valide les requêtes du front
- Orchestration des services
- Gestion des statuts HTTP et des réponses
- Sécurisation des endpoints, restriction d'accès

### Documentation complémentaire

- Dictionnaire de méthodes (annexe page 19)
- Dictionnaire des endpoints (annexe page 21)



# Composants du MVC en pratique

[page 38-39]

[annexe 17-20]

```
public User createUser(RegisterUserDto input, RoleEnum role) {
    if (role == RoleEnum.SUPER_ADMIN) {
        log.error("SuperAdmin creation is unauthorized");
        throw new AuthException(AuthError.FORBIDDEN_CREATION);
    }
    if (userRepository.existsByEmail(input.email())){
        log.error("Email already used");
        throw new AuthException(AuthError.ALREADY_EXISTS);
    }
    if (userRepository.existsByUsername(input.username())){
        log.error("Username already used");
        throw new AuthException(AuthError.ALREADY_EXISTS);
    }
    User newUser = new User()
        .setUsername(input.username())
        .setEmail(input.email())
        .setPassword(passwordEncoder.encode(input.password()))
        .setStatus(StatusEnum.ACTIVE)
        .setRole(roleService.findByName(role).orElseThrow(
            () -> new RoleException(RoleError.NOT_FOUND))
        );
    try {
        return (userRepository.save(newUser));
    } catch (Exception e) {
        log.error(e.getMessage());
        throw new ServerException(ServerError.DATABASE_ISSUE);
    }
}
```

Exemple de service, méthode **createUser()** de **UserService**

- Centralise la logique métier de création d'utilisateur
- Valide les données reçues (unicité email, format, etc.)
- Hash le mot de passe avant enregistrement  
(méthode *passwordEncoder* de *Spring Security*)
- Attribue le rôle par défaut à l'utilisateur
- Appelle le repository pour sauvegarder l'utilisateur dans la table `gem_Users` de la base de données
- Gère les exceptions (personnalisés) et retourne un résultat adapté



# Gestion des routes protégées

[page 40]  
[annexes 21-22]

- Sécurisation centralisée des endpoints
- Contrôle d'accès selon le rôle utilisateur

Endpoints /api/v1/admin

Méthode	Endpoint	Corps attendu	Réponse principale	Description	Rôle requis
GET	/allUsers	-	List<UserBasicDTO>	Tous les utilisateurs	ADMIN/SUPER_ADMIN
GET	/roles	-	List<RoleEnum>	Tous les rôles	ADMIN/SUPER_ADMIN
POST	/createUser	RegisterUserDto	UserBasicDTO	Créer utilisateur	ADMIN/SUPER_ADMIN
POST	/createAdmin	RegisterUserDto	UserBasicDTO	Créer admin	SUPER_ADMIN
POST	/createSuperAdmin	RegisterUserDto	UserBasicDTO	Créer super admin	SUPER_ADMIN

Extrait du dictionnaire des endpoints

- Authentification par token JWT



Exemple d'implémentation (AdminController) voir page 40 :

Dans le contrôleur AdminController, les annotations **@PreAuthorize** permettent de restreindre l'accès selon le rôle, **transmis par le token JWT** lors de la requête.

**@PreAuthorize("hasAnyRole('ADMIN','SUPER\_ADMIN')")** sur la classe : toutes les routes sont accessibles uniquement aux administrateurs.

Faibles ( <b>OWASP</b> )	Description	Conséquences	Contre-mesures appliquées
Injection SQL	Insertion de code SQL malicieux dans une requête	Vol, modification ou suppression de données	Utilisation de requêtes préparées et ORM
Cross-Site Scripting (XSS)	Insertion de scripts malicieux dans les pages web	Vol de session, redirection, défiguration	Échappement des données affichées, validation côté serveur
Cross-Site Request Forgery	Exploitation de la session d'un utilisateur pour réaliser des actions à son insu	Actions non autorisées, modification de données	Vérification du token CSRF, authentification JWT
Fuite de données sensibles	Mauvaise gestion des données confidentielles (logs, réponses API, etc.)	Exposition d'informations critiques	Utilisation de DTOs, filtrage des champs, logs anonymisés
Brute force sur les mots de passe	Tentatives répétées de connexion avec différents mots de passe	Prise de contrôle de comptes	Limitation du nombre d'essais, délai, hashage fort (BCrypt)

**Merci de votre attention.**

**Place à la démonstration !** 





# Dossier de Projets

# Rapport de Stage

Développeur Web & Web Mobile

Stage réalisé avec **Gemeny Software**

du 17 février au 02 mai 2025

Durée : 10 semaines

par Yoann Le Goff

