

VIEILLARD--GRELET Tom

Développement d'application web



BTS SIO2 - 2023/2024

SOMMAIRE

1. Expression des besoins.....	3
1.1. Contexte, domaine, processus métier.....	3
1.2. Demandeur, acteurs, utilisateurs.....	3
1.3. Étude de l'existant, diagnostic.....	4
1.4. Description de la demande, objectifs, bénéfices attendus.....	5
1.5. Spécifications fonctionnelles.....	6
2. Conception, Spécifications Techniques.....	8
2.1. Description de la solution.....	8
2.2. Outils logiciels de la solution.....	9
2.3. Architecture matérielle et logicielle de la solution (schémas).....	10
2.4. Besoins techniques, ressources (humaines, matérielles, logicielles et budgétaires, coûts).....	13
2.5. Analyse des données (modélisation, diagramme de classes, schéma relationnel)....	14
2.6. IHM (interfaces homme-machine), Maquettage.....	19
2.7. Conduite de projet : décomposition en tâches, structure d'équipe, planning (Gantt), durée.....	23
3. Développement.....	24
3.1. Réalisation des interfaces et programmes conformes aux spécifications fonctionnelles attendues.....	24
3.2. Dossier de programmation, codes sources documentés et commentés.....	29
4. Exploitation, Mise en production.....	34
4.1. Tests (unitaires, fonctionnels, intégration).....	34
4.2. Installation, déploiement, fonctionnement éventuel en double avec l'ancienne procédure.....	41
5. Bilan.....	42

1. Expression des besoins

1.1. Contexte, domaine, processus métier

Dans le cadre de ma formation en BTS SIO, j'ai réalisé un stage d'une durée de 6 semaines dans la société **PARTNER Informatique**. Le travail à réaliser se concentre sur le développement d'application web.

1.2. Demandeur, acteurs, utilisateurs



Le siège social de la société **PARTNER Informatique** est basé dans la Saône-et-Loire à Mâcon (lieu du stage) et existe depuis 1986.

L'entreprise a pour secteur d'activité "Infrastructure informatique et téléphonie" et "Logiciels de gestion et d'ERP". Celle-ci est en liaison avec 600 clients fidèles. Elle développe plusieurs logiciels de gestion telle que Carbone 14.

Entreprises de négocié et sociétés de services	Artisans et BTP	Industries et sous-traitance	Enseignements, associations et collectivités	Agro-alimentaires, agricoles et viticoles

Domaine professionnel en contact avec PARTNER Informatique

Le service informatique est composé d'une quarantaine d'employés répartie dans les activités suivantes :

- Techniciens et ingénieurs pour le service infrastructure.
- Consultants, développeurs et chefs de projets pour le service « Gestion & ERP »
- Chargés d'affaires (Chaque client a un interlocuteur dédié et stable dans le temps)

1.3. Étude de l'existant, diagnostic

Dans le cadre de mon stage, plusieurs technologies informatiques sont à utiliser. Les principaux langages utilisés sont le Java et le TypeScript. Les technologies exploitées sont Spring Boot et Angular.

1.3.1. Mission 1 : Développement d'une API lié à une base de données

Pour acquérir les compétences nécessaires aux développements des futurs projets, la création d'une API REST liée une base de données MariaDB a été développé pour maîtriser l'outil Spring Boot.

1.3.2. Mission 2 : Ajout de fonctionnalités sur une application Web existante

Pour développer mes compétences, j'ai décidé de m'intéresser au projet web réalisée par un autre stagiaire présent pendant mon stage. Le projet est une application web utilisant le framework Angular 17. Le but de l'application, nommé "DevOps manager", est de montrer d'autres applications web et de surveiller leur état.

Une fois le projet terminé, il sera déployé et utilisé par l'entreprise pour la gestion de leurs applications web.

1.3.3. Mission 3 : Récupération d'informations présentent sur un site (Scraping)

Le service de communication de l'entreprise souhaite récupérer l'adresse email, le nom-prénom du gérant et l'adresse géographique de tous les notaires des départements aux alentours. Ces informations peuvent être récupérées via le site <https://www.notaires.fr/fr>.

1.3.4. Mission 4 : Développement d'une application Web liée à une API (continuité de la mission 1)

Afin de maîtriser les bases du Framework Angular 17, j'ai créé ma propre application Web. Cette application communique avec l'API réalisée en début de stage (Mission 1).

1.3.5. Mission 5 : Évolution d'une application Web existante

Le club "CHARNAY BASKET BOURGOGNE SUD" organise parfois des événements sportifs. Pendant ces périodes, il héberge des joueuses dans son internat. Le club souhaite avoir une application Web leur permettant de gérer les informations de l'internat (chambres, types de repas...), de programmer et de visualiser des événements et d'assigner à des joueuses des chambres et des repas durant ces périodes.

1.4. Description de la demande, objectifs, bénéfices attendus

1.4.1. Mission 1 : Développement d'une API lié à une base de données

Afin de pouvoir maîtriser les différentes technologies utilisées, un petit projet en autonomie m'a été donné pour apprendre à utiliser les nouvelles notions de celles-ci. Le projet consiste à développer une API REST avec Spring Boot qui communique avec une base de données MariaDB. Le thème du projet porte sur l'un de mes centres d'intérêt qui est le jeu vidéo. Ce thème me permet d'utiliser mes connaissances pour réaliser une application dans un contexte que je maîtrise.

Le projet est uniquement à titre d'entraînement et ne sera pas utilisé par l'entreprise par la suite.

1.4.2. Mission 2 : Ajout de fonctionnalités sur une application Web existante

Les fonctionnalités à ajouter consistent à ajouter un système d'actualisation du "token" de connexion, rédiger l'utilisateur vers la page d'accueil si celui-ci est déjà connecté et d'ajouter un système de déconnexion automatique au bout d'un certain temps d'inactivité.

1.4.3. Mission 3 : Récupération d'informations présentent sur un site (Scraping)

Le service de communication me demande de développer une application permettant de récupérer des informations (email, nom-prénom du gérant et l'adresse géographique) de notaires présentent sur le site <https://www.notaires.fr/fr>. Les informations récupérées seront stockées et structurées dans un fichier CSV.

1.4.4. Mission 4 : Développement d'une application Web liée à une API (continuité de la mission 1)

Dans le cadre de cette mission, l'objectif principal était de développer une application web interagissant avec l'API précédemment créée. Cette tâche visait à renforcer mes compétences en utilisant le Framework Angular 17 tout en mettant en pratique les concepts de communication entre une application frontale et une API back-end. L'application devait permettre aux utilisateurs d'interagir facilement avec les données fournies par l'API. Les bénéfices attendus comprenaient une meilleure maîtrise des technologies utilisées, ainsi qu'une entrée dans le domaine du développement web full-stack.

Le thème et la finalité du projet sont les mêmes que la mission 1.

1.4.5. Mission 5 : Évolution d'une application Web existante

On me demande de terminer l'application et de retravailler la partie d'édition des événements pour qu'elle soit plus simple à utiliser. L'application permettra une meilleure gestion de l'hébergement et de l'internat.

1.5. Spécifications fonctionnelles

1.5.1. Mission 1: Développement d'une API lié à une base de données

L'API créé doit pouvoir communiquer avec une base de données selon les quatre opérations de base pour la persistance des données : CRUD (Create, Read, Update, Delete). Par exemple, celle-ci me permettra de récupérer une liste de jeu ou de modifier le prix d'un jeu sur une plateforme spécifique.

La structure du projet est celle de la technologie Spring Boot.

1.5.2. Mission 2 : Ajout de fonctionnalités sur une application Web existante

Pour la partie régénération de token :

En cas d'expiration du token de connexion, celui-ci doit être régénéré à partir de son "refresh_token". Le "refresh_token" est généré à la connexion de l'utilisateur. Si le "refresh_token" est, lui aussi, expiré, alors l'utilisateur sera déconnecté.

Pour la partie d'auto-redirection :

Lorsque qu'un utilisateur se connecte au site, un couple token/refresh_token est créé et enregistré en local sur le navigateur. Si l'utilisateur accède au site en passant par l'URL classique ou de login (exemple : www.domaine.fr ou www.domaine.fr/login) et que celui-ci s'était déjà connecté auparavant, alors le token sera régénéré. En cas de succès, l'utilisateur sera redirigé vers l'accueil sinon vers la page de login.

Pour la partie de déconnexion pour inactivité :

Lorsque aucune action (ex : mouvement de souris, utilisation du clavier, ...) n'est réalisée par l'utilisateur au bout d'une minute, une compte-à-rebours de 30 minutes est lancé. Si l'utilisateur n'a pas fait d'action pendant les 29 prochaines minutes, un popup informatif s'affiche avec la question "Etes-vous inactif ?" suivie du temps avant déconnexion. Si l'utilisateur ne réagit toujours pas durant cette dernière minute, il est déconnecté.

1.5.3. Mission 3 : Récupération d'informations présentent sur un site (Scraping)

L'application consiste à récupérer le code source de la page du site <https://www.notaires.fr/fr>, via du "web scraping", pour récupérer l'URL de la page de chaque notaire. Une fois les liens récupérés et stockés par le programme Python, le code source de chaque page de notaire récupéré par les URL stockés précédemment sont scannés. Les informations d'email, de nom-prénom et d'adresse géographique sont ensuite récupérés et stockés dans un fichier Excel.

1.5.4. Mission 4 : Développement d'une application Web liée à une API (continuité de la mission 1)

L'objectif était de créer une application web interagissant avec l'API de la mission 1 pour récupérer, afficher, modifier, ajouter et supprimer des données via une interface Web. J'ai créé des éléments avec le Framework Angular 17 pour manipuler les données via des boutons et des formulaires Web.

J'ai utilisé une architecture pour l'application comme celle utilisée dans l'entreprise.

1.5.5. Mission 5 : Évolution d'une application Web existante

L'application doit pouvoir gérer les chambres (numéro, capacité, type de chambre), les types de repas, les joueuses et les évènements/périodes. L'utilisateur doit pouvoir gérer les réservations de chaque jour pour chaque période.

2. Conception, Spécifications Techniques

2.1. Description de la solution

2.1.1. Mission 1 : Développement d'une API lié à une base de données

La solution consiste à :

- Créer une base de données MariaDB
- Développer une API REST en CRUD
- Faire communiquer l'API et la base de données

2.1.2. Mission 2 : Ajout de fonctionnalités sur une application Web existante

Les fonctionnalités à ajouter consistent à :

- Régénérer le token de connexion à partir de son refresh_token en cas d'expiration de celui-ci.
- Rediriger automatique l'utilisateur sur la page d'accueil si celui-ci est déjà connecté et sur la page de login dans le cas contraire.
- Développer un système de détection d'inactivité et de déconnexion automatique.

2.1.3. Mission 3 : Récupération d'informations présentent sur un site (Scraping)

La solution consiste à :

- Créer un programme Python permettant de récupérer le code source de la page du site <https://www.notaires.fr/fr>
- Récupérer et stocker les URL de page de notaire via le code source récupéré
- Récupérer le code source de chaque URL et de filtrer et stocker les informations notaire dans un fichier CSV
- Créer une interface web permettant de faciliter le scraping et de vérifier les informations récupérées

2.1.4. Mission 4 : Développement d'une application Web liée à une API (continuité de la mission 1)

La solution consiste à :

- Créer une application Web avec le Framework Angular 17
- Lier l'API de la mission 1 au projet
- Communiquer avec l'API avec de manipuler les données de la base de données

2.1.5. Mission 5 : Évolution d'une application Web existante

La solution consiste à :

- Corriger les erreurs de l'application
- Réaliser une nouvelle interface pour les évènements et l'assignation de chambre
- Développer les nouvelles interfaces
- Porter la version d'Angular du projet vers Angular 17

2.2. Outils logiciels de la solution

Logiciels et service utilisés :



Langages utilisés :

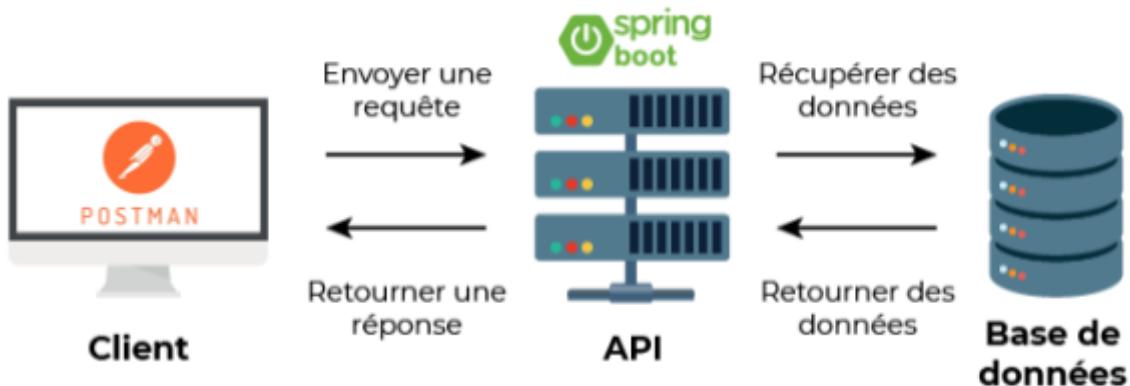


TypeScript

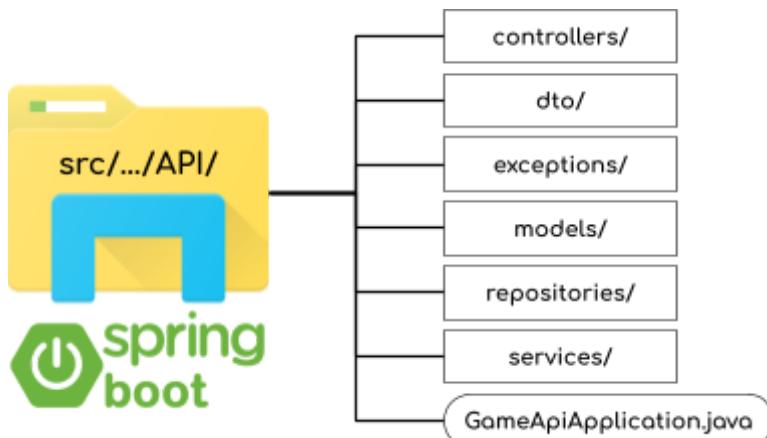


2.3. Architecture matérielle et logicielle de la solution (schémas)

2.3.1. Mission 1: Développement d'une API lié à une base de données

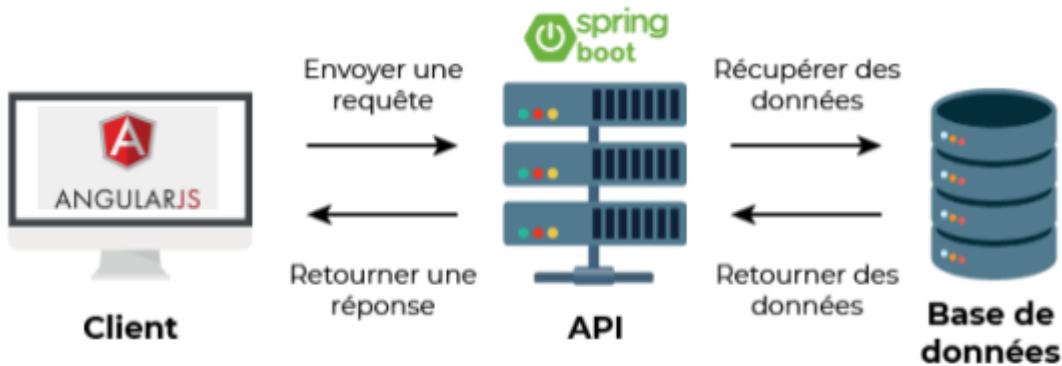


Arborescence du projet :



- Le dossier "controllers/" contient les classes qui gèrent les requêtes HTTP et dirige le trafic vers les services appropriés.
- Le dossier "dto/" contient une copie des classes models personnalisée pour chaque type de traitement (ex: ReadCategorieDTO pour la lecture, CreateCategorieDTO pour la création et UpdateCategorieDTO pour la modification d'information).
- Le dossier "exceptions/" contient des classes d'exceptions personnalisées.
- Le dossier "models/" contient les classes métiers des entités de la base de données.
- Le dossier "repositories/" contient les classes qui fournissent un accès aux données et gère la persistance des objets métier dans la base de données.
- Le dossier "services/" contient les classes avec la logique métier de l'application et encapsule les opérations métier disponibles pour les contrôleurs.
- "GameApiApplication.java" est la classe principale pour gérer le projet.

2.3.2. Mission 2 : Ajout de fonctionnalités sur une application Web existante

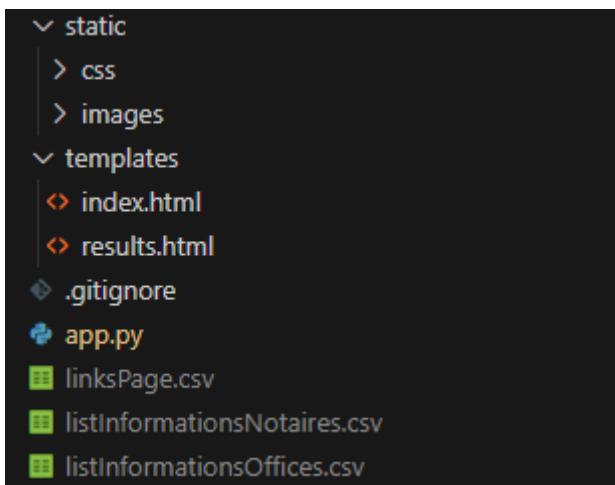


L'application ayant été développée par l'autre stagiaire plusieurs semaines avant mon arrivée, ne connaissant pas encore tout du framework Angular et ayant participé sur le projet principalement en tant qu'observateur et pour aider au développement des fonctionnalités demandé lors de ma deuxième semaine de stage, je n'ai pas pu me concentrer pleinement à la structuration de projet et de son arborescence.

→ Je n'ai pas récupéré de grandes informations pour la partie 2.5.

2.3.3. Mission 3 : Récupération d'informations présentent sur un site (Scraping)

Arborescence du projet :



→ Le dossier "templates/" contient les pages HTML :

- "index.html" contient le formulaire renseignant l'URL du site à scraper, le nombre de pages à scanner et un extrait de lien commun aux pages de notaire.
- "results.html" permet d'avoir un affichage dans un tableau des informations récupérées.
- Le fichier "app.py" contient les méthodes permettant de scraper et de récupérer les informations d'un site.
- Le fichier "linksPage.csv" contient les liens vers les pages de notaire récupérées.
- Le fichier "listInformationsNotaires.csv" contient les informations de chaque notaire récupérées via les liens de "linksPage.csv".

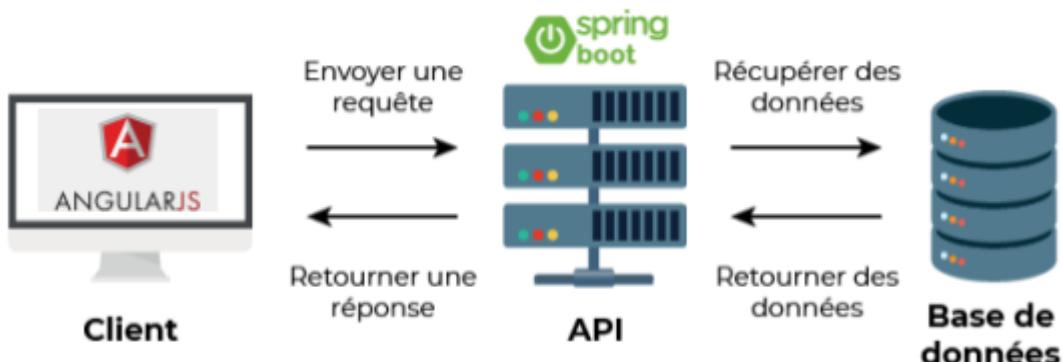
2.3.4. Mission 4 : Développement d'une application Web liée à une API (continuité de la mission 1)

Arborescence du projet :

```

    < src
      < app
        > components      → Contient les pages et les composants (formulaire, cards, ...)
        > models          → Contient les modèles pour les données manipulées
        > services         → Contient les méthodes permettant de communiquer avec l'API
        # app.component.css
        < app.component.html
        TS app.component.spec.ts   → Fichiers de configurations
        TS app.component.ts
        TS app.config.server.ts
        TS app.config.ts          → Fichier pour la gestion des routes
        TS app.routes.ts          → Contient les images du projet
        > assets
    
```

2.3.5. Mission 5 : Évolution d'une application Web existante



La mission comporte deux applications : une application Web avec Angular pour le front et une API lié à une base de données MySQL pour le back.

La structure de l'application front est sous la même forme que la mission 4 (*partie 2.3.4*). La structure de l'API est la même que la mission 1 (*partie 2.3.1*).

2.4. Besoins techniques, ressources (humaines, matérielles, logicielles et budgétaires, coûts)

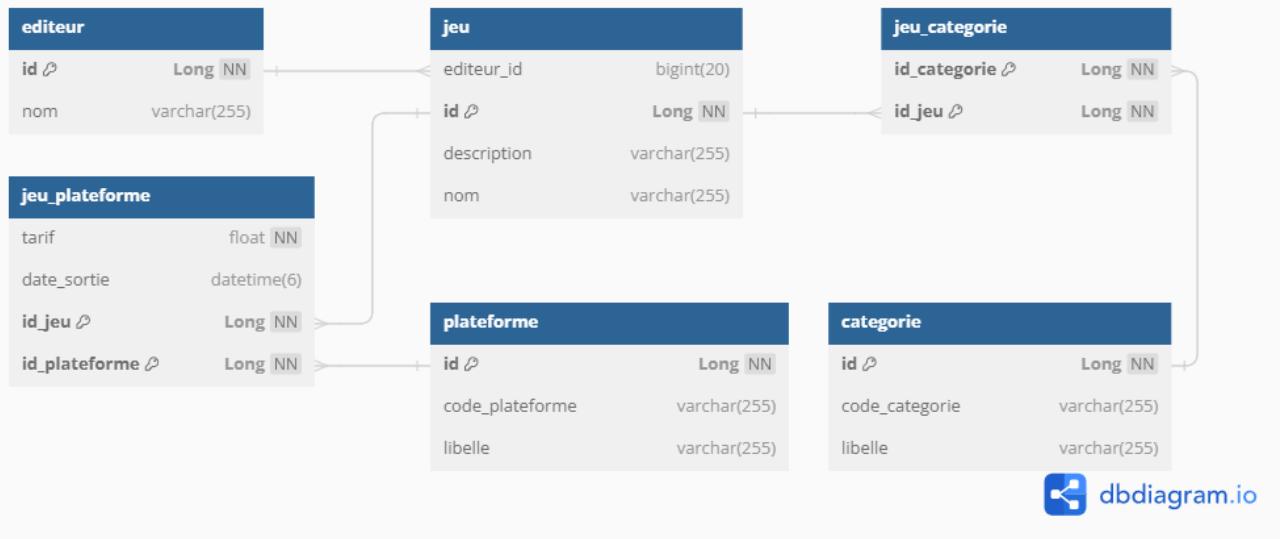
Ressources matérielles : Dans le cadre du stage, l'entreprise m'a mis à disposition un ordinateur pour développer les différents projets.

Ressources logicielles : L'entreprise met à disposition des dépendances, pour les projets, contenant des méthodes pratiques et utiles pour une meilleure efficacité de développement.

2.5. Analyse des données (modélisation, diagramme de classes, schéma relationnel)

2.5.1. Mission 1: Développement d'une API lié à une base de données

Base de données de l'API:



Le thème de la base de données est basé sur la gestion de jeu vidéo avec des éditeurs, des catégories de jeu et les plateformes sur lesquelles sont disponibles les jeux.

Classes du projet :

Méthodes de classe service (ex : JeuService.java) :

```
public ReadJeuDTO createJeu(CreateJeuDTO createJeuDTO)
public ReadJeuDTO updateJeu(String query, UpdateJeuDTO updateJeuDTO)
public Page<ReadJeuDTO> searchJeux(String query, Integer page, Integer size, String orderBy)
public void deleteJeu(final Long id)
public ReadJeuDTO getFullReadEntityDTO(ReadJeuDTO readJeuDTO)
```

Méthodes de classe controller (ex : JeuController.java) :

```
@PostMapping("/jeu")
public ResponseEntity<ReadJeuDTO> createJeu(@RequestBody CreateJeuDTO jeuDTO)

@GetMapping("")
public Iterable<ReadJeuDTO> getAllJeux()
```

```

@GetMapping("/search")
public Page<ReadJeuDTO> getJeux(@RequestParam String query, @RequestParam(defaultValue = "0")

@GetMapping("/jeu/{id}")
public ReadJeuDTO getJeu(@PathVariable("id") final Long id)

@PutMapping("/jeu/{id}")
public ResponseEntity<ReadJeuDTO> updateJeu(@PathVariable("id") final Long id,
    @RequestBody UpdateJeuDTO jeuDTO)

@DeleteMapping("/jeu/{id}")
public void deleteJeu(@PathVariable("id") final Long id)

```

Classes métiers pour la lecture, l'écriture et la modification d'entité :

ReadJeuDTO.java	CreateJeuDTO.java et UpdateJeuDTO.java
private Long id; private String nom; private String description; private ReadEditeurJeuDTO editeur; private List<Long> idCategories = new ArrayList<>(); private List<ReadJeuPlateformesDTO> jeuPlateformes = new ArrayList<>();	private String nom; private String description; private IdDTO editeur;

La classe **ReadJeuDTO** est utilisée pour l'affichage d'une entité complète.

Les classes **CreateJeuDTO** et **UpdateJeuDTO** sont uniquement utilisées pour traiter une entité.

Par exemple :

- Lors de la création d'une entité, l'ID étant automatiquement généré, il n'y a pas besoin de le renseigner.
- Pour la modification d'une entité, on souhaite uniquement modifier les champs de l'entité identifiée via son ID.
- Pour la récupération des informations d'une entité, on a besoin de tous ses champs y compris son ID.

2.5.2. Mission 3 : Récupération d'informations présentent sur un site (Scraping)

Méthode dans le fichier app.py:

```
// Redirige vers la page index.html
@app.route('/')
def index():

// Récupère les informations du formulaire de la page index.html et redirige vers la
page results.html
@app.route('/scrape', methods=[ 'POST'])
def submit():

// Récupère et stocke le lien des pages Notaire via les paramètres de recherche
def scrapeLinks(urlSiteCible, nombrePage, genericTermeInURL):

// Récupère les informations des pages Notaire via les liens stockés
def scrapeDatas(genericTermeInURL):

// Redirige vers la page results.html
@app.route('/results')
def results():

// Envoie au client le fichier et le fait télécharger
@app.route('/download')
def download():
```

2.5.3. Mission 4 : Développement d'une application Web liée à une API (continuité de la mission 1)

Base de données de l'API de la mission 1 utilisée.

Classes du projet :

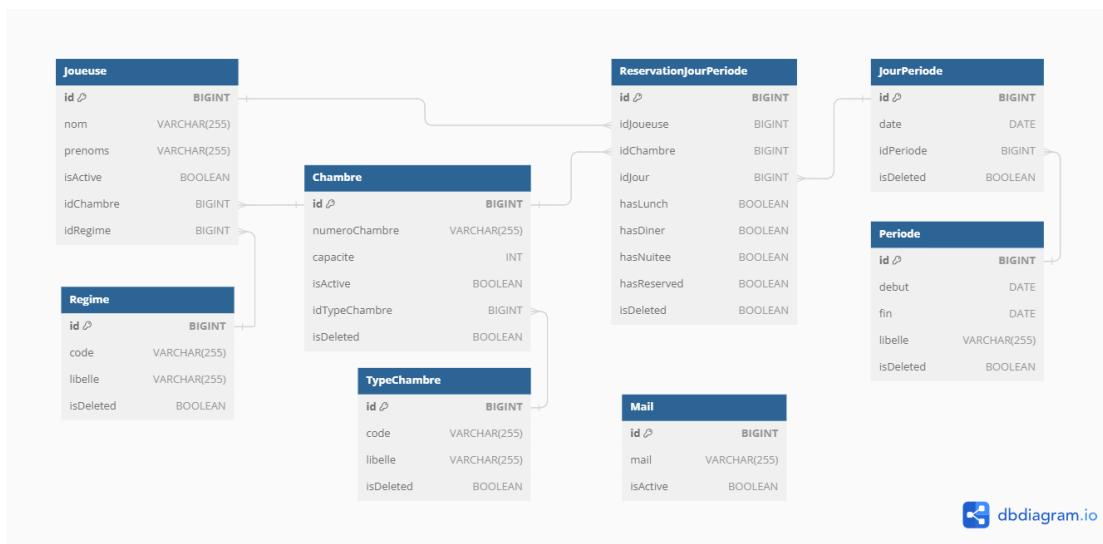
Méthodes de la classe game.service.ts (même structure pour les autres services) :

```
getAllGames()
searchGames(query: string = "id>0", page: number = 0, size: number = 999, orderBy: string = "nom", order: string = "ASC")
getGameInfos(gameId: number)
create(data: any): Observable<any>
update(gameId: number, data: any)
delete(gameId: number): Observable<any>
```

Exemple classe métier :

game.model.ts	game.model.ts
<pre>id?: number; nom!: string; description!: string; editeur!: Editeur idCategories?: number[] = []; JeuPlateforme?: JeuPlateforme[] = [];</pre>	<pre>jeu!: Game; plateforme!: Plateforme; dateSortie!: Date; tarif!: number;</pre>

2.5.4. Mission 5 : Évolution d'une application Web existante



Les classes de l'API prennent la même forme que l'API de la mission 1 (*partie 2.5.1*).

Exemple de models de données :

Model de la table Periode pour la base de données

```
public class Periode extends BaseEntity {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private Date debut;
    private Date fin;
    private String libelle;

    @Transient
    private List<JourPeriode> jours;
}
```

ModelDTO de l'API pour briser le lien avec la base de données

```
public class PeriodeDto extends BaseEntityDto {
    private Long id;
    private Date debut;
    private Date fin;
    private String libelle;

    private List<JourPeriodeDto> jours;
}
```

```
public class ReservationJourPeriodeDto extends BaseEntityDto {
    private Long id;
    private JoueuseDto joueuse;
    private ChambreDto chambre;
    private JourPeriodeDto jour;
    private TypeChambreDto lunch;
    private TypeChambreDto diner;
    private NuiteeDto nuitee;

    private Boolean hasReserved = false;
}
```

Les classes de l'application Web prennent la même forme que l'application de la mission 4 (*partie 2.5.3*).

Dans le développement d'API et d'application Web utilisant des API, il est commun d'utiliser cette même structure qu'est le modèle CRUD (Create, Read, Update, Delete).

Développement d'application web

2.6. IHM (interfaces homme-machine), Maquettage

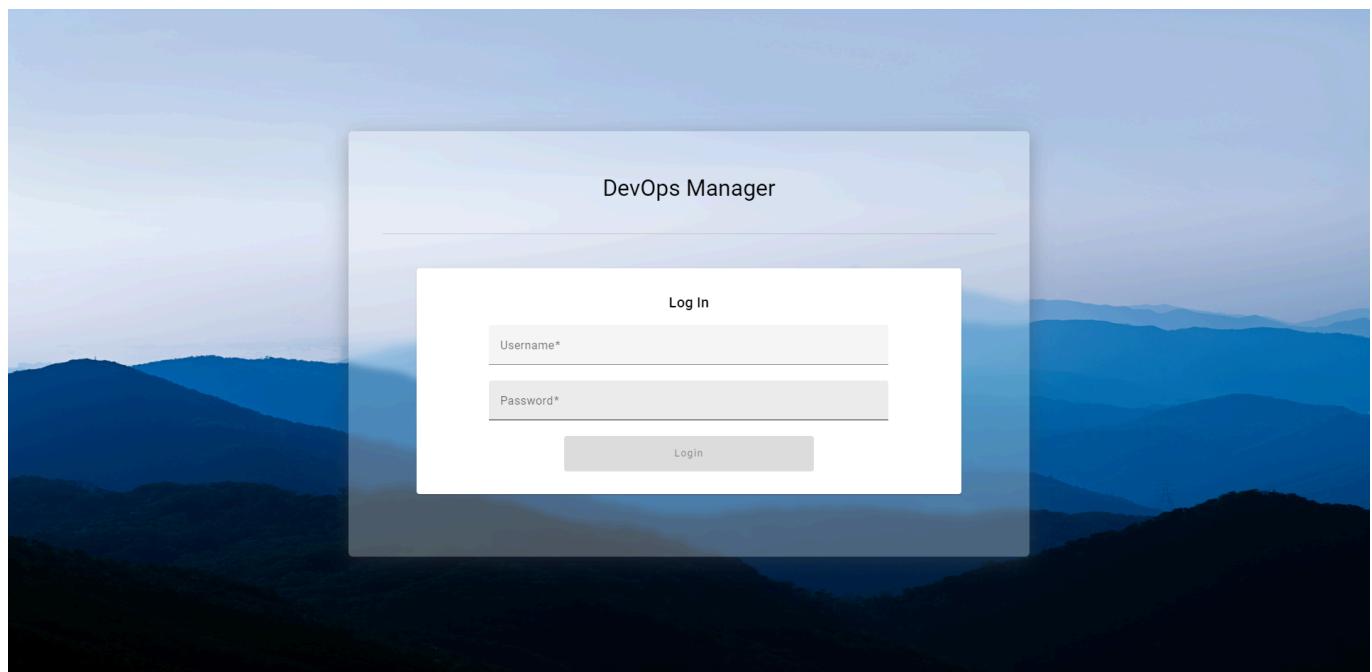
2.6.1. Mission 1: Développement d'une API lié à une base de données

Interface de l'application Postman permettant l'exécution des requêtes HTTP de l'API :

The screenshot shows the Postman application interface. On the left, there's a sidebar titled 'History' listing various API requests made on March 4. In the main central area, a 'GET' request is selected with the URL 'http://localhost:9001/jeux'. The 'Headers' tab shows '(6)' headers listed. Below the URL input field, there's a table for 'Query Params' with one row containing 'Key' and 'Value'. At the bottom of the main panel, there's a small illustration of an astronaut launching a rocket and the text 'Click Send to get a response'.

2.6.2. Mission 2 : Ajout de fonctionnalités sur une application Web existante

Interface de connexion (login) à l'application web :



2.6.3. Mission 3 : Récupération d'informations présentent sur un site (Scraping)

Interface web du formulaire pour le scraping :

The screenshot shows a white rectangular form titled "Notairinfo" centered over a background of stylized mountains at sunset. The form contains three input fields: "URL du site ciblé:" and "Nombre total de page (optionnel):" each with a single-line input box, and "Terme générique dans l'URL cible:" with a multi-line input box below them. A large blue button labeled "Récupérer les informations" is positioned at the bottom of the form.

Interface web du tableau affichant les informations récupérées :

The screenshot shows a table titled "Informations récupérées" with a "Télécharger le fichier CSV" button above it. The table has a header row with columns: "Lien Notaire.fr", "Mail", "Téléphone", "Nom Prénom", "Rue", "Code postal", "Commune", and "Site". The body of the table is currently empty, showing only the column headers.

2.6.4. Mission 4 : Développement d'une application Web liée à une API (continuité de la mission 1)

Interface web de la page d'accueil :

The screenshot shows the homepage of a website called "GamesDir". The header is blue with the site name. On the left, there's a sidebar with links: "Accueil", "Ajouter un jeu", and "Modifier des données". The main area displays a grid of nine game cards against a blue background with a mountain silhouette. Each card contains a game title, a short description, and the publisher's name.

Game Title	Description	Publisher
Assassin's Creed Mirage	Assassin's Creed Mirage, un jeu d'action-aventure, plonge les joueurs dans un Moyen-Orient vibrant d...	Ubisoft
Battlefield	Battlefield, série de jeux de tir en vue subjective d'EA et DICE, plonge les joueurs dans des combat...	ELECTRONIC ARTS
gfvdf	dfv...	Ubisoft
Hogwarts Legacy : L'Héritage de Poudlard	Hogwarts Legacy : L'Héritage de Poudlard est un RPG d'action-aventure immersif en monde ouvert qui s...	WARNER BROS INTERACTIVE
Mario Kart 8 Deluxe	Mario Kart™ 8 Deluxe, c'est la course délivrante dans un monde coloré! Choisissez parmi 42 personnages...	Nintendo
Minecraft	Minecraft, un jeu de construction emblématique, invite les joueurs à placer des blocs et à se lancer...	Mojang
Overwatch	Overwatch, jeu d'action en équipe, propose des affrontements palpitants avec une liste de héros en c...	ACTIVISION-BLIZZARD
Super Mario Odyssey	Explorez des endroits loin du royaume Champignon avec Mario et son nouvel allié, Cappy, dans une imm...	Nintendo
The Legend of Zelda: Breath of the Wild	The Legend of Zelda: Breath of the Wild est un jeu d'action-aventure où Link se réveille d'un sommeil...	Nintendo

Items per page: 10 | 1 – 9 of 9 | < >

Interface web des informations d'un jeu :

This screenshot shows the detailed information page for "Assassin's Creed Mirage". At the top, there's a pencil icon and the game's title. Below it, a button says "Passer en mode édition". The main text describes the game as an action-adventure set in a vibrant Middle Eastern conflict zone where players take on the role of Basim, a rebel assassin. It highlights the transformation narrative, immersive city exploration, contracts, and missions. Below the text, it lists the editor as Ubisoft and the category as Action. A table provides platform details, release dates, and prices.

Plateformes	Date de sortie	Tarif
PC (Windows)	03/03/2025	31.00 €
Xbox (Xbox Series X/S)	17/07/2024	16.00 €

Interface web d'ajout d'un jeu :

This screenshot shows a form titled "Jeu à ajouter" (Add Game). It consists of four input fields: "Nom du jeu*" (Game name), "Description du jeu*" (Game description), "Editeur*" (Publisher), and "Catégories*" (Categories). Each field has a placeholder text and a dropdown arrow for selection.

2.6.5. Mission 5 : Évolution d'une application Web existante

Ancienne interface web de gestion des évènements (période) et de l'assignation des chambres :

The screenshot shows a web application for managing bookings. At the top, there's a navigation bar with a logo for 'CBBS CHAMPS BASKET' and a link to 'Mon profil'. On the left, a sidebar lists categories: RÉSERVATIONS, CHAMBRES, JOUEUSES, NUITÉES, RÉGIMES, TYPES DE CHAMBRES, and TYPES DE REPAS. The main content area is titled 'Gestion des hébergements' and 'Réservations'. It shows a search form for a period from '22/03/2024' to '26/03/2024' with a label 'Tournois basket 2024'. Below this is a booking summary for three days:

Date	Nom	Chambre	Nuitée	Déjeuner	Dîner
22/03	Jade A	1	Tout Inclus	américain	italien
23/03	Louise B	1	Tout Inclus	américain	viennois
24/03	Emma C				

Buttons for 'Annuler' (Cancel), 'Valider' (Validate), and 'Suivant' (Next) are visible.

Interface web de gestion des chambres :

The screenshot shows a web application for managing rooms. On the left, a sidebar has a 'Chambres' button. The main area has a 'Créer une chambre' button. To the right, a modal window is open for 'ÉDITION DES CHAMBRES'. It contains fields for 'Numéro' (set to 4), 'Capacité' (set to 3), 'Type de chambre' (set to 'Chambre double'), and a 'Status' toggle switch labeled 'Active'. Buttons for 'Annuler' (Cancel) and 'Valider' (Validate) are at the bottom.

Numéro	Capacité
1	3
3	1
2	2

2.7. Conduite de projet : décomposition en tâches, structure d'équipe, planning (Gantt), durée

Première semaine :

- Visite des locaux
- Prise en main de la technologie Spring Boot :
 - Création d'une base de données MariaDB
 - Conception d'une API liée à une base de données MariaDB
- Début de la prise en main du framework Angular

Deuxième semaine :

- Fin de la prise en main des bases du framework Angular via les tutoriels officiels
- Observation et aide au développement de fonctionnalités sur une application web Angular réalisée par l'autre stagiaire présent
- Réalisation d'un projet de web scraping
- Début du développement d'un projet Angular, à titre d'entraînement, associé à l'API réalisée au début du stage

Troisième semaine :

- Création d'une application Web avec Angular lié à l'API réalisée en début de stage
- Reprise d'une application Web existante

Quatrième semaine :

- Étude du code d'une application Web (Mission 5)
- Développement et structuration des formulaires de réservations
- Réorganisation du code de l'API

Cinquième semaine :

- Redéveloppement du fonctionnement de la création des formulaires de période et de réservation
- Tentative de génération de PDF résumant les réservations d'une période
- Démonstration de l'application à un membre de la gestion du CBBS

Sixième semaine :

- Ajout de fonctionnalité pour l'envoi de mail lors d'une modification des informations d'une période et/ou de ses réservations
- Déploiement de l'application

3. Développement

3.1. Réalisation des interfaces et programmes conformes aux spécifications fonctionnelles attendues

3.1.1. Mission 1: Développement d'une API lié à une base de données

Récupération de données via l'API :

GET | <http://localhost:9000/editeurs>

Retour de la requête :

```
{
  "id": 1,
  "nom": "Ubisoft",
  "jeux": [
    {
      "id": 5,
      "nom": "Assassin's Creed Mirage",
      "description": "Assassin's Creed Mirage, un jeu d'action-aventure, plonge les joueurs dans un Moyen-Orient vibrant de conflits entre Assassins et Templiers.",
      "éditeur": { "id": 1, "nom": "Ubisoft" },
      "idCategorie": [ 1 ],
      "jeuPlateformes": [
        { "idPlateforme": 1, "dateSortie": "2025-03-02T23:00:00.000+00:00", "tarif": 31.0 },
        { "idPlateforme": 10, "dateSortie": "2024-07-16T22: 00: 00.000+00: 00", "tarif": 16.0 }
      ]
    }
  ],
  [
    {
      "id": 2,
      "nom": "Mojang",
      "jeux": [
        {
          "id": 2,
          "nom": "Minecraft",
        }
      ]
    }
  ]
}
```

Ajout de donnée via l'API :

POST | <http://localhost:9000/editeurs/editeur>

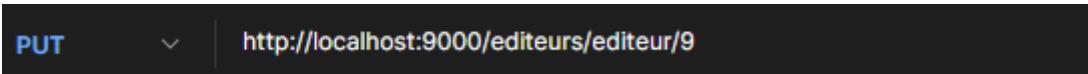
Valeur en entrée :

```
{"nom": "ROCKSTAR NORTH"}
```

Retour de la requête :

```
{
  "id": 9,
  "nom": "ROCKSTAR NORTH",
  "jeux": []
}
```

Modification de donnée via l'API :



Valeur en entrée :

```

[{"id": 9, "nom": "Rockstar North", "jeux": []}

```

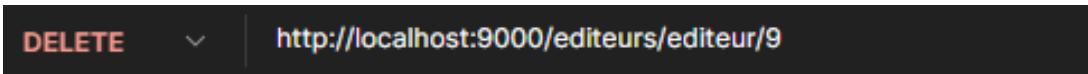
Retour de la requête :

```

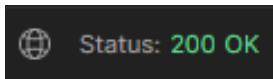
[{"id": 9, "nom": "Rockstar North", "jeux": []}

```

Suppression de donnée via l'API :



Indication que la requête a fonctionnée :



3.1.2. Mission 2 : Ajout de fonctionnalités sur une application Web existante

Affichage du popup d'inactivité lorsqu'il reste 11 secondes avant déconnexion :

A screenshot of a web application interface titled "DevOps Manager". The left sidebar lists navigation items: Dashboard, Applications, Criticité, Environnements, Incidents, Règles de vérification, and Utilisateurs. The "Applications" section is active, showing a table with columns: ID application, Nom, Code, Description, and Options. A modal dialog is overlaid on the page, asking "Etes-vous inactif ?" (Are you inactive?) and displaying "Temps avant déconnexion : 11 s" (Time before logout: 11 s).

3.1.3. Mission 3 : Récupération d'informations présentent sur un site (Scraping)

Fichier CSV contenant le lien des pages notaire :

```
linksPage.csv x
linksPage.csv > data
1 https://www.notaires.fr/fr/notaire/benedicte-eckly-gardez?search_params=eyJkaXJlY3RvcnlfbG9jYXRpb25zIjo1MTEiLCJwYWdlIjoiMCJ9
2 https://www.notaires.fr/fr/notaire/valentin-vovor?search_params=eyJkaXJlY3RvcnlfbG9jYXRpb25zIjo1MTEiLCJwYWdlIjoiMCJ9
3 https://www.notaires.fr/fr/notaire/marianne-prezioso?search_params=eyJkaXJlY3RvcnlfbG9jYXRpb25zIjo1MTEiLCJwYWdlIjoiMCJ9
4 https://www.notaires.fr/fr/notaire/maxime-venditti?search_params=eyJkaXJlY3RvcnlfbG9jYXRpb25zIjo1MTEiLCJwYWdlIjoiMCJ9
5 https://www.notaires.fr/fr/notaire/cedric-chevaleyre?search_params=eyJkaXJlY3RvcnlfbG9jYXRpb25zIjo1MTEiLCJwYWdlIjoiMCJ9
6 https://www.notaires.fr/fr/notaire/dominique-ballarin?search_params=eyJkaXJlY3RvcnlfbG9jYXRpb25zIjo1MTEiLCJwYWdlIjoiMCJ9
7 https://www.notaires.fr/fr/notaire/claudie-guerin?search_params=eyJkaXJlY3RvcnlfbG9jYXRpb25zIjo1MTEiLCJwYWdlIjoiMCJ9
8 https://www.notaires.fr/fr/notaire/audrey-chassouiller?search_params=eyJkaXJlY3RvcnlfbG9jYXRpb25zIjo1MTEiLCJwYWdlIjoiMCJ9
9 https://www.notaires.fr/fr/notaire/isabelle-vincent-martin?search_params=eyJkaXJlY3RvcnlfbG9jYXRpb25zIjo1MTEiLCJwYWdlIjoiMCJ9
```

Fichier CSV contenant les informations de notaire :

```
listInformationsNotaires.csv x
listInformationsNotaires.csv > data
1 "https://www.notaires.fr/fr/notaire/benedicte-eckly-gardez?search_params=eyJkaXJlY3RvcnlfbG9jYXRpb25zIjo1MTEiLCJwYWdlIjoiMCJ9"
2 ",bazaille.associes@notaires.fr,0478730111,Maître Béatrice ECKLY-GARDEZ,23 RUE DENFERT ROCHEREAU,69700,GIVORS,https://bazaille-associes.com
3 "https://www.notaires.fr/fr/notaire/valentin-vovor?search_params=eyJkaXJlY3RvcnlfbG9jYXRpb25zIjo1MTEiLCJwYWdlIjoiMCJ9"
4 ",valentin.vovor@notaires.fr,0487915500,Maître Valentin VOVOR,302 Cours Lafayette,69003,LYON,
5 "https://www.notaires.fr/fr/notaire/marianne-prezioso?search_params=eyJkaXJlY3RvcnlfbG9jYXRpb25zIjo1MTEiLCJwYWdlIjoiMCJ9
6 ",officedeurope@notaires.fr,0478958140,Maître Marianne PRÉZIOSO,62 RUE DE BONNEAU,69003,LYON,
7 "https://www.notaires.fr/fr/notaire/maxime-venditti?search_params=eyJkaXJlY3RvcnlfbG9jYXRpb25zIjo1MTEiLCJwYWdlIjoiMCJ9
8 ",scp.bonnefondetassociés@notaires.fr,0478420081,Maître Maxime VENDITTI,45 RUE DE LA REPUBLIQUE,69002,LYON,https://bonnefond-venditti.fr
9 "https://www.notaires.fr/fr/notaire/cedric-chevaleyre?search_params=eyJkaXJlY3RvcnlfbG9jYXRpb25zIjo1MTEiLCJwYWdlIjoiMCJ9
10 ",notaires.ecully@notaires.fr,0478339260,Maître Cédric CHEVALEYRE,4 ALLEES DES TULLISTES,69130,CULLY,
11 "https://www.notaires.fr/fr/notaire/dominique-ballarin?search_params=eyJkaXJlY3RvcnlfbG9jYXRpb25zIjo1MTEiLCJwYWdlIjoiMCJ9
12 ",office69041.vernaison@notaires.fr,0478461033,Maître Dominique BALLARIN,1 SQUARE CARDINAL,69390,VERNAISON,https://benardetassociés.notaires.fr
13 "https://www.notaires.fr/fr/notaire/claudie-guerin?search_params=eyJkaXJlY3RvcnlfbG9jYXRpb25zIjo1MTEiLCJwYWdlIjoiMCJ9
14 ",claudie.guerin@notaires.fr,0472196342,Maître Claudie GUERIN,"14, B Chemin du Professeur Dépret",69160,TASSIN-LA-DEMI-LUNE,https://claudieguerin.com
15 "https://www.notaires.fr/fr/notaire/audrey-chassouiller?search_params=eyJkaXJlY3RvcnlfbG9jYXRpb25zIjo1MTEiLCJwYWdlIjoiMCJ9
16 ",alcaix-et-associes-lexel@notaires.fr,0472745340,Maître Audrey CHASSOUILLER,91 COURS LAFAYETTE,69006,LYON,
17 "https://www.notaires.fr/fr/notaire/isabelle-vincent-martin?search_params=eyJkaXJlY3RvcnlfbG9jYXRpb25zIjo1MTEiLCJwYWdlIjoiMCJ9
18 ",office.vincentmartin@notaires.fr,0474030522,Maître Isabelle VINCENT-MARTIN,10 impasse Chavane,69870,LAMURE-SUR-AZERGUES,http://vincentmartinnotaire.com
19 "https://www.notaires.fr/fr/notaire/caroline-salanson-bottazzi?search_params=eyJkaXJlY3RvcnlfbG9jYXRpb25zIjo1MTEiLCJwYWdlIjoiMSJ9
20 ",scp.leufflendeforme.69017@notaires.fr,0472847200,Maître Caroline SALANSON-BOTTAZZI,144 Avenue Maréchal de Saxe,69003,LYON,https://ofc-salanson-bottazzi.com
21 "https://www.notaires.fr/fr/notaire/remy-samson?search_params=eyJkaXJlY3RvcnlfbG9jYXRpb25zIjo1MTEiLCJwYWdlIjoiMSJ9
22 ",officenotarialdelapasserelle@notaires.fr,0478427830,Maître Rémy SAMSON,36 QUAI SAINT ANTOINE,69002,LYON,https://samson-david-gasiot.fr
23 "https://www.notaires.fr/fr/notaire/frederic-lombardo?search_params=eyJkaXJlY3RvcnlfbG9jYXRpb25zIjo1MTEiLCJwYWdlIjoiMSJ9
24 ",office-notarial.genas@notaires.fr,0478901003,Maître Frédéric LOMBARDO,38 RUE DE LA REPUBLIQUE,69740,GENAS,
25 "https://www.notaires.fr/fr/notaire/severine-girardon?search_params=eyJkaXJlY3RvcnlfbG9jYXRpb25zIjo1MTEiLCJwYWdlIjoiMSJ9
```

3.1.4. Mission 4 : Développement d'une application Web liée à une API (continuité de la mission 1)

Tentative d'ajout d'un jeu dans la base de données :

Jeu à ajouter

Nom du jeu*

super mario bros

Description du jeu*

Super Mario Bros. is a platform game. In the game, Mario must race through the Mushroom Kingdom and save Princess Toadstool (later Princess Peach) from Bowser. Mario jumps, runs, and walks across each level. The worlds are full of enemies, platforms, and open holes.

Editeur*

Nintendo

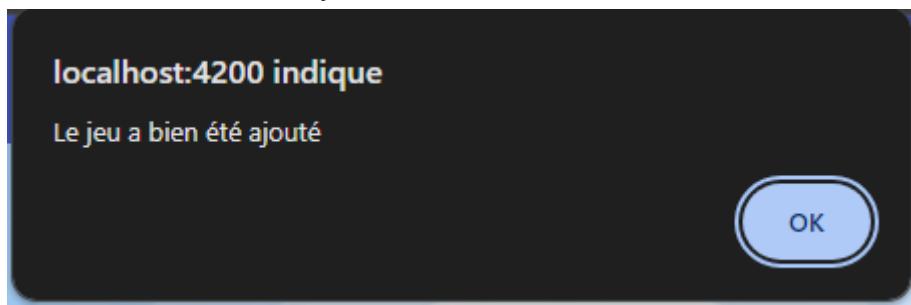
Catégories sélectionnées : Action, RPG,

Catégories*

(2 éléments)

Ajouter

Message signalant la réussite de l'ajout :



Tentative de modification de l'éditeur d'un jeu :

Assassin's Creed Mirage

Assassin's Creed Mirage, un jeu d'action-aventure, plongez-vous dans l'univers de l'Assassin, et découvrez un récit de transformation intense.

Editeur : ACTIVISION-BLIZZARD INC.

Enregistrer le nouvel éditeur

Catégorie :

3.1.5. Mission 5 : Évolution d'une application Web existante

Nouvelle interface pour la gestion des périodes :

Formulaire d'ajout de période :

Interface de gestion d'une période et de ces réservations :

3.2. Dossier de programmation, codes sources documentés et commentés

3.2.1. Mission 1: Développement d'une API lié à une base de donnée

Méthode "GET(jeu/{id})" de la classe "JeuController.java" :

```
/**
 * Read - Get an entity
 *
 * @return - An ReadEntityDTO Object
 */
@GetMapping("/jeu/{id}")
public ReadJeuDTO getJeu(@PathVariable("id") final Long id) throws Exception {
    String query = "id==" + id;
    // Get an entity page with one entity with the id in the query
    Page<ReadJeuDTO> jeuPage = jeuService.searchJeux(query, 0, 1, "");
    // If the page is empty return null
    if (jeuPage.isEmpty()){
        return null;
    }
    // Return the first entity of the page if present
    return jeuPage.getContent().get(0);
}
```

Méthode "searchJeux()" de la classe "JeuService.java" :

```
/**
 * Read - Get one or many entities
 */
public Page<ReadJeuDTO> searchJeux(String query, Integer page, Integer size, String orderBy) throws Exception {

    Page<ReadJeuDTO> readJeuDTOPage = genericService.searchEntity(Jeu.class,
        ReadJeuDTO.class, query, page, size, orderBy);

    for (ReadJeuDTO readJeuDTO : readJeuDTOPage) {
        getFullReadEntityDTO(readJeuDTO);
    }
    return readJeuDTOPage;
}
```

Méthode "searchEntity()" de la classe "GenericService.java":

```
/**  
 * Search for entities in the database and convert them to  
 * @param entity      Class of the entity in database  
 * @param readEntityDTO Class of entity to return  
 * @param page        Entity page number  
 * @param size         Number of entities per page  
 * @param orderBy     ex : id=ASC  
 * @return An Entity Page containing the searched entities  
 */  
public Page<READENTITYDTO> searchEntity(Class<ENTITY> entityClass,  
Class<READENTITYDTO> readEntityDTOClass, String query, Integer page, Integer size,  
String orderBy) throws Exception {  
    // Search for entities in the database with méthode piqlV2Search imported  
    // from PIQLV2Searchable (dependency of Partner Informatique)  
    Page<ENTITY> entityBDD = piqlV2Search(entityClass, query, page, size,  
    orderBy);  
    // If no entity found return null  
    if (entityBDD == null) {  
        throw new Exception("Aucune entité trouvée");  
    }  
    // Convert the entity to DTO and return it  
    return entityBDD.map(item -> mapper.map(item, readEntityDTOClass));  
}
```

3.2.2. Mission 2 : Ajout de fonctionnalités sur une application Web existante

Méthode permettant de régénérer le token à chaque fois que celui-ci expire :

```
nextLogin(seconds: any) {
  console.log(seconds);

  let stop = setInterval(() => {
    if (!['/login', '/'].includes(window.location.pathname)) {
      this.loginService.getNewToken().subscribe((res) => {
        this.loginService.lastTokenDate = new Date();
        localStorage.setItem('token', JSON.stringify(res));
      });
    }
  }, seconds * 1000);
}
```

→ Méthode exécutée après la connexion

→ Exécution de la régénération du token au moment de son expiration : la variable "seconds" correspond à son temps de validité

→ Vérification que l'utilisateur n'est pas sur la page de login

→ Après régénération du token : enregistrement de la date de création du nouveau token et stockage de celui-ci en local.

Méthode permettant de rediriger l'utilisateur sur la page d'accueil si celui-ci est déjà connecté :

```
public isAlreadyAuthenticated(){

  let token = localStorage.getItem('token');

  if(token === null || token == ""){
    return null;
  }else{
    this.getNewToken().subscribe((res) => {
      this.router.navigateByUrl('/accueil')
    }, (error) => {
      localStorage.clear();
    });
  }
}
```

→ Récupère le contenu de la variable "token" stocké en local

→ Vérifie le contenu de la variable obtenu : redirection sur la page d'accueil en cas d'invalidité

→ Régénération du token via le refresh_token récupéré dans la variable local

→ En cas de réussite de la régénération du token, redirection vers la page d'accueil

→ En cas d'échec (erreur de régénération), redirection vers la page de login

Méthode permettant de modifier le temps restant avant déconnexion :

```
startWatching() {
  this.idleSubscription = interval(this.idleCheckInterval * 1000).pipe(
    throttle(() => interval(1000))
  ).subscribe(()=>{
    const now = new Date();
    let timeLeft = (this.timeout * 1000) - (new Date().getTime() - this.lastActivity?.getTime()!);

    if (timeLeft >= -2000) {
      this.timeLefting.next(timeLeft);
    }
  })
}
```

→ Méthode exécutée lorsque l'utilisateur ne réalise plus d'action sur la page.

3.2.3. Mission 4 : Développement d'une application Web liée à une API (continuité de la mission 1)

Récupération des informations d'un jeu :

```
constructor(@Inject(MAT_DIALOG_DATA) public game: any,
           private gameService: GameService,
           private editeurService: EditeurService,
           private router: Router) {
  this.gameInfos = game;
  this.editeurInfos = this.gameInfos.editeur;

  this.editeurService.getAllEditeurs().subscribe((data: any) => {
    this.listEditeurs = data.content;
  });

  this.dataSource = this.gameInfos.plateformes;
}
```

Récupération des informations de tous les éditeurs

```
// Récupération de la liste de tous les éditeurs
getAllEditeurs() {
  return this.httpClient.get(`${environment.apiUrl}editeurs`);
}

// Récupération des informations d'un éditeur en fonction de son id
getEditeurInfos(editeurId: number) {
  return this.httpClient.get(`${environment.apiUrl}editeurs/editeur/${editeurId}`);
}

// Modification des informations d'un éditeur en fonction de son id
update(editeurId: number, data: any) {
  return this.httpClient.put(`${environment.apiUrl}editeurs/editeur/${editeurId}`, data);
}
```

Développement d'application web

3.2.4. Mission 5 : Évolution d'une application Web existante

Méthode de l'API permettant la création d'une période dans la base de données:

```
@Override
public PeriodeDto create(PeriodeDto dto) throws Exception {
    Periode periode = mapper.dtoToEntity(dto);

    // Création de la période dans la base de données
    periode = genericService.create(periode);

    // Vérification de la validité de la période
    if(periode.getFin().before(periode.getDebut())) {
        throw new BadRequestException(message:"La date de fin ne peut pas être antérieure à la date de début");
    }

    // Instanciation des jours de la période
    List<JourPeriode> dates = JourPeriodeMapper.INSTANCE.listDtosToListEntities(utilities.generateListDateOfPeriode(dto));

    // Création des jours de la période dans la base de données
    jourPeriodeRepository.saveAll(dates);

    // Crédit des réservations pour chaque jour de la période dans la base de données
    reservationJourPeriodeRepository.saveAll(utilities.generateListReservationOfPeriode(dates, joueuseService.getAll()));

    return mapper.entityToDto(periode);
}
```

Classe contenant des méthodes utiles :

→ Génération d'une liste de date pour une période

```
public static List<JourPeriodeDto> generateListDateOfPeriode(PeriodeDto periode){
    // Ajout dans une liste de tous les jours de la période
    List<JourPeriodeDto> days = new ArrayList<>();
    Calendar calendar = Calendar.getInstance();

    Long offset = (periode.getFin().getTime() - periode.getDebut().getTime()) / (1000 * 3600 * 24);
    calendar.setTime(periode.getDebut());

    JourPeriodeDto jour = new JourPeriodeDto(calendar.getTime(), periode);
    days.add(jour);

    for(int i = 0; i <= offset; i++) {
        calendar.add(Calendar.DAY_OF_MONTH, amount:1);
        JourPeriodeDto jourP = new JourPeriodeDto(calendar.getTime(), periode);
        days.add(jourP);
    }

    return days;
}
```

→ Génération d'une liste de réservation pour chaque jour d'une période

```
public static List<ReservationJourPeriode> generateListReservationOfPeriode(List<JourPeriode> joursPeriode, List<JoueuseDto> joueuses){
    List<ReservationJourPeriode> reservations = new ArrayList<>();

    for (JourPeriode jour : joursPeriode) {
        for (JoueuseDto joueuse : joueuses) {
            ReservationJourPeriode res = new ReservationJourPeriode();
            res.setId(id:null);
            res.setJour(jour);
            res.setJoueuse(JoueuseMapper.INSTANCE.dtoToEntity(joueuse));
            res.setChambre(JoueuseMapper.INSTANCE.dtoToEntity(joueuse).getChambreParDefaut());
            res.setDiner(diner:null);
            res.setNuitee(nuitee:null);
            res.setLunch(lunch:null);
            res.setHasReserved(hasReserved:false);
            res.setIsDeleted(isDeleted:false);

            reservations.add(res);
        }
    }

    return reservations;
}
```

4. Exploitation, Mise en production

4.1. Tests (unitaires, fonctionnels, intégration)

4.1.1. Mission 1: Développement d'une API lié à une base de données

Tests de requête ayant été réalisé pour s'assurer du bon fonctionnement de l'API :

Tests requêtes [GET]	Tests requêtes [POST]
<p>Obtenir toutes les entités d'une table :</p> <ul style="list-style-type: none"> → jeux → editeurs → categories → jeux-categories → plateformes → jeux-plateformes <p>Exemples de requête pour obtenir une entité via son ID :</p> <ul style="list-style-type: none"> → jeux/jeu/0 → jeux/jeu/2 → editeurs/editeur/0 → editeurs/editeur/2 → jeux-plateformes/jeu-plateforme/0-0 → jeux-plateformes/jeu-plateforme/2-2 → jeux-plateformes/jeu/0 → jeux-plateformes/jeu/2 → jeux-plateformes/plateforme/0 → jeux-plateformes/plateforme/2 <p>Exemples de requête pour obtenir une entité via la query :</p> <ul style="list-style-type: none"> → jeux/search?query=id=in=(0) → jeux/search?query=id=in=(2, 3, 4) → editeurs/search?query=id=in=(0) → editeurs/search?query=id=in=(2, 3, 4) → jeux-categories/search?query=idJeu==0;idCategorie==0 → jeux-categories/search?query=idCategorie=in=(2, 3, 4) → jeux-categories/search?query=idJeu=in=(2, 3, 4) 	<p>Exemples de requête pour ajouter une entité dans la bdd :</p> <ul style="list-style-type: none"> → jeux/jeu {"nom": "test", "description": "desc", "editeur": {"id": 1}} → editeurs/editeur {"nom": "test"} → categories/categories {"libelle": "test", "codeCategorie": "11111"} → jeux-categories/jeu-categories {"idJeu": 2, "idCategorie": 2} → plateformes/plateforme {"libelle": "test", "codePlateforme": "11111"} → jeux-plateformes/jeu-plateforme {"idJeu": 4, "idPlateforme": 3, "dateSortie": "2023-01-01", "tarif": 5}
Tests requêtes [PUT]	Tests requêtes [DELETE]
<p>Exemples de requête pour modifier une entité via son ID :</p> <ul style="list-style-type: none"> → jeux/jeu/12 {"nom": "test2", "description": "desc2", "editeur": {"id": 2}} → editeurs/editeur/6 {"nom": "test2"} → categories/categories/6 {"libelle": "test2", "codeCategorie": "22222"} → plateformes/plateforme/6 {"libelle": "test2", "codePlateforme": "22222"} → jeux-plateformes/jeu-plateforme/2-1 {"dateSortie": "2023-01-01", "tarif": 10} 	<p>Exemples de requête pour supprimer une entité via son ID :</p> <ul style="list-style-type: none"> → jeux/jeu/12 → editeurs/editeur/6 → categories/categories/6 <p>Supprimer l'entitée d'idJeu=2 et d'idCategorie=3</p> <ul style="list-style-type: none"> → jeux-categories/jeu-categories/2-3 <p>Supprimer la relation jeu-categories de tous les jeux d'idJeu=2</p> <ul style="list-style-type: none"> → jeux-categories/jeu/2 <p>Supprimer la relation jeu-categories de toutes les categories d'idCategorie=2</p> <ul style="list-style-type: none"> → jeux-categories/categories/2

Test de la requête de suppression pour une entité existante :

The screenshot shows a Postman request configuration. The method is set to **DELETE**, and the URL is <http://localhost:9000/editeurs/editeur/9>. The status bar at the bottom right indicates **Status: 200 OK**.

Code 200 OK donc requête réussie.

Test de la requête de suppression pour une entité inexisteante :

The screenshot shows a Postman request configuration. The method is set to **DELETE**, and the URL is <http://localhost:9000/editeurs/editeur/20>. The status bar at the bottom right indicates **Status: 400 BAD REQUEST**. The response body is displayed in JSON format:

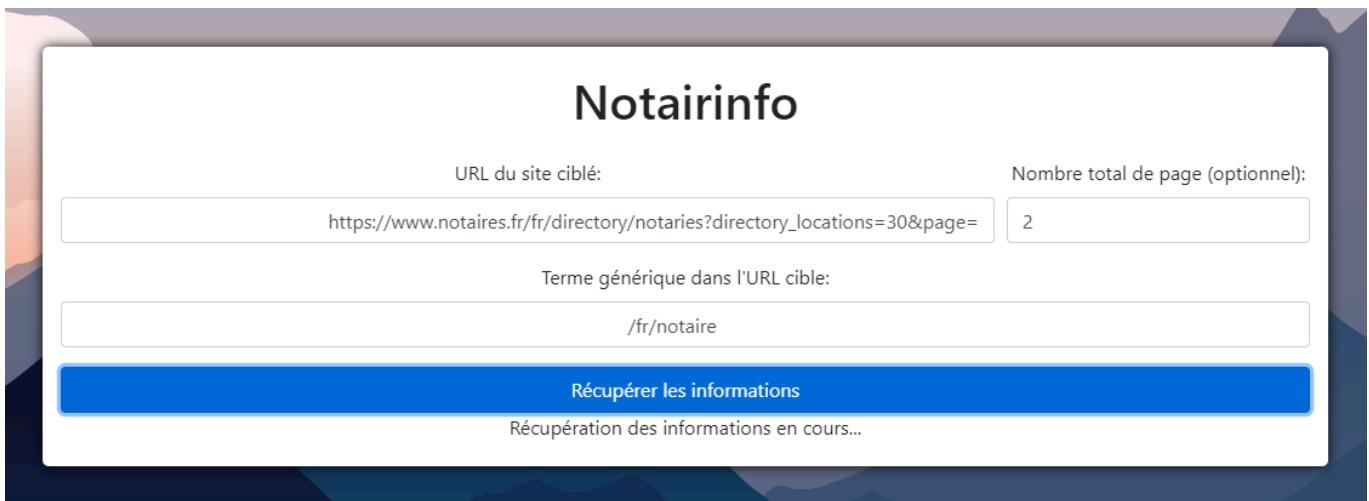
```
1  {
2      "message": "Entity non trouvée",
3      "timestamp": "Fri Mar 08 17:05:20 CET 2024",
4      "status": "BAD_REQUEST"
5 }
```

Code 400 BAD REQUEST donc requête échouée.

4.1.2. Mission 3 : Récupération d'informations présentent sur un site (Scraping)

Tests de récupération des informations des notaires de Saône-et-Loire des 2 premières du site via le lien

[“https://www.notaires.fr/fr/directory/notaries?directory_locations=30&page=”](https://www.notaires.fr/fr/directory/notaries?directory_locations=30&page=)



Informations sur l'état du scraping (18 notaires à récupérer) :

```
Récupération des informations de la page : https://www.notaires.fr/fr/directory/notaries?directory_locations=30&page=1

-----
[=-----] 1/18
[==-----] 2/18
[====-----] 3/18
[=====-----] 4/18
[=====-----] 5/18
[=====-----] 6/18
[=====-----] 7/18
[=====-----] 8/18
[=====-----] 9/18
[=====-----] 10/18
[=====-----] 11/18
[=====-----] 12/18
[=====-----] 13/18
[=====-----] 14/18
[=====-----] 15/18
[=====-----] 16/18
[=====-----] 17/18
[=====-----] 18/18
127.0.0.1 - - [16/Mar/2024 18:47:43] "POST /scrape HTTP/1.1" 302 -
127.0.0.1 - - [16/Mar/2024 18:47:43] "GET /results HTTP/1.1" 200 -
127.0.0.1 - - [16/Mar/2024 18:47:43] "GET /static/css/styles.css HTTP/1.1" 304 -
127.0.0.1 - - [16/Mar/2024 18:47:43] "GET /static/images/bg.jpg HTTP/1.1" 304 -
```

Affichage des informations récupérées :

Informations récupérées

[Télécharger le fichier CSV](#)

Mail	Téléphone	Nom Prénom	Rue	Code postal	Commune	Site
didier.mathy@notaires.fr	0385604040	Maître Didier MATHY	15 route de Louhans	71580	SAGY	
officedethiard@notaires.fr	0385900760	Maître Marie-Elise CANOVA	1 RUE DE THIARD	71100	CHALON- SUR-SAÔNE	https://officedethiard-canova-jeannin-viellard.notaires.fr
laurence.vernet@notaires.fr	0385711100	Maître Laurence VERNET	102 ROUTE DE CHALON	71270	PIERRE-DE- BRESSE	https://vernet-baud.notaires.fr
valluche.louhans@notaires.fr	0385750264	Maître Guillaume VALLUCHE	15 rue d'Alsace	71500	LOUHANS	
office.nigaud@notaires.fr	0385770606	Maître Laurence NIGAUD	32 RUE JEAN JAURES	71200	LE CREUSOT	http://www.nigaud-lecreusot.notaires.fr

4.1.3. Mission 5 : Évolution d'une application Web existante

Tentative d'ajout d'une période :

Ajouter une nouvelle période

Date de début *

29/03/2024

Date de fin *

30/03/2024

Libellé *

Tournois 2024

Valider

Annuler

Réussite → Nouvelle période présente dans la liste de période :

+ Créer une période

Période du 29/03/2022 au 30/03/2022

Libelle : Tournois 2022



Période du 29/03/2023 au 30/03/2023

Libelle : Tournois 2023



Période du 29/03/2024 au 30/03/2024

Libelle : Tournois 2024



Réservations du 5 mars 2024 :

1 05/04 2 06/04

Coralie CHABRIER <input checked="" type="checkbox"/>	KANKOU COULIBALY <input checked="" type="checkbox"/>	KAYANA TRAYLOR <input type="checkbox"/>
Chambre * 1	Nuitée * Avec petit dej	
Déjeuner * Sur place	Diner * A emporter	
Johanna MUZET <input type="checkbox"/>		

Suivant **Valider**

Réservations du 6 mars 2024 :

1 05/04 2 06/04

Coralie CHABRIER <input type="checkbox"/>	KANKOU COULIBALY <input type="checkbox"/>	KAYANA TRAYLOR <input type="checkbox"/>
Johanna MUZET <input type="checkbox"/>		

Recopier jour précédent **Précédent** **Valider**

Tentative de copie des réservations du 5 mars sur le 6 mars :

1 05/04 2 06/04

Coralie CHABRIER <input checked="" type="checkbox"/>	KANKOU COULIBALY <input checked="" type="checkbox"/>	KAYANA TRAYLOR <input type="checkbox"/>
Chambre * 1	Nuitée * Avec petit dej	
Déjeuner * Sur place	Diner * A emporter	
Johanna MUZET <input type="checkbox"/>		

Recopier jour précédent **Précédent** **Valider**

Interface avant tentative de modification de la date de fin de période :

Période Du 05/04 au 06/04

Date de début * 05/04/2024 Date de fin * 06/04/2024

Libellé * Test

Valider **Annuler**

1 05/04 2 06/04

Après tentative de modification de la date de fin de période :

Période Du 05/04 au 13/04

Date de début * 05/04/2024 Date de fin * 13/04/2024

Libellé * Test

Valider **Annuler**

1 05/04 2 06/04 3 07/04 4 08/04 5 09/04 6 10/04 7 11/04 8 12/04 9 13/04

Réservations réalisées :

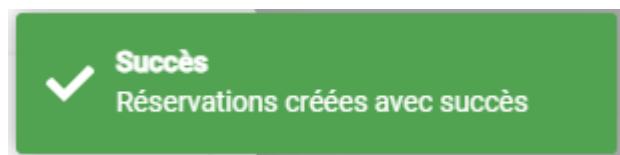
05/04 06/04

Recopier jour précédent **Précédent** **Valider**

Coralie CHABRIER	<input checked="" type="checkbox"/>	KANKOU COULIBALY	<input checked="" type="checkbox"/>	KAYANA TRAYLOR	<input checked="" type="checkbox"/>
Chambre *	1	Nuitée *	Avec petit dej	Chambre *	1
Déjeuner *	Sur place	Diner *	A emporter	Nuitée *	Sans petit dej
				Déjeuner *	Pas de repas
				Diner *	A emporter

Johanna MUZET

Tentative d'enregistrement des réservations :



Ajout d'une joueuse "test" :

ID	Nom	Prénoms	Chambre par défaut	Régime	Active	Actions
1	CHABRIER	Coralie	1	Pas de régime	OUI	⋮
2	COULIBALY	KANKOU	1	Halal	OUI	⋮
3	TRAYLOR	KAYANA	2	Végétarien	OUI	⋮
4	MUZET	Johanna	3	Pas de régime	OUI	⋮
5	test	test	1	Pas de régime	OUI	⋮

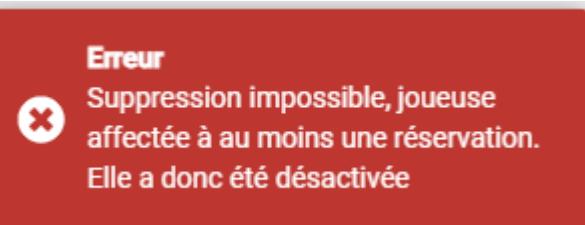
Nouvelle joueuse ajoutée à toutes les réservations de période :

05/04 06/04 07/04 08/04 09/04 10/04 11/04 12/04 13/04

Recopier jour précédent Précédent Suivant Valider

Coralie CHABRIER	KANKOU COULIBALY	KAYANA TRAYLOR
Johanna MUZET	test test	
Chambre * 1 Nutée * Avec petit dej Déjeuner * Sur place Diner * Pas de repas		

Tentative de suppression d'une joueuse :



Résultat : joueuse désactivée

ID	Nom	Prénoms	Chambre par défaut	Régime	Active	Actions
5	test	test	1	Pas de régime	NON	⋮

Résultat : la joueuse n'apparaît plus dans les réservations

Coralie CHABRIER	KANKOU COULIBALY	KAYANA TRAYLOR
Johanna MUZET		

4.2. Installation, déploiement, fonctionnement éventuel en double avec l'ancienne procédure.

4.2.1. Mission 5 : Évolution d'une application Web existante

Afin d'avoir un retour sur le travail réalisé, j'ai déployé une version démo de l'application avec un jeu d'essai pour chaque table de la base de données.

 dbtemp	→ Base de données temporaire contenant le jeu d'essai
 frontCBBS	→ Front : Application Web Angular compilé
 cbbsBackAPI.jar	→ Back : API compilé pour la communication Front/Base de données

Pour le déploiement du back :



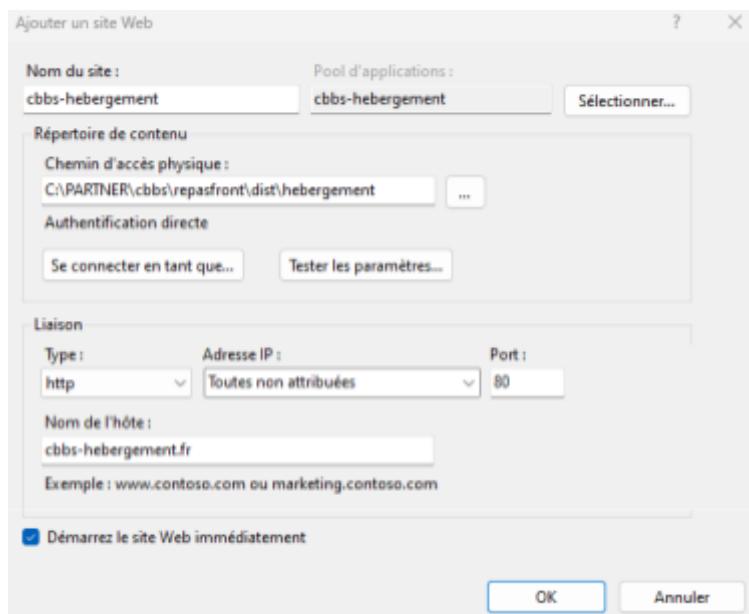
Utilisation de NSSM (*Non-Sucking Service Manager*) pour installer et lancer le back en tant que service Windows simplement et rapidement.

Pour le déploiement du front :



Utilisation de l'application IIS (Internet Information Services) de Microsoft pour héberger le site Web.

Configuration de l'ébergement du site :



Avec ces configurations, il est possible d'accéder localement au site depuis l'URL *cbbs-hebergement.fr*.

Développement d'application web

5. Bilan

Lors de mes six semaines de stage, j'ai pu prendre en main les différentes technologies utilisées dans l'entreprise. Mes compétences informatiques ont pu évoluer pour exploiter de nouvelles pratiques de développement tel que la généricité en informatique ou la structuration des fichiers dans un projet. De plus, j'ai appris à créer ma propre API REST reliée à une base de données MariaDB et j'ai appris à utiliser le framework Angular me permettant ainsi de réaliser des applications Web complètes. Mes connaissances en tant que développeur Full-Stack ont grandement évolué. Pour finir, j'ai appris à compiler et à déployer une application back et front. Je suis très content de réaliser mon stage dans l'entreprise Partner Informatique.