KARABENLI Reyhan BTS SIO

PROJET LA GOURMETISE

SOMMAIRE

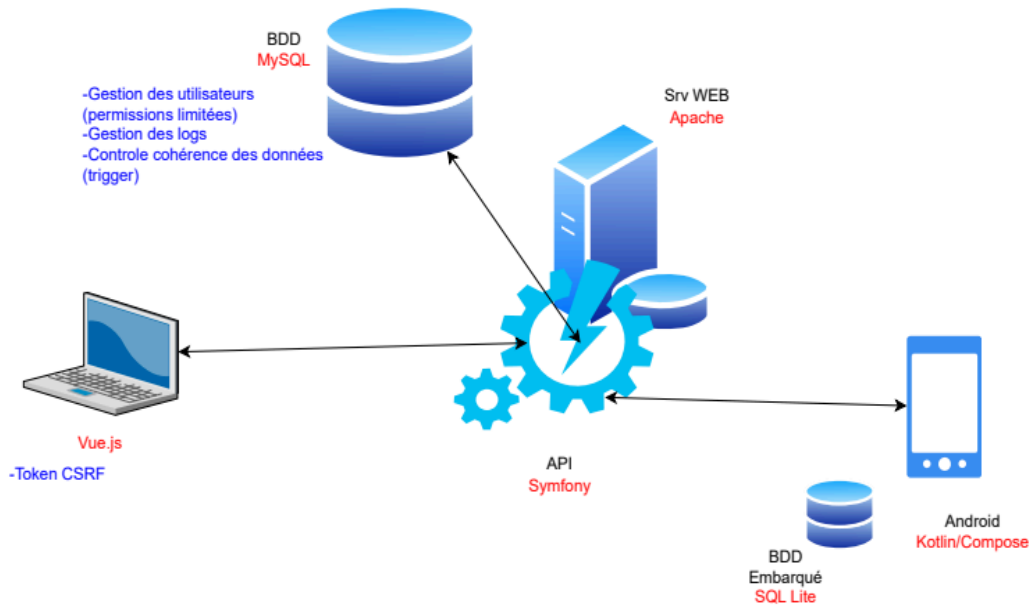
CONTEXTE

L'entreprise La Gourmetise, a été fondée en 2010 par un groupe de passionnés de la gastronomie et de la boulangerie. Depuis sa création, La Gourmetise s'est engagée à promouvoir l'excellence dans le domaine de la boulangerie et de la pâtisserie. Sa mission est de célébrer l'art de la boulangerie en mettant en lumière les artisans talentueux qui créent des produits exceptionnels.

Le concours de la meilleure boulangerie est une initiative visant à identifier, reconnaître et récompenser les boulangers et établissements de boulangerie qui excellent dans la création de produits de haute qualité. La Gourmetise souhaite créer une plateforme qui permettra aux boulangers participants de partager leur passion pour la boulangerie et aux amateurs de découvrir les meilleures boulangeries de la région.

Le concours est ouvert aux boulangers et établissements de boulangerie de la région. Pour participer, il suffit de s'inscrire au concours en remplissant le formulaire d'inscription en ligne. Dans un deuxième temps, les juges se rendent dans les boulangeries participant au concours et votent pour leurs boulangeries préférées à l'aide d'une application mobile préalablement installée sur leur téléphone. A la fin du concours seront désignées les 3 meilleures boulangeries.

SCHÉMA D 'ARCHITECTURE DE LA SOLUTION



DESCRIPTION DE L'ENVIRONNEMENT DE DÉVELOPPEMENT

L'API sera basé sur le framework Symfony (PHP 8.3.10) et l'application web sur le framework Vue.js (Javascript) en utilisant composition API. L'environnement de développement intégré pour l'API et l'application web est PhpStorm. Le modèle d'architecture logicielle est ici MVC (modèle-vue-contrôleur).

Du côté mobile, l'application est codée en Kotlin et utilise le framework d'interface utilisateur (UI) JetPack Compose d'Android. L'environnement de développement intégré est Android Studio.

La solution de gestion de versions du projet est Github.

DESCRIPTION DE L'ORGANISATION DU PROJET

La méthode de gestion de projet est une approche agile c'est-à-dire qu'elle privilégie la flexibilité, l'adaptation rapide aux changements en divisant le travail en petites tâches ou ce que l'on appelle "sprints".

Afin de suivre l'avancement du projet, il est nécessaire d'utiliser un outil de gestion des tâches, le choix a été Trello.

Dans le but d'avoir un code cohérent et maintenable au niveau de l'API, l'application web ou mobile, le choix de conventions/normes de développement est indispensable. Ainsi les variables seront en anglais et en camelCase.

Les maquettes et schémas seront réalisés sur Figma ou draw.io, les diagrammes de classe à l'aide de StarUML.

BACKLOG DE PRODUIT

Le backlog de produit est une liste hiérarchisée des tâches qui définit donc quelle fonctionnalité est prioritaire. Celui du projet Gourmetise est ci-dessous.

Backlog de produit *La Gourmetise*

	N°	En tant que ...	je souhaite ...	Priorité
Application web [AppWeb]	1	gestionnaire	paramétrer les données du concours (calendrier)	Basse
	2	visiteur	créer un compte pour participer au concours	Moyenne
	3	utilisateur	me connecter	Moyenne
	4	participant	inscrire ma boulangerie au concours	Haute
	5	participant	ajouter des photos/vidéos de ma boulangerie	Moyenne
	6	gestionnaire	générer et publier le classement du concours	Haute
	7	gestionnaire	gérer les candidatures au concours	Moyenne
	8	visiteur	visualiser le classement du concours	Haute
	9	participant	voir le score obtenu par ma boulangerie	Moyenne
Application mobile [AppMobile]	10	juge	récupérer la liste des participants au concours	Haute
	11	juge	évaluer un participant (une boulangerie)	Haute
	12	juge	envoyer mes évaluations	Haute

API

API Création d'un utilisateur

L'API de création d'utilisateur permet de créer des comptes utilisateurs pour l'application web notamment.Elle offre un endpoint RESTful pour interagir avec. Cette API ne demande pas de jeton d'authentification (token JWT).

POST localhost:8000/api/register

Méthode : `POST`

URL : `/api/register`

Description : Crée un nouvel utilisateur.

Paramètres du corps :

- **`email` (string, requis) : L'email de l'utilisateur qui sera son identifiant.**
- **`rôle` (string, requis) :Le rôle de l'utilisateur (par exemple, "participant", "gérant").**
- **`password` (string, requis) : Le mot de passe de l'utilisateur.**
- **Réponse :**

`201 Created` : Si l'utilisateur est créé avec succès.

Exemple entrée

```json
{
    "email": "test20@gmail.com",
    "role": "participant",
    "password": "test20"
}
```

Exemple réponse

```
{
    "message": "Création de compte réussit",
    "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpYXQiOjE3MzQ2
        80xNbC9sfB4fg9JLg-R7mrhDoYBeNNBsklQ0-MRUcgICA_AWfn4G81KvoY4
        GIIhJxx0xLrpjvjAXxRGcrZLxwkgQT0XVTDmS56e-yZWII7Cuj8Y0UuCU9F
}
```

400 Bad Request : Si l'email est déjà utilisé.

Exemple réponse

```
{
    "message": "Cet email est déjà utilisé."
}
```

400 Bad Request : Si le concours a déjà commencé.

Exemple réponse

```
{
    "message": "Les inscriptions au concours sont fermés, vous ne pouvez plus vous inscrire."
}
```

API Connexion à son compte

L'API de connexion permet de se connecter sur l'application web.Elle offre un endpoint RESTful pour interagir avec. Cette API ne demande pas de jeton d'authentification (token JWT)

.

POST localhost:8000/api/login

Méthode : `POST`

URL : `/api/login`

Description : Connexion à son compte.

Paramètres du corps :

- **`email` (string, requis) : L'email de l'utilisateur qui sera son identifiant.**
- **`password` (string, requis) : Le mot de passe de l'utilisateur.**
- **Réponse :**

`200 OK`: Si l'utilisateur est connecté avec succès.

Exemple entrée

```
{
    "email": "f@gmail.com",
    "password": "f"
}
```

Exemple sortie


```
{
    "message": "Connexion réussit",
    "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpYXQiOjE3NDQzNjAxNDks
        2A2B1VAciuv3fivSSNA3pK2roRTGZ0_R5Z001GkGzqLXxoe55XtGeBVIJeqL8SOIq-i
        0qUswkL0PVH_C5sUBYULWe1IGiZ7gmpMKJA8_bdsrD_Wt47mD0Ks7uRVTyyR20HQ6iF
}
```


400 Bad Request : Si le mot de passe ou l'email est incorrect.


```
{
    "message": "Mot de passe ou email incorrect"
}
```


API Inscription d'une boulangerie

L'API permet d'inscrire une boulangerie au concours via l'application web.Elle offre un endpoint RESTful pour interagir avec. Cette API demande d'être authentifié donc d'avoir un jeton d'authentification (token JWT) car la boulangerie est reliée à un compte utilisateur. Ainsi, pour s'inscrire, la personne doit disposer d'un compte utilisateur et doit être connectée.

POST localhost:8000/api/bakeries

Méthode : `POST`

URL : `/api/bakeries`

Description : Inscription d'une boulangerie au concours.

Paramètres du corps :

- **`siret`(string, requis) : Le numéro siret de la boulangerie, qui est son identifiant.**
- **`name`(string, requis) : Le nom de la boulangerie.**
- **`street`(string, requis) : La rue de la boulangerie.**
- **`postal_code`(string, requis) : La code postal de la boulangerie.**
- **`city`(string, requis) : La ville de la boulangerie.**
- **`telephone_number`(string, requis) : Le numéro de téléphone de la boulangerie.**

- `contact_name`(string, requis) : Le nom du contact relié à la boulangerie.
- `bakery_description`(string, requis) : La description de la boulangerie.
- `products_description`(string, requis) : La description des produits de la boulangerie
- `user`(objet, requis) : L'utilisateur qui veut inscrire la boulangerie.
    - `email`(string, requis) : L'email de l'utilisateur.
- Réponse :

`201 Created` : Si la boulangerie est créée avec succès.

Exemple entrée

```
{
    "siret": "127 456 789 00032",
    "name": "ma boulangerie test3",
    "street": "37 rue du test",
    "postal_code": "38200",
    "city": "Vienne",
    "telephone_number": "04 24 98 01 78",
    "contact_name": "M.Test",
    "bakery_description": "La description de la boulangerie",
    "products_decription": "La description des produits",
    "user": {
        "email": "test5@gmail.com"
    }
}
```

Exemple réponse

```
{
    "message": "Boulangerie inscrite avec succès ! "
}
```

409 Conflict : Si l'email est déjà utilisé, l'utilisateur a déjà inscrit une boulangerie.

Exemple réponse

```
{
    "message": "Vous avez déjà inscrit une boulangerie avec ce compte."
}
```

409 Conflict : Si le numéro de siret est déjà utilisé

Exemple réponse

```
{
    "message": "Numéro de siret déjà existant, cette boulangerie est déjà inscrite."
}
```

400 Bad Request : Si les inscriptions aux concours ne sont pas encore ouvertes.

Exemple réponse

```
{
    "message": "Les inscriptions aux concours ne sont pas encore ouvertes."
}
```

400 Bad Request : Si le concours a déjà commencé.

Exemple réponse

```
{
    "message": "Le concours a déjà débuté, les inscriptions sont fermés."
}
```

400 Bad Request : Si les informations envoyées sont incorrectes ou incomplètes.

Exemple entrée

```
{
    "siret": "",
    "name": "",
    "street": "23s avenue Victor Hugo",
    "postal_code": "69000",
    "city": "Lyon",
    "telephone_number": "04 24 98 01 78",
    "contact_name": "M.Test",
    "bakery_description": "Une petite boulangerie Marie Blachère au coin
        l'odeur alléchante des produits fraîchement cuits.",
    "products_decription": "La description des produits",
    "user": {
        "email": "test6@gmail.com"
    }
}
```

Exemple réponse

```
{
    "message": "Les informations de la boulangerie ne sont pas complètes."
}
```

400 Bad Request : Si le mail ne correspond à aucun compte utilisateur.

Exemple sortie

```
{
    "message": "Vous n'avez pas de compte."
}
```

400 Bad Request : Si l'utilisateur n'a pas le rôle participant.

Exemple sortie

```
{
    "message": "Vous n'etes pas un participant."
}
```

API EXTERNE Communes

GET https://geo.api.gouv.fr/communes?codePostal={param}

L'API permet de récupérer les villes reliées à un code postal passé en paramètre. Ainsi, lorsque l'utilisateur remplit le formulaire et notamment le code postal, au niveau des villes une liste déroulante apparaît et a comme valeurs la réponse de l'API.Cela guide et aide l'utilisateur sur le formulaire d'inscription.

API Récupérer la liste des boulangeries + paramètres du concours

L'API permet de récupérer la liste des boulangeries participantes ainsi que les paramètres du concours. Cette API ne demande pas de jeton d'authentification (token JWT).

GET localhost:8000/api/mobile/bakeries

Méthode : `GET`

URL : `/api/mobile/bakeries`

Description : Récupérer la liste des boulangeries et les paramètres du concours.

Réponses :

`200 OK` : Aucun problème, les données sont récupérées.

Exemple sortie


```
{
    "bakeries": [
        {
            "siret": "010 201 021 52352",
            "name": "ma petite boulangerie",
            "street": "89 rue de Gea",
            "postal_code": "38200",
            "city": "Chuzelles",
            "telephone_number": "10 20 30 10 20",
            "bakery_description": "La description de la boulangerie",
            "products_decription": "La description des produits"
        },
        {
            "siret": "152 121 212 12414",
            "name": "Petit pain",
            "street": "34 rue des communes",
            "postal_code": "38200",
            "city": "Seyssuel",
            "telephone_number": "01 45 62 34 47",
            "bakery_description": "la description de la boulangerie",
            "products_decription": "la description des produits"
        }
    ],
    "contestParams": {
        "startRegistration": "2024-04-14T00:00:00+00:00",
        "endRegistration": "2025-05-11T00:00:00+00:00",
        "startEvaluation": "2024-05-12T00:00:00+00:00",
        "endEvaluation": "2025-07-06T00:00:00+00:00"
    }
}
```


400 Bad Request : Si les votes ne sont pas encore ouverts.

Exemple sortie


```
{
    "message": "Les votes au concours ne sont pas encore ouverts."
}
```


409 Conflict : S'il n'y a aucune boulangerie inscrites au concours.

Exemple sortie


```
{
    "message": "Il n'y a aucune boulangerie inscrite"
}
```

API Enregistrement de notes

L'API d'enregistrement des notes permet d'enregistrer en bdd MySQL les notes envoyées depuis l'application mobile par les juges.Elle offre un endpoint RESTful pour interagir avec. Cette API ne demande pas de jeton d'authentification (token JWT).

POST localhost:8000/api/mobile/evaluations

Méthode : `POST`

URL : `/api/mobile/evaluations`

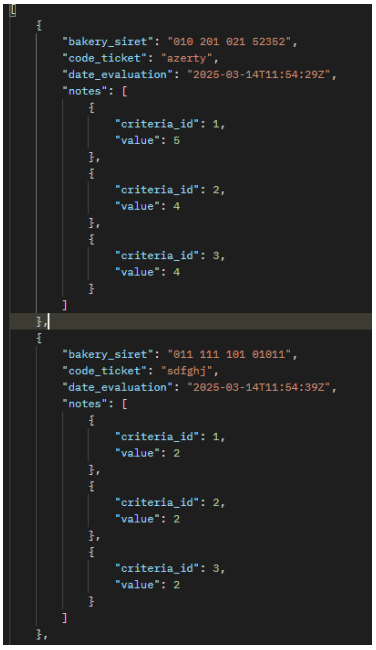
Description : Enregistre les notes reçues depuis l'application mobile.

Paramètres du corps :

- **`bakery_siret`(string, requis) :Le numéro de siret de la boulangerie qui est son identifiant.**
- **`code_ticket`(string, requis): Le code du ticket de l'évaluation**
- **`date_evaluation`(string, requis) :La date de l'évaluation.**
- **`notes`(tableau, requis) :Tableau contenant les notes des différents critères. → un tableau d'objet de la forme {criteria id: int, value: int}**
- **Réponse :**

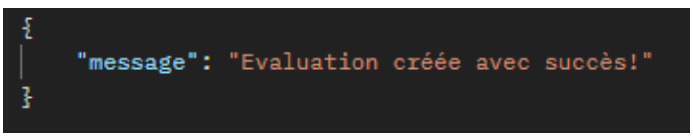
`201 Created` : Si l'utilisateur est créé avec succès.

Exemple entrée



```
[
    {
        "bakery_siret": "010 201 021 52352",
        "code_ticket": "azerty",
        "date_evaluation": "2025-03-14T11:54:29Z",
        "notes": [
            {
                "criteria_id": 1,
                "value": 5
            },
            {
                "criteria_id": 2,
                "value": 4
            },
            {
                "criteria_id": 3,
                "value": 4
            }
        ]
    },
    {
        "bakery_siret": "011 111 101 01011",
        "code_ticket": "sdfghj",
        "date_evaluation": "2025-03-14T11:54:39Z",
        "notes": [
            {
                "criteria_id": 1,
                "value": 2
            },
            {
                "criteria_id": 2,
                "value": 2
            },
            {
                "criteria_id": 3,
                "value": 2
            }
        ]
    },
```

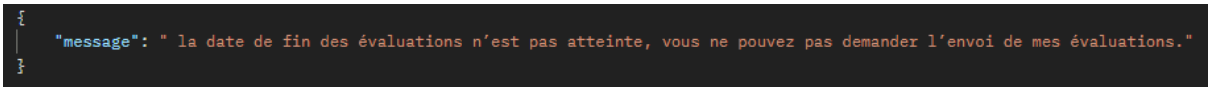
Exemple réponse



```
{
    "message": "Evaluation créée avec succès!"
}
```

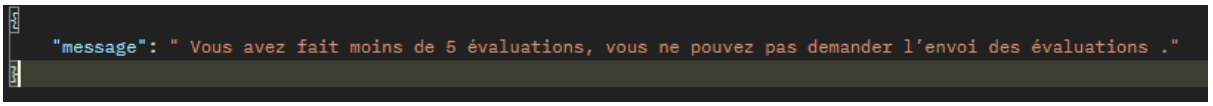
400 Bad Request : La date de fin des évaluations n’est pas atteinte.

Exemple sortie



400 Bad Request : Le nombre d'évaluations est inférieur à 5.

Exemple sortie



API Génération du classement

L’API permet de générer le classement du concours en tant que gestionnaire, sans le publier, c’est uniquement de l’affichage.Elle offre un endpoint RESTful pour interagir avec. Cette API demande un jeton d’authentification (token JWT) et en plus de cela d’avoir le ROLE_GERANT.

GET localhost:8000/api/getFullRanking

Méthode : `GET`

URL : `/api/getFullRanking`

Description : Renvoie le classement général du concours afin de pouvoir le consulter en tant que gérant.

Réponses :

`200 OK` : Aucun problème, les données sont récupérées.

Exemple sortie

```
{
    "rankings": [
        {
            "siret": "005 456 789 00974",
            "name": "La Mie Gourmande",
            "average_score": "12.0000"
        },
        {
            "siret": "127 456 789 00998",
            "name": "Marie Blachère",
            "average_score": "12.0000"
        },
        {
            "siret": "452 465 656 56555",
            "name": "test",
            "average_score": "12.0000"
        },
        {
            "siret": "789 415 002 74263",
            "name": "Pierre",
            "average_score": "10.6667"
        },
        {
            "siret": "123 415 673 44444",
            "name": "Paul",
            "average_score": "9.8000"
        },
        {
            "siret": "403 415 666 66666",
            "name": "Jak",
            "average_score": "9.3333"
        },
```


401 Unauthorized : Un token d'authentification jwt est nécessaire.

Exemple sortie


```
{
    "code": 401,
    "message": "Expired JWT Token"
}
```


403 Forbidden: Il faut être authentifié avec le rôle gérant.

Exemple sortie


```
{
    "message": "Vous n'etes pas autorisé à consulter le classement général."
}
```


400 Bad Request: Il faut que la date d'évaluation soit passée.

Exemple sortie

```
{
    "message": "Les évaluations aux concours ne sont pas encore finies."
}
```


API Publication du concours

L'API permet de publier le classement du concours en tant que gestionnaire.Elle offre un endpoint RESTful pour interagir avec. Cette API demande un jeton d'authentification (token JWT) et en plus de cela d'avoir le ROLE_GERANT.

PATCH localhost:8000/api/updateContestParams

Méthode : PATCH

URL : `/api/contestParams`

Description : Permet de modifier les paramètres du concours afin de publier ce dernier.

Réponses :

`200 OK` : Aucun problème, le concours est publié.

Exemple entrée


```
{
    "published": true
}
```


Exemple sortie


```
{
    "message": "Paramètres du concours mis à jour."
}
```


401 Unauthorized : Un token d'authentification jwt est nécessaire.

Exemple sortie


```
{
    "code": 401,
    "message": "Expired JWT Token"
}
```

403 Forbidden: Il faut être authentifié avec le rôle de gérant.

Exemple sortie


```
{
    "message": "Vous n'etes pas autorisé à modifier les paramètres de concours."
}
```


API Affichage du classement du concours

L'API permet d'afficher le classement du concours, le top 3.Elle offre un endpoint RESTful pour interagir avec. Cette API ne demande pas un jeton d'authentification (token JWT).

GET localhost:8000/api/getRankingTop3

Méthode : `GET`

URL : `/api/getRankingTop3`

Description : Renvoie le top 3 des meilleures boulangeries.

Réponses :

`200 OK` : Aucun problème, les données sont affichées.

Exemple sortie


```
{
    "rankings": [
        {
            "siret": "005 456 789 00974",
            "name": "La Mie Gourmande",
            "average_score": "12.0000"
        },
        {
            "siret": "452 465 656 56555",
            "name": "test",
            "average_score": "12.0000"
        },
        {
            "siret": "127 456 789 00998",
            "name": "Marie Blachère",
            "average_score": "12.0000"
        }
    ]
}
```

400 Bad Request: Le classement du concours n'est pas publié.

Exemple sortie

```
{
    "message": "Le classement n'a pas été publié."
}
```

API Affichage des résultats de sa boulangerie au concours

L'API permet d'afficher le résultat du concours pour sa boulangerie.Elle offre un endpoint RESTful pour interagir avec. Cette API demande un jeton d'authentification (token JWT) et d'avoir le rôle participant.

GET localhost:8000/api/getMyRank

Méthode : `GET`

URL : `/api/getMyRank`

Description : Renvoie les résultats de sa boulangerie.

Réponses :

`200 OK` : Aucun problème, les données sont affichées.

Exemple sortie

```
{
    "myRank": {
        "siret": "127 456 789 00998",
        "name": "Marie Blachère",
        "average_score": "12.0000",
        "rank_value": 1
    }
}
```

401 Unauthorized : Un token d'authentification jwt est nécessaire.

Exemple sortie

```
{
    "code": 401,
    "message": "Expired JWT Token"
}
```

400 Bad Request: Le classement du concours n’est pas publié.

Exemple sortie

```
{
    "message": "Le classement n'a pas été publié."
}
```

400 Bad Request: L’utilisateur n’a pas inscrit de boulangerie.

Exemple sortie

```
{
    "message": "La boulangerie associée à votre compte n'a pas été trouvé ou n'existe pas"
}
```

API Modification des paramètres du concours

L’API permet de modifier les paramètres du concours tel que les différentes dates.Elle offre un endpoint RESTful pour interagir avec. Cette API demande un jeton d’authentification (token JWT) et d’avoir le rôle gérant.

PATCH localhost:8000/api/contestParams/dates

Méthode : PATCH

URL : `/api/contestParams/dates`

Description : Permet de modifier les paramètres du concours tel que les différentes dates.

Réponses :

`200 OK` : Aucun problème, le concours est modifié.

Exemple entrée

```
{
  "startRegistration": "2025-04-02",
  "endRegistration": "2025-04-03",
  "startEvaluation": "2025-04-04",
  "endEvaluation": "2025-04-05"
}
```

Exemple sortie


```
{
    "message": "Paramètres du concours mis à jour."
}
```


400 Bad Request: Le concours a déjà commencé.

Exemple sortie


```
{
    "message": "Les inscriptions au concours ont déjà commencé, vous ne pouvez plus les modifier."
}
```


400 Bad Request: Vous n'êtes pas connecté en tant que gérant.

Exemple sortie


```
{
    "message": "Vous n'etes pas autorisé à consulter modifier les paramètres de concours."
}
```

APPLICATION WEB

DIAGRAMME DE CLASSE

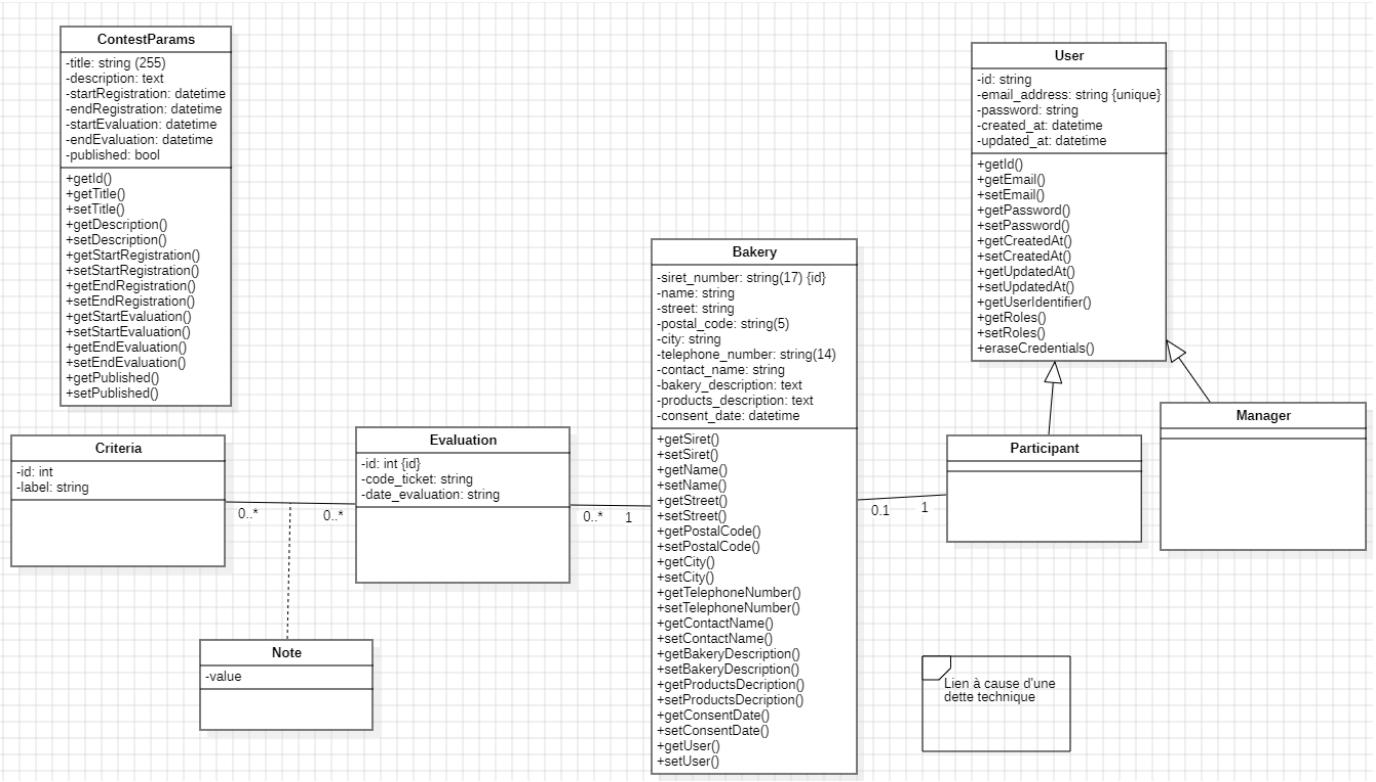
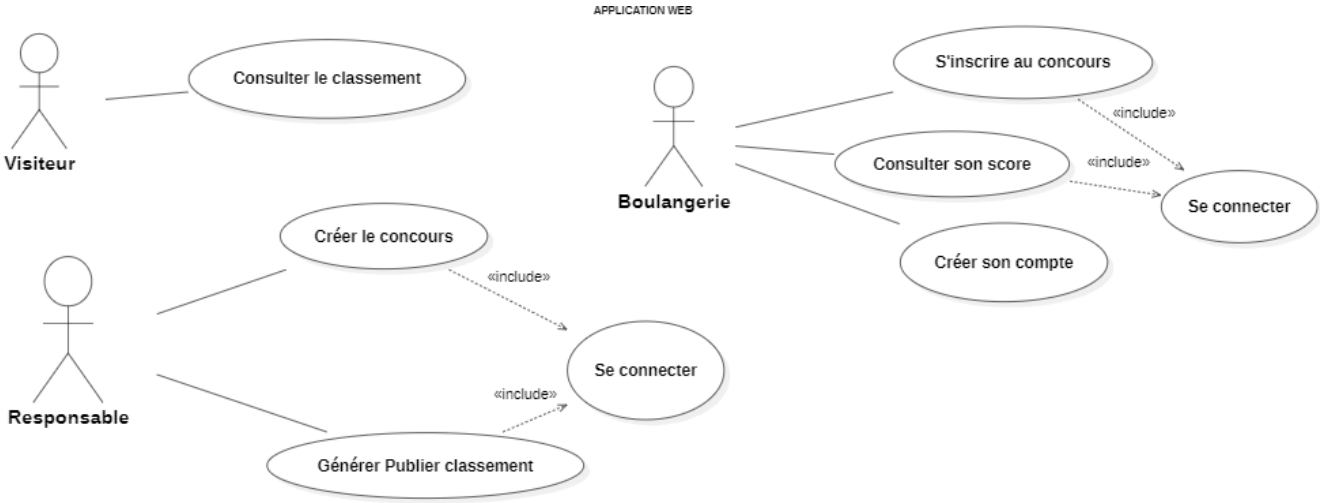
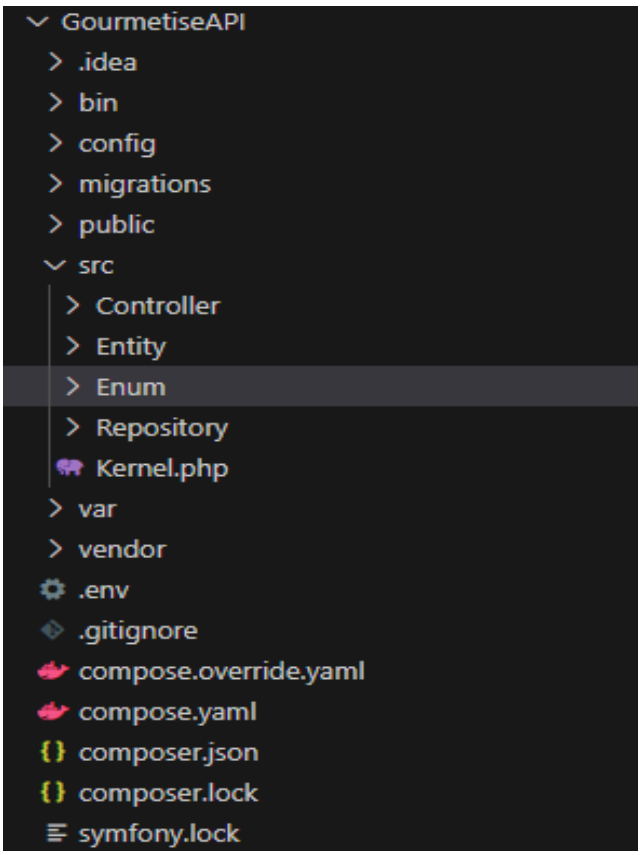
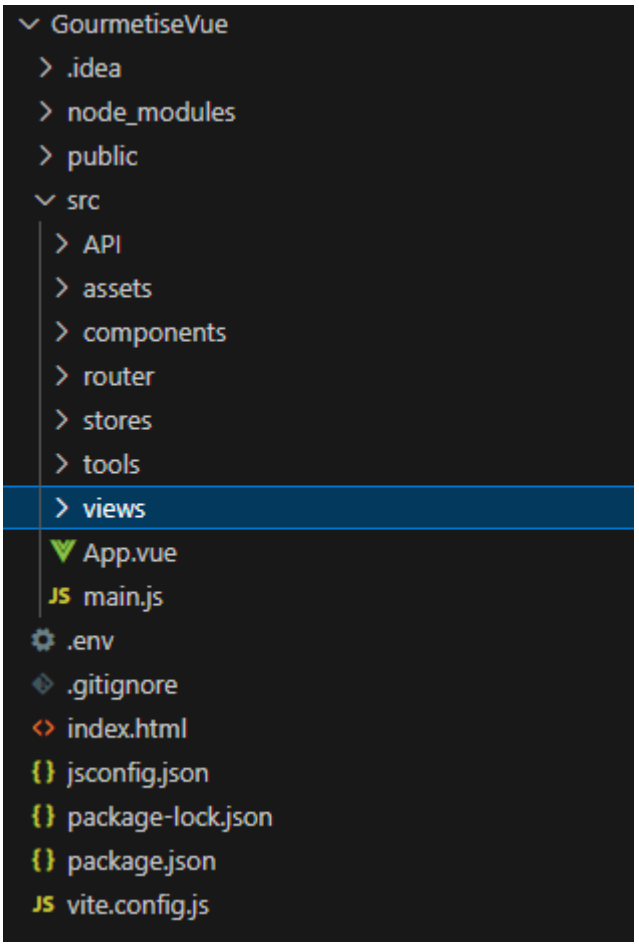


DIAGRAMME DE CAS D'UTILISATION




GourmetiseAPI
.idea
bin
config
migrations
public
src
Controller
Entity
Enum
Repository
Kernel.php
var
vendor
.env
.gitignore
compose.override.yaml
compose.yaml
composer.json
composer.lock
symfony.lock


GourmetiseVue
.idea
node_modules
public
src
API
assets
components
router
stores
tools
views
App.vue
main.js
.env
.gitignore
index.html
jsconfig.json
package-lock.json
package.json
vite.config.js

APPLICATION MOBILE

DIAGRAMME DE CLASSE

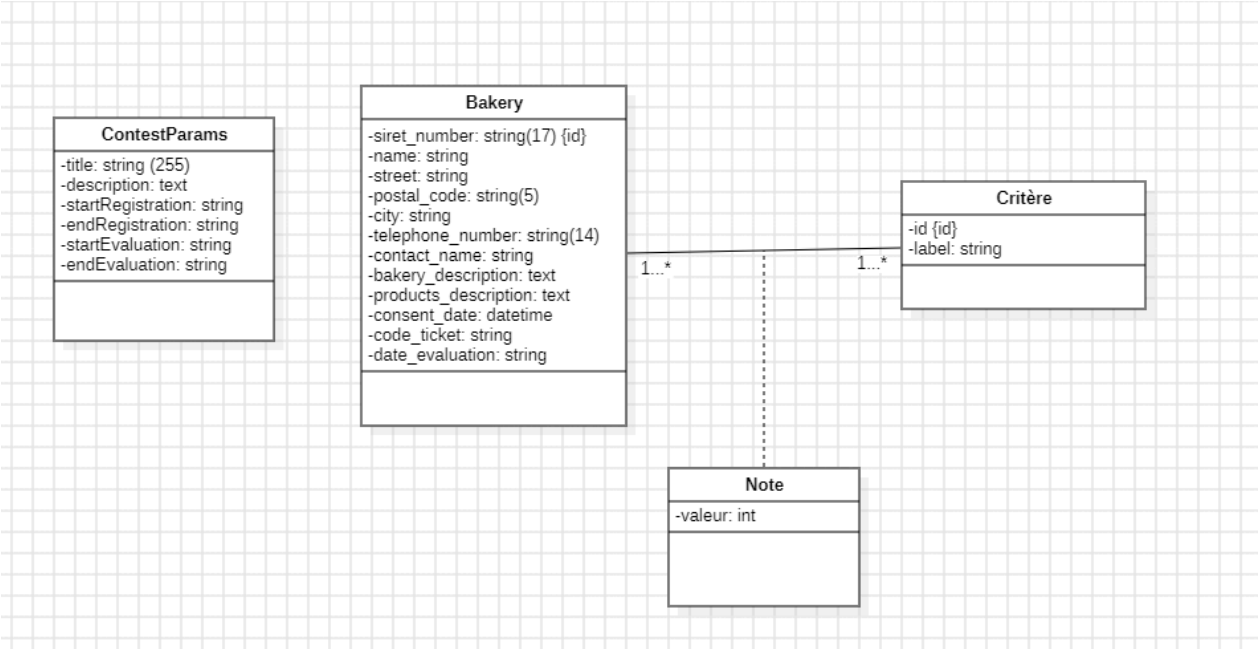
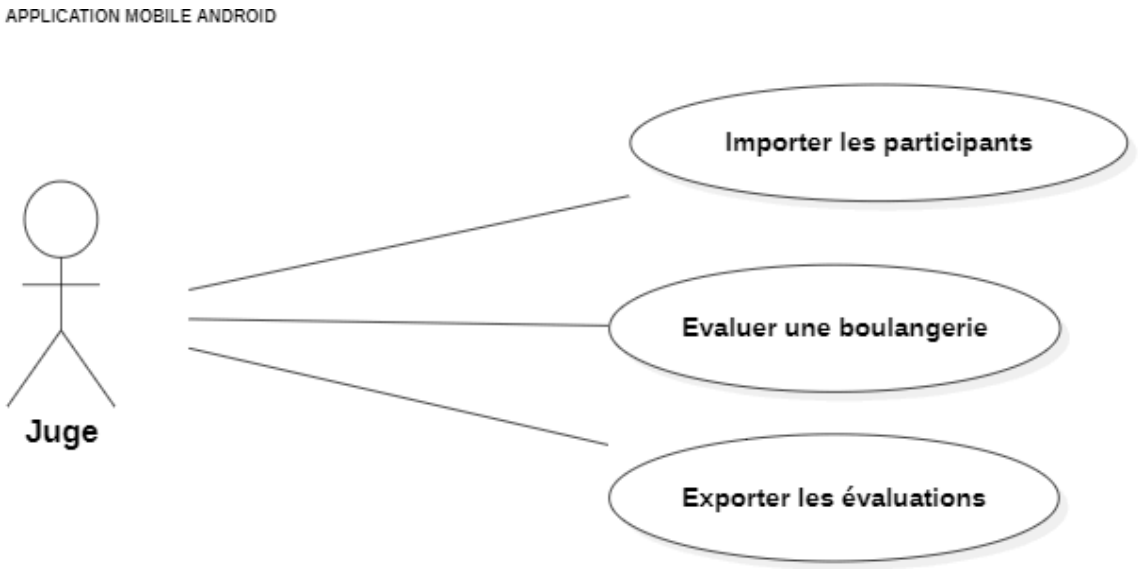


DIAGRAMME DE CAS D'UTILISATION



SÉCURITÉ

La mise en place de moyens de sécurité sur le projet Gourmetise permet de protéger les données des utilisateurs, prévenir les attaques malveillantes et garantir la confidentialité des informations. De plus, la mise en place de conformité avec les réglementations telle que le RGPD est essentielle pour éviter des sanctions légales.

LOGS

Les logs, ou journalisation, permettent de suivre et d'enregistrer toutes les actions et événements de notre base de données, assurant ainsi une traçabilité complète des opérations pour faciliter le diagnostic et la résolution de problèmes.

Pour ce faire, les logs s'effectuent sur une base à part et à l'aide de 3 triggers (créer, supprimer et modifier) pour chacune des tables de notre base de données.

```sql
CREATE TABLE logs
(
    id INT AUTO_INCREMENT,
    table_name VARCHAR(50),
    data_id VARCHAR(50),
    action VARCHAR(10),
    user VARCHAR(50),
    date_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    CONSTRAINT pk_gourmetise_logs PRIMARY KEY(id)
);
```

La CNIL recommande de limiter les données personnelles dans les logs et que ces derniers ont une durée de conservation de 6 mois à 1 an.Afin de respecter ce principe, une procédure qui purge les logs est nécessaire.

```sql
DELIMITER $$

CREATE PROCEDURE logs_purge()
BEGIN
    DELETE FROM logs
    WHERE date_time < DATE_SUB(NOW(), INTERVAL 6 MONTH);

END$$
```

EXPRESSION RÉGULIÈRE

La mise en place de vérification en back et front end des données rentrées par l'utilisateur permet d'éviter des failles XSS. Les expressions régulières empêchent de nombreuses failles XSS où l'attaquant injecte du code malveillant.La vérification

côté serveur est toujours effectuée car il ne faut jamais faire confiance aux entrées utilisateurs.“Never Trust User Input”.

TOKEN JWT

Le JWT (JSON Web Token) est un moyen sécurisé de transmettre des informations entre un serveur et un client sous forme d'un jeton signé, généralement utilisé pour l'authentification et l'autorisation dans les applications web.

Pour ce faire l’API utilise le bundle Lexik qui est "lexik/jwt-authentication-bundle" afin de gérer les jetons JWT. Chaque endpoint demandant une authentification ou un certain rôle demande la présence d’un jeton JWT dans l’en-tête de la requête, que seul le serveur possédant la clé secrète peut vérifier pour en garantir l’authenticité.