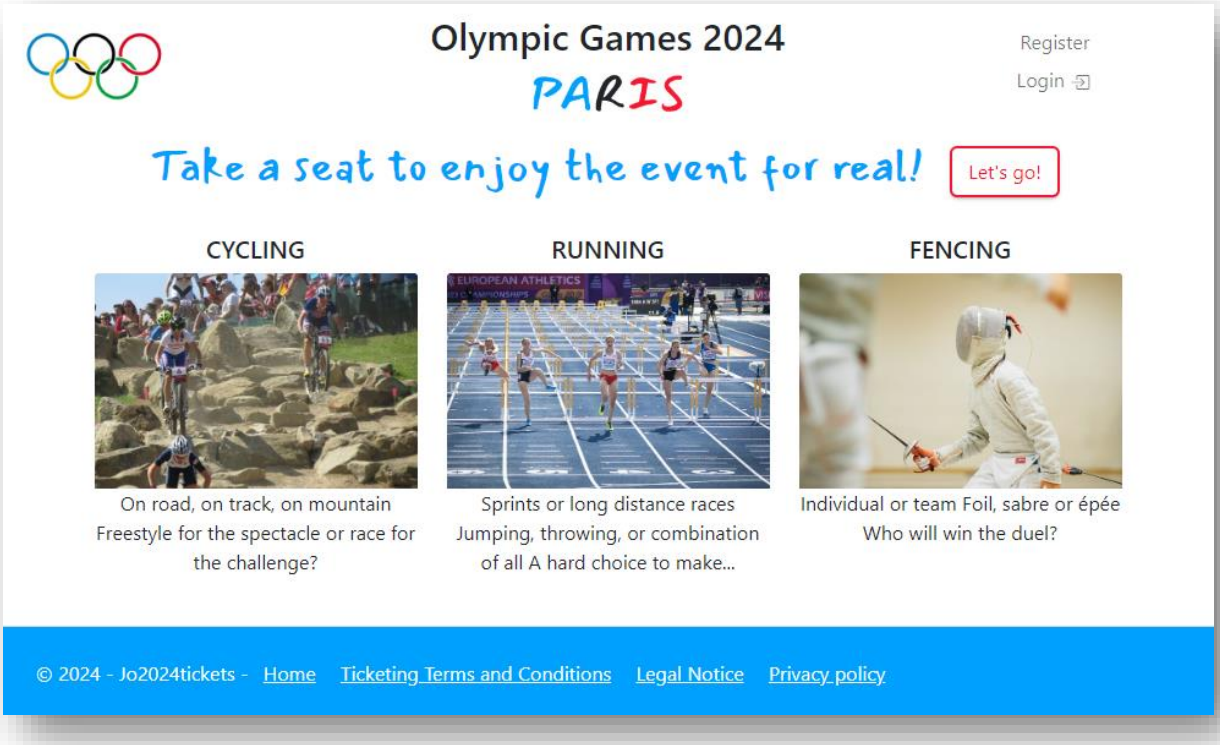
Site web
Jo2024Tickets
Documentation technique



Ce document présente toutes les informations techniques du site Jo2024tickets, pour assurer la maintenance et l'évolution du site.

Date révision	*Nom rédacteur*	*Modifications apportées*
25/04/2024	G. Vieillescazes	Création

Table des matières

1. Contexte fonctionnel

1.1 Acteurs du projet

- Un développeur full-stack : développement de l'application,
- Un chef de projet : gestion du projet.

1.2 Contraintes métier

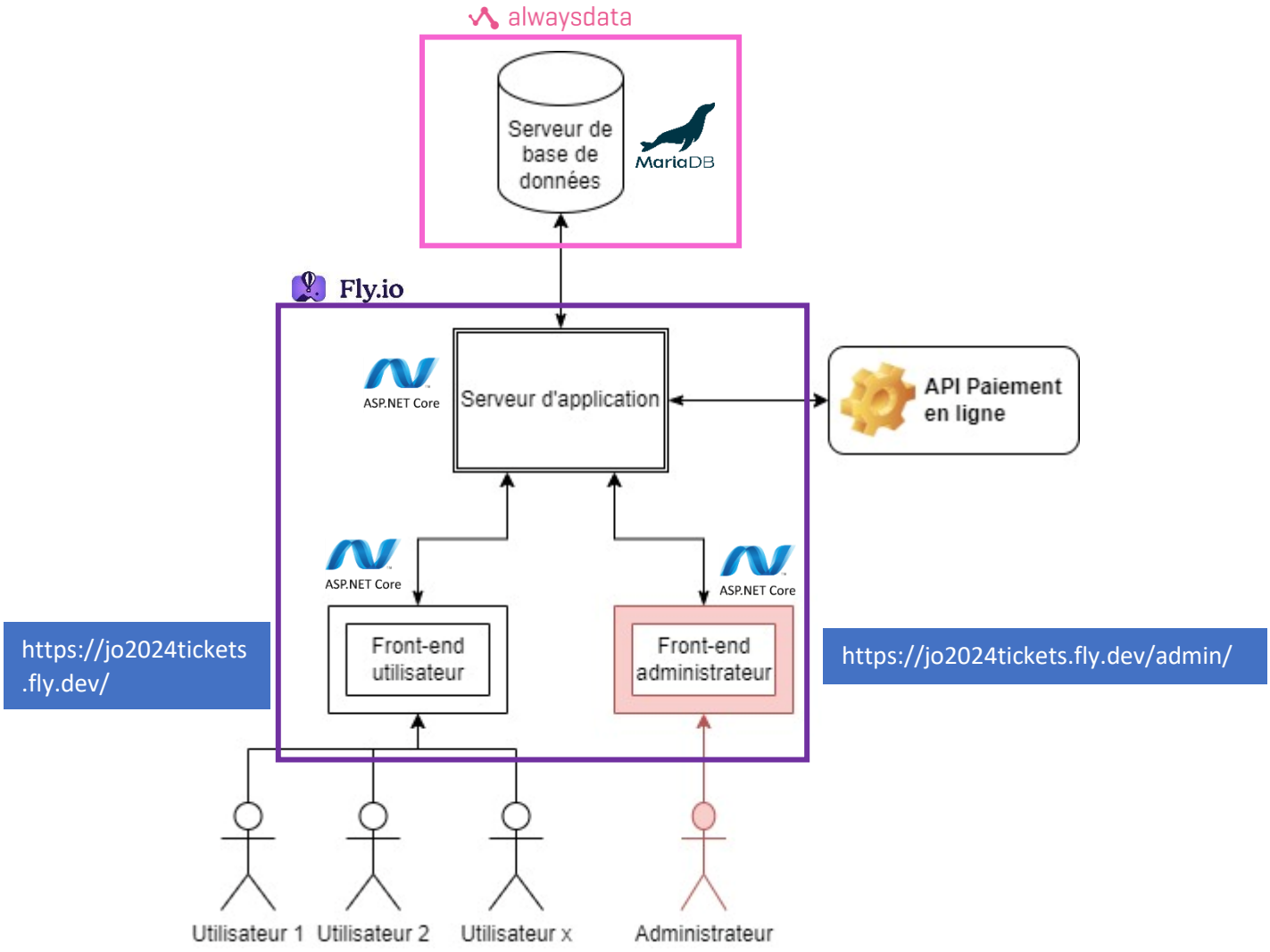
- Application disponible 2 mois avant les JO pour permettre la réservation des billets (soit le 26/05/2024)
- Application accessible sur internet depuis le monde entier, avec gestion des surcharges de requêtes http
- Application disponible sur tous les terminaux et tous les systèmes d'exploitation
- Serveur avec une disponibilité maximale
- Sécurisation des billets
- Sécurisation des données de l'application

1.3 Objectif du rendu final

- Une application pour assurer la vente en ligne des billets des JO
- Une application pour assurer le contrôle d'accès des détenteurs de billets le jour de l'événement, et gérer la vente des billets

2. Contexte technique

2.1 Architecture



2.2 Hébergement du site

Les front-end et back-end du site sont hébergés sur les serveurs Fly.io, avec la configuration suivante :

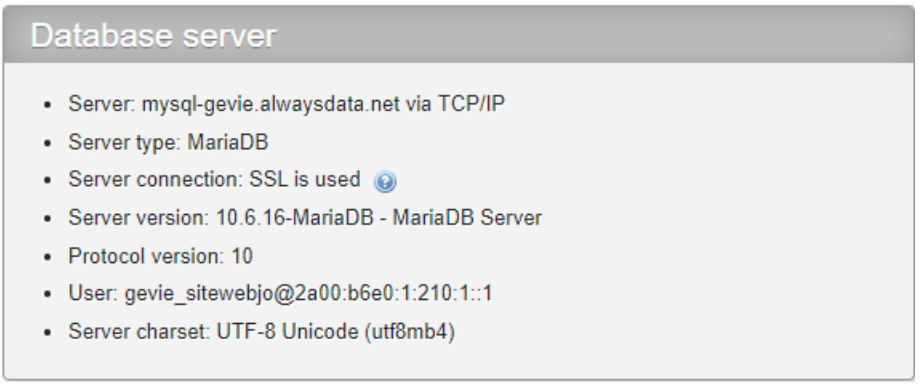
App Settings

```
app = "jo2024tickets"
primary_region = "cdg"

[http_service]
auto_start_machines = true
auto_stop_machines = true
force_https = true
internal_port = 8_080
min_machines_running = 0
processes = [ "app" ]

[[vm]]
cpu_kind = "shared"
cpus = 1
memory = "1gb"
```

La base de données est hébergée sur les serveurs AlwaysData.com, avec la configuration suivante :



2.3 SGBD

2.3.1 Choix du SGBD

Le SGBD a été choisi pour être compatible avec l'hébergeur de la base de données (AlwaysData.com) et pour qu'il soit facilement intégrable dans une solution .Net avec EntityFramework, Identity Framework et la génération de base de données en code first.

Environnement de développement :

SGBD : PhpMyAdmin, via XAMPP

Base de données : MariaDb, version 10.4.28

Nom de la base de données : gevie_sitewebjo_basedev

Chaine de connexion (mot de passe à ajouter) : server=127.0.0.1;database=gevie_sitewebjo_basedev;uid=root;pwd="";

Environnement de production :

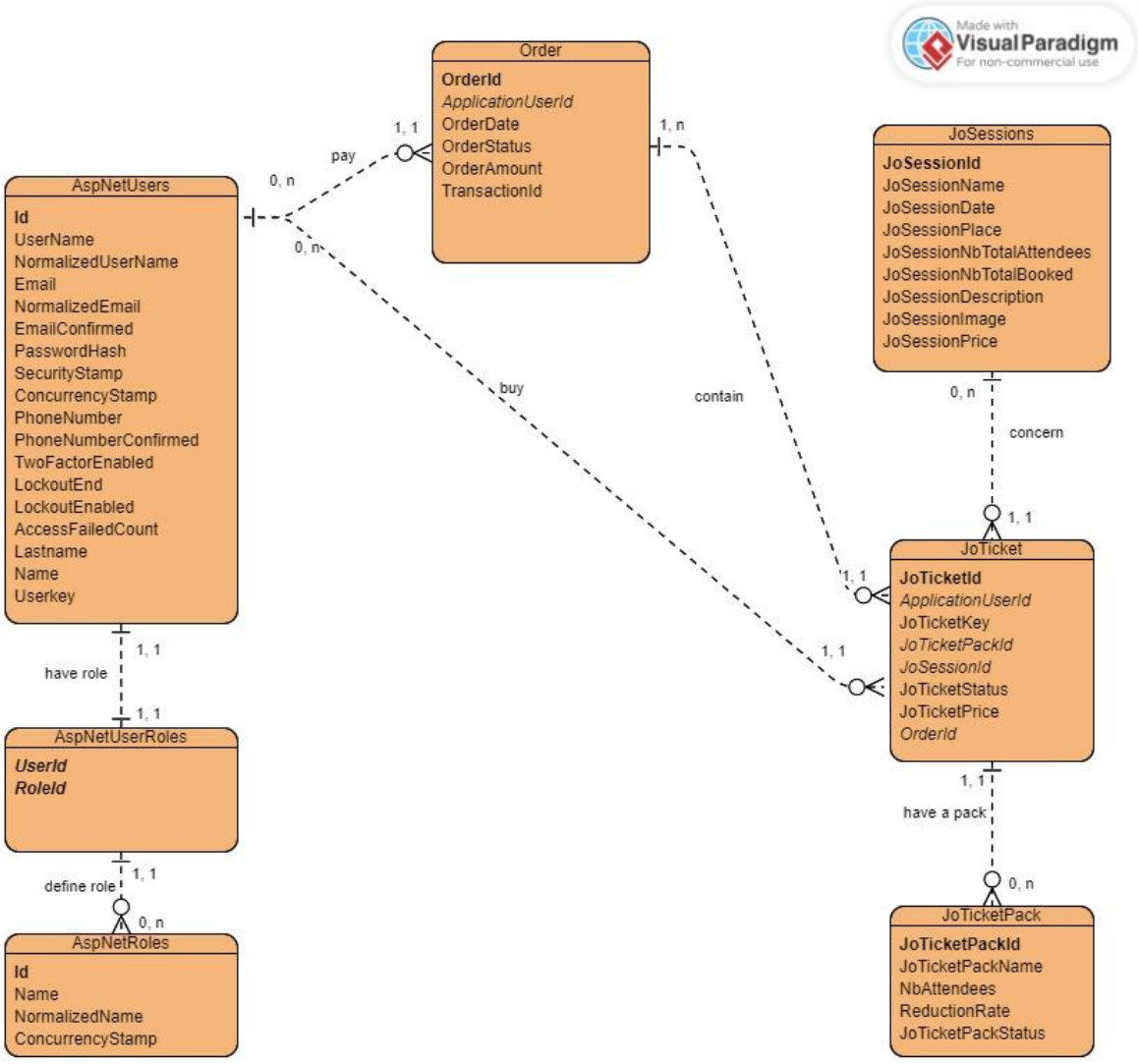
SGBD : PhpMyAdmin, via AlwaysData.com

Base de données : MariaDb, version 10.6.16

Nom de la base de données : gevie_sitewebjo_baseprod

Chaine de connexion (mot de passe à ajouter) : server=mysql-gevie.alwaysdata.net;database=gevie_sitewebjo_baseprod;uid=gevie_sitewebjo;pwd="";

2.3.2 MCD



Note : AspNetCore.Identity créé des tables qui ne sont pas toutes utilisées dans cette application, elles ne sont pas représentées dans ce schéma. Néanmoins elles existent dans la base de données.

2.4 Librairies utilisées

2.4.1 Entity Framework Core

Ce framework est utilisé comme Object-Relational Mapping.

Licence : MIT

2.4.2 Asp Net Core Identity

Ce framework est utilisé pour la gestion des comptes utilisateurs : création de comptes, connexions, gestion des accès, …

Licence : MIT

2.4.3 Pomelo.EntityFrameworkCoreMySql

Ce framework est utilisé comme fournisseur d'Entity Framework Core pour la base de données MariaDB.

Licence : MIT

2.4.4 SendGrid

SendGrid est un service d'envoi d'email, préconisé par Microsoft. Il est utilisé pour confirmer l'adresse email du compte utilisateur et récupérer le mot de passe dans ASP.NET Core. Il est aussi utilisé pour l'envoi des tickets et facture à l'utilisateur.

Compte SendGrid utilisé : gevie46000@gmail.com (connexion avec Google authentification)

Clé API SendGrid utilisée : *SiteWebJO*

Compte email utilisé (sender SendGrid) : gevie46000@gmail.com

(cet email devra être remplacé par un email officiel avec le nom de domaine)

Package nuget utilisé : *SendGrid* version 9.29.3

Licence : MIT

2.4.5 QRCoder

Ce package est utilisé pour créer des QR code basé sur des chaines de caractères.

https://www.nuget.org/packages/QRCoder/#readme-body-tab

Licence : MIT

2.4.6 MSTest

MSTest est utilisé pour réaliser les tests unitaires de l'application.

Les tests unitaires sont réalisés dans le projet *SiteWebJo2.UnitTests* de la solution.

Licence : MIT

2.4.7 Instascan

Cette librairie est utilisée pour la lecture de QR code avec la webcam du terminal. Librairie Javascript

Licence : MIT

2.5 Gestion de code source

La gestion de code source utilisée est Git, avec un stockage du code dans GitHub à l'adresse suivante :

https://github.com/GeVie46/SiteWebJO2

3. Sécurité de l'application

3.1 Sécuriser les données contre le vol et la perte

Les données stockées sous AlwaysData.com sont sauvegardées quotidiennement en datacenter distinct et accessibles 24/7. La SGBD est entièrement infogérée. Pour plus d'informations sur la sécurité de l'hébergeur (inclus le plan de continuité d'activité) :

https://help.alwaysdata.com/fr/s%C3%A9curit%C3%A9/

Concernant le stockage de l'application sur Fly.io, les informations de sécurité de cet hébergeur se trouve ici :

https://fly.io/docs/about/security/

3.2 Authentification de l'utilisateur

Lors de la création du compte, on utilise l'authentification à 2 facteurs, avec un mot de passe et une vérification de l'adresse email saisie pour le compte. Ce service d'authentification est géré par ASP NET CORE Identity.

Le niveau de sécurité du mot de passe utilisateur est vérifié, il doit être au minimum composé de : un caractère majuscule, un caractère minuscule, un chiffre et un caractère spécial. Il doit être d'au moins 6 caractères.

3.3 Sécurité des données saisies par l'utilisateur

Les données saisies par l'utilisateur sont sécurisées contre l'injection SQL :

- Création / modification / suppression du compte / d'offre commerciale : on utilise des attributs de contrôle de données ASP.NET : [Required], [StringLength(…)], …
- Saisie des informations de paiement : cette partie est sécurisée directement par l'API de paiement.

L'envoi des requêtes est également sécurisé avec des tokens CSRF.

3.4 Stockage des clés de sécurité nécessaires à l'application

Dans l'environnement de développement, les clés sont stockées dans le user secrets manager de Visual Studio.

Dans l'environnement de production, elles sont stockées dans le secret vault de l'application déployée sur fly.io. Documentation sur le stockage de secrets sous fly.io :

https://fly.io/docs/reference/secrets/#setting-secrets

Les secrets stockés sont :

- La clé d'authentification à l'API de paiement, nommée ApiPaymentKey
- La chaine de connexion à la base de données, nommée DATABASE_URL
- La clé de l'API SendGrid, nommée SendGridKey.

3.5 Stockage des données sensibles dans la base de données

Les données sensibles sont stockées dans la base de données sous format haché PBKDF2 avec HMAC-SHA256, 128-bit salt, 256-bit subkey, 10000 iterations.

Les données concernées sont le mot de passe utilisateur.

3.6 Sécurisation des billets

Pour sécuriser les billets, on utilise la méthode de salage :

- A la création du compte utilisateur, une clé de salage (128 bits) est générée aléatoirement et stockée dans la base de donnée (table User).
- Pour chacun des tickets achetés, une autre clé est générée aléatoirement et stockée dans la base de données (table Ticket).
- Le QR code est généré avec les informations suivantes : hachage de la concaténation (clé utilisateur + clé ticket), et l'email de l'utilisateur.

  { Userkey : hash(*user key* + *ticket key*) , email : *User.Email* }

- Le jour de l'événement, le QR code sera scanné, on pourra récupérer le nom et prénom de l'utilisateur. On comparera ensuite le hachage de la clé définitive (dans le QR code) avec le hachage de la concaténation des clés utilisateurs et tickets (dans la base de données).

L'algorithme de hashage utilisé est SHA-256.

Les clés sont générées avec la fonction RandomNumberGenerator().

3.7 Autorisation d'accès

L'accès aux différentes pages du site est géré par les rôles :

- Un rôle 'user' : pour les utilisateurs qui sont connectés à leur compte,

- Un rôle 'admin' : pour les administrateurs. Un seul compte administrateur existe, il est créé directement dans la base de données.

Liste des autorisations d'accès (liste des contrôleurs et des actions de contrôleur de l'application) :

	Visiteur	Utilisateur (authentifié)	Administrateur
Home page	X	X	X
Présentation des tickets disponibles	X	X	X
Présentation des offres commerciales (utilisateur)	X	X	X
Se connecter	X	X	X
Créer un compter utilisateur	X		
Présentation des conditions générales de vente	X	X	X
Présentation des conditions de confidentialité	X	X	X
Présentation des mentions légales	X	X	X
Ajouter un billet à son panier		X	
Afficher son panier		X	
Supprimer un billet à son panier		X	
Faire le paiement de son panier		X	
Modifier les billets de son panier		X	
Présentation des offres commerciales (back office)			X
Vérifier le billet (authenticité et contrôle d'identité)			X
Ajouter ou supprimer des offres commerciales			X

4. Evolutions futures

4.1 Evolutions fonctionnelles

L'application peut intégrer de nouvelles fonctionnalités telles que :

- La gestion des langues (anglais, français, …)
- La gestion de sessions sportives : ajout, modification, suppression de session
- La possibilité de modifier le nom et prénom de la personne détenant les billets, pour qu'ils puissent être différent de la personne achetant les billets. Il faudra néanmoins garder l'email de l'acheteur sur le billet pour pouvoir faire le contrôle d'accès.
- L'affichage des commandes passées, et des billets achetés quand un utilisateur est connecté
- Ajouter un statut au billet, pour savoir s'il a déjà été utilisé pour accéder à l'événement. L'accès à l'événement sera ainsi bloqué à la 2e présentation du même billet.
- Contrôler la disponibilité des places quand on modifie la quantité de billet dans le panier

Cette application peut être utilisée pour les Jeux Paralympiques 2024.

4.2 Evolutions techniques

Ci-dessous les évolutions possibles pour améliorer l'application :

- Améliorer le référencement du site : ajouter des mots clés dans les métadonnées du site, ajouter des liens sortants et backlinks.
- Améliorer le rendu des pages. Par exemple, mettre un carrousel pour la page de présentation des événements, avec une photo pleine page
- OrderId est un entier aujourd'hui, mais il vaudrait mieux le passer en UUID, pour éviter que 2 commandes aient le même numéro si 2 utilisateurs font une commande au même moment. Pour une application déployée sur plusieurs serveurs, ceci faciliterait également la synchronisation des différentes bases de données.
- Limiter les requêtes redondantes à la base de données lors de la gestion de la commande (création de ticket, de QR code, …) : transmettre les données plutôt qu'uniquement les id de tickets et de JoSessions.
- Implémenter la gestion des logs avec la librairie Log4net, pour suivre les levées d'exceptions en mode production : https://logging.apache.org/log4net/