

Documentation Technique

Glossaire	2
Réflexions initiale sur le sujet	4
1. Compréhension des Besoins et Objectifs du Projet	4
2. Analyse des Exigences Fonctionnelles (US)	4
3. Choix Techniques	4
4. Architecture de l'Application	5
5. Sécurité et Performances	5
6. Déploiement et Gestion du Projet	5
7. Livrables	5
8. Etapes développement	5
Configuration de l'environnement de travail	6
1. Pré-requis : Installation de Homebrew	6
2. Installation de PHP	6
3. Installation de Composer	7
4. Installation de Node.js et NPM	8
5. Installation de MySQL	8
6. Installation de Laravel	8
8. Configuration de l'Environnement du Projet	11
Guide de Déploiement et Configuration	12
Configuration Hostinger	12
Configuration Atlas	14
Configuration du Fichier .env	14
Résolution des erreurs	16
1. Erreur bad auth : authentication failed	16
2. Erreur : Organisation des fichiers sur le serveur	17
3. Erreur : Suppression du fichier hot.	17
4. Erreur : Configuration du Fichier .htaccess à la Racine	18
5. Erreur : Configuration du Fichier .htaccess dans le Dossier Public	18
6. Erreur : Configuration du Fichier .env	19
7. Erreur : Configuration de MongoDB Atlas	19

Glossaire

.env

Fichier de configuration utilisé par Laravel pour définir les variables d'environnement nécessaires à l'exécution de l'application, comme les connexions à la base de données et les paramètres d'authentification.

.htaccess

Fichier de configuration utilisé sur les serveurs web Apache pour définir des règles de routage, de redirection, et des permissions de fichiers.

Atlas (MongoDB Atlas)

Service cloud de MongoDB qui permet d'héberger des bases de données NoSQL et d'y accéder via des clusters sécurisés.

Base de données relationnelle

Type de base de données structurant les données en tables interconnectées via des relations, par exemple MySQL.

Base de données NoSQL

Type de base de données qui stocke des données sous forme non relationnelle, comme des documents ou des clés-valeurs, par exemple MongoDB.

Breeze

Package Laravel minimaliste qui fournit des fonctionnalités de base pour l'authentification et la structure des applications front-end.

Cluster

Groupe de serveurs ou d'instances de bases de données dans MongoDB Atlas, conçu pour fournir une redondance et une scalabilité.

Composer

Gestionnaire de dépendances pour PHP, utilisé pour installer des packages et frameworks comme Laravel.

Frontend

Partie de l'application visible par l'utilisateur, gérée ici avec React et Tailwind CSS.

GitHub

Plateforme de gestion de code source permettant le suivi des versions et le travail collaboratif via des branches.

Homebrew

Gestionnaire de paquets pour macOS permettant d'installer et de maintenir des outils en ligne de commande comme PHP, Node.js, et MySQL.

Hostinger

Plateforme d'hébergement web utilisée pour déployer l'application en production.

Inertia.js

Framework de communication front-end/back-end permettant d'utiliser Laravel pour la logique serveur et React ou Vue.js pour le rendu client.

Livewire

Package Laravel qui permet de créer des interfaces dynamiques sans nécessiter de JavaScript explicite.

Migration

Processus permettant de créer ou modifier des tables dans une base de données via des scripts Laravel (commandes artisan).

MongoDB

Base de données NoSQL orientée documents utilisée pour stocker les statistiques des consultations dans ce projet.

MySQL

Système de gestion de bases de données relationnelles utilisé pour stocker les données structurées du projet.

Node.js

Environnement d'exécution JavaScript utilisé pour gérer les dépendances front-end et compiler les assets

.

npm (Node Package Manager)

Gestionnaire de paquets JavaScript permettant d'installer des bibliothèques front-end et outils nécessaires au projet.

PHP

Langage de programmation utilisé pour développer le back-end de l'application avec Laravel.

Production

Environnement final de déploiement de l'application, optimisé pour la performance et la sécurité.

React

Bibliothèque JavaScript utilisée pour développer des interfaces utilisateur dynamiques et réactives.

Route

Chemin défini dans une application web pour connecter une URL à une action ou à une vue spécifique.

SSL (Secure Sockets Layer)

Protocole de sécurité utilisé pour chiffrer les connexions entre un serveur web et un navigateur.

SSH (Secure Shell)

Protocole réseau utilisé pour accéder à distance à un serveur et exécuter des commandes en ligne de commande.

Tailwind CSS

Framework CSS utilitaire utilisé pour créer des interfaces utilisateur modernes et réactives.

Terminal

Interface en ligne de commande utilisée pour exécuter des commandes système ou spécifiques au projet.

URI (Uniform Resource Identifier)

Adresse unique qui identifie une ressource, comme une base de données ou un serveur, utilisée ici pour connecter MongoDB à l'application.

Vite

Gestionnaire de modules et outil de compilation utilisé pour optimiser les assets front-end dans Laravel.

VSCode (Visual Studio Code)

Environnement de développement intégré (IDE) utilisé pour coder, éditer et gérer le projet.

XAMPP Environnement de développement PHP incluant Apache, MySQL, PHP, et Perl, utilisé pour configurer le projet localement.

Réflexions initiale sur le sujet

1. Compréhension des Besoins et Objectifs du Projet

Le projet vise à développer une application web pour le Zoo Arcadia, permettant aux visiteurs de visualiser les animaux, leurs états, les services du zoo et les horaires d'ouverture. L'application doit également refléter les valeurs du zoo, être intuitive et facile à naviguer, et comporter des fonctionnalités spécifiques pour les vétérinaires, employés, et administrateurs.

2. Analyse des Exigences Fonctionnelles (US)

Les exigences fonctionnelles, ou User Stories (US), sont clairement définies:

US 1 - Page d'accueil: Présentation du zoo, ses habitats, services, animaux, et avis des visiteurs.

US 2 - Menu de l'application: Navigation intuitive avec accès rapide aux sections clés.

US 3 - Vue globale des services: Description et gestion des services offerts par le zoo.

US 4 - Vue globale des habitats: Détails sur les habitats et les animaux, incluant les rapports de santé des vétérinaires.

US 5 - Avis des visiteurs: Système de commentaires avec validation par les employés.

US 6 - Espace Administrateur: Gestion des utilisateurs, services, habitats, animaux, et visualisation des rapports.

US 7 - Espace Employé: Validation des avis, gestion des services et alimentation des animaux.

US 8 - Espace Vétérinaire: Gestion des rapports de santé des animaux et avis sur les habitats.

US 9 - Connexion: Authentification des administrateurs, vétérinaires, et employés.

US 10 - Contact: Formulaire de contact pour les visiteurs.

US 11 - Statistique de consultation des animaux: Suivi des consultations des animaux pour des analyses statistiques.

3. Choix Techniques

Pour répondre aux exigences, un ensemble de technologies doit être choisi judicieusement :

Front-end: React, tailwind.CSS pour une interface utilisateur réactive et moderne.

Back-end: Laravel

Communication back-end, front-end : Inertia.js

Base de données relationnelle: MySQL pour gérer les relations.

Base de données NoSQL: MongoDB pour stocker les statistiques de consultation. **Déploiement:** Hostinger

4. Architecture de l'Application

L'application suivra une architecture MVC (Model-View-Controller) pour une séparation claire des préoccupations, facilitant ainsi le développement:

- **Modèle:** Gère les données et la logique métier.
- **Vue:** Présente les données à l'utilisateur.
- **Contrôleur:** Interagit avec le modèle et la vue, répondant aux actions de l'utilisateur.

5. Sécurité et Performances

La sécurité est cruciale, surtout pour l'authentification et la gestion des données sensibles :

- **Authentification sécurisée:** Utilisation de HTTPS, salage et hachage des mots de passe.
- **Validation des entrées:** Pour prévenir les injections SQL et les scripts inter sites (XSS).

6. Déploiement et Gestion du Projet

Un déploiement efficace nécessite :

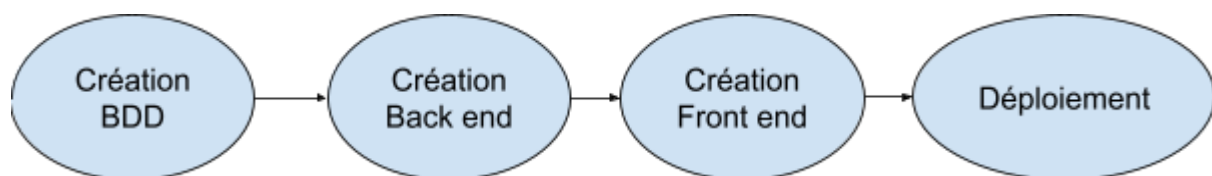
- **GitHub:** Pour la gestion du code source, avec des branches pour le développement et la production.
- **Kanban:** Utilisation de Trello pour la gestion de projet, avec des colonnes pour les fonctionnalités prévues, en cours, et terminées.

7. Livrables

Les livrables incluent :

- Code source sur un dépôt GitHub public.
- Liens vers les applications déployées.
- Documentation technique et utilisateur en PDF.
- Charte graphique avec les maquettes.
- Fichiers SQL pour la base de données.
- Manuel d'utilisation détaillé.
- Documentation de la gestion de projet, incluant les diagrammes et le modèle conceptuel de données.

8. Etapes développement



Configuration de l'environnement de travail

Voici un guide pour configurer l'environnement de travail pour le projet Arcadia sur un poste vierge, sous macOS, avec les besoins technologiques suivants : **Laravel**, **Inertia.js**, **Breeze**, **MySQL**.

Ce projet nécessitera également l'installation d'un IDE; (environnement de travail intégré) ainsi que d'un environnement de développement php tel que xaamp.

Télécharger xaamp

<https://www.apachefriends.org/fr/download.html>

1. Pré-requis : Installation de Homebrew

Homebrew est un gestionnaire de paquets très pratique pour installer facilement divers outils sur macOS.

<https://brew.sh/fr/>

2. Installation de PHP

Laravel nécessite PHP pour fonctionner. J'utilise Homebrew pour l'installer.

Installer PHP :

Dans le terminal du poste :

brew install php

Vérifier l'installation :

```
joakmann@MBP-de-Joakmann ~ % php -v
PHP 8.3.13 (cli) (built: Oct 22 2024 18:39:14) (NTS)
Copyright (c) The PHP Group
Zend Engine v4.3.13, Copyright (c) Zend Technologies
    with Zend OPcache v8.3.13, Copyright (c), by Zend Technologies
joakmann@MBP-de-Joakmann ~ %
```

3. Installation de Composer

Composer est le gestionnaire de dépendances pour PHP, nécessaire pour installer Laravel.

Installer Composer :

brew install composer

```
joakmann@MBP-de-Joakmann ~ % composer -v

Composer version 2.7.7 2024-06-10 22:11:12

Usage:
  command [options] [arguments]

Options:
  -h, --help                Display help for the given command. When no command is given display help for the list command
  -q, --quiet               Do not output any message
  -V, --version             Display this application version
  --ansi|--no-ansi         Force (or disable --no-ansi) ANSI output
  -n, --no-interaction     Do not ask any interactive question
  --profile                Display timing and memory usage information
  --no-plugins             Whether to disable plugins.
  --no-scripts            Skips the execution of all scripts defined in composer.json file.
  -d, --working-dir=WORKING-DIR If specified, use the given directory as working directory.
  --no-cache              Prevent use of the cache
  -v|vv|vvv, --verbose    Increase the verbosity of messages: 1 for normal output, 2 for more verbose output and 3 for debug

Available commands:
  about                Shows a short information about Composer
  archive              Creates an archive of this composer package
  audit                Checks for security vulnerability advisories for installed packages
  browse              [home] Opens the package's repository URL or homepage in your browser
  bump                Increases the lower limit of your composer.json requirements to the currently installed versions
  check-platform-reqs Check that platform requirements are satisfied
  clear-cache          [clearcache|cc] Clears composer's internal package cache
  completion           Dump the shell completion script
  config              Sets config options
  create-project       Creates new project from a package into given directory
  depends             [why] Shows which packages cause the given package to be installed
  diagnose            Diagnoses the system to identify common errors
  dump-autoload       [dumpautoload] Dumps the autoloader
  exec                Executes a vendored binary/script
  fund                Discover how to help fund the maintenance of your dependencies
  global              Allows running commands in the global composer dir ($COMPOSER_HOME)
  help                Display help for a command
  init                Creates a basic composer.json file in current directory
  install             [i] Installs the project dependencies from the composer.lock file if present, or falls back on the composer.json
  licenses            Shows information about licenses of dependencies
  list                List commands
  outdated            Shows a list of installed packages that have updates available, including their latest version
  prohibits          [why-not] Shows which packages prevent the given package from being installed
  reinstall           Uninstalls and reinstalls the given package names
  remove             [rm|uninstall] Removes a package from the require or require-dev
  require            [r] Adds required packages to your composer.json and installs them
  run-script         [run] Runs the scripts defined in composer.json
  search             Searches for packages
  self-update        [selfupdate] Updates composer.phar to the latest version
  show              [info] Shows information about packages
  status            Shows a list of locally modified packages
  suggests           Shows package suggestions
  update            [u|upgrade] Updates your dependencies to the latest version according to composer.json, and updates the composer.lock file
  validate           Validates a composer.json and composer.lock
```

4. Installation de Node.js et NPM

Laravel et Inertia.js requièrent Node.js pour gérer les dépendances front-end et compiler les assets.

Installer Node.js et NPM :

brew install node

```
joakmann@MBP-de-Joakmann ~ % node -v
v21.7.3
joakmann@MBP-de-Joakmann ~ %
```

```
joakmann@MBP-de-Joakmann ~ % npm -v
10.7.0
joakmann@MBP-de-Joakmann ~ %
```

5. Installation de MySQL

Le projet utilise MySQL comme base de données. Homebrew facilite également son installation.

Installer MySQL :

brew install mysql

6. Installation de Laravel

composer global require laravel/installer

```
[joakmann@MBP-de-Joakmann ~ % composer global require laravel/installer
Changed current directory to /Users/joakmann/.composer
./composer.json has been updated
Running composer update laravel/installer
Loading composer repositories with package information
Updating dependencies
Nothing to modify in lock file
Writing lock file
Installing dependencies from lock file (including require-dev)
Nothing to install, update or remove
Generating autoload files
18 packages you are using are looking for funding.
Use the `composer fund` command to find out more!
No security vulnerability advisories found.
Using version ^5.9 for laravel/installer
joakmann@MBP-de-Joakmann ~ %
```


7. Création d'un Nouveau Projet Laravel avec Breeze et Inertia.js

Créer un nouveau projet Laravel :

```
joakmann@MBP-de-Joakmann desktop % cd
joakmann@MBP-de-Joakmann ~ % source ~/.zshrc
joakmann@MBP-de-Joakmann ~ % laravel ~v
Laravel Installer 5.9.2

Usage:
  Command [options] [arguments]

Options:
  -h, --help            Display help for the given command. When no command is given display help for the list command
  -q, --quiet           Do not output any message
  -V, --version         Display this application version
  --ansi|--no-ansi     Force (or disable --no-ansi) ANSI output
  -n, --no-interaction Do not ask any interactive question
  -vv|--vv, --verbose  Increase the verbosity of messages: 1 for normal output, 2 for more verbose output and 3 for debug

Available commands:
  completion  Dump the shell completion script
  help       Display help for a command
  list       List commands
  new        Create a new Laravel application
joakmann@MBP-de-Joakmann ~ % laravel new arcadia-app

Laravel

Would you like to install a starter kit?
Laravel Breeze

Which Breeze stack would you like to install?
  o Blade with Alpine
  o Livewire (Volt Class API) with Alpine
  o Livewire (Volt Functional API) with Alpine
  > React with Inertia
  o Vue with Inertia
```

Une fois le stack sélectionné l'environnement s'installe et je peux utiliser la commande `cd arcadia-app` pour interagir avec mon application laravel dans VsCode.

```
vite v5.4.10 building for production...
✓ 352 modules transformed.
public/build/manifest.json          7.20 kB | gzip: 0.89 kB
public/build/assets/app-Cy8ptAJZ.css 33.51 kB | gzip: 6.50 kB
public/build/assets/InputLabel-CcjsXPdZ.js 0.21 kB | gzip: 0.18 kB
public/build/assets/GuestLayout-D-n35dAZ.js 0.50 kB | gzip: 0.33 kB
public/build/assets/PrimaryButton-h_NY9qrC.js 0.50 kB | gzip: 0.34 kB
public/build/assets/TextInput-BZ79hLGR.js 0.60 kB | gzip: 0.39 kB
public/build/assets/Dashboard-C6Brqj-K.js 0.64 kB | gzip: 0.37 kB
public/build/assets/Edit-_f_TkQ0w.js 1.12 kB | gzip: 0.52 kB
public/build/assets/ForgotPassword-OPi9HVfK.js 1.14 kB | gzip: 0.65 kB
public/build/assets/ConfirmPassword-DC98zqIZ.js 1.16 kB | gzip: 0.64 kB
public/build/assets/VerifyEmail-BmCciogk.js 1.26 kB | gzip: 0.70 kB
public/build/assets/ResetPassword-Ct26UijC.js 1.78 kB | gzip: 0.70 kB
public/build/assets/Login-BcUvFkGF.js 2.09 kB | gzip: 0.94 kB
public/build/assets/Register-DdXd4jEv.js 2.28 kB | gzip: 0.82 kB
public/build/assets/UpdateProfileInformationForm-D_hpJwMn.js 2.29 kB | gzip: 1.00 kB
public/build/assets/UpdatePasswordForm-CcLexKGM.js 2.34 kB | gzip: 0.86 kB
public/build/assets/ApplicationLogo-FD4qeMu0.js 3.12 kB | gzip: 1.32 kB
public/build/assets/AuthenticatedLayout-CjJwDVJe.js 5.83 kB | gzip: 1.90 kB
public/build/assets/transition-DJt30atv.js 14.60 kB | gzip: 5.77 kB
public/build/assets/Welcome-BcPTB80L.js 19.29 kB | gzip: 5.86 kB
public/build/assets/DeleteUserForm-wanUj9Gc.js 29.89 kB | gzip: 10.59 kB
public/build/assets/app-Bf3YzXn3.js 257.57 kB | gzip: 86.56 kB
✓ built in 1.07s

[INFO] Breeze scaffolding installed successfully.
[INFO] Application ready in [arcadia-app]. You can start your local development using:

+ cd arcadia-app
+ npm install && npm run build
+ composer run dev

New to Laravel? Check out our bootcamp and documentation. Build something amazing!

joakmann@MBP-de-Joakmann ~ % cd arcadia-app
joakmann@MBP-de-Joakmann arcadia-app %
```

Installer Inertia.js : Inertia.js permet d'intégrer des applications Vue.js avec Laravel.

```
joakmann@MBP-de-Joakmann arcadia-app % composer require inertiajs/inertia-laravel

./composer.json has been updated
Running composer update inertiajs/inertia-laravel
Loading composer repositories with package information
Updating dependencies
Nothing to modify in lock file
Writing lock file
Installing dependencies from lock file (including require-dev)
Nothing to install, update or remove
Generating optimized autoload files
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi

 INFO  Discovering packages.

inertiajs/inertia-laravel ..... DONE
laravel/breeze ..... DONE
laravel/pail ..... DONE
laravel/sail ..... DONE
laravel/sanctum ..... DONE
laravel/tinker ..... DONE
nesbot/carbon ..... DONE
nunomaduro/collision ..... DONE
nunomaduro/termwind ..... DONE
pestphp/pest-plugin-laravel ..... DONE
tightenco/ziggy ..... DONE

86 packages you are using are looking for funding.
Use the `composer fund` command to find out more!
> @php artisan vendor:publish --tag=laravel-assets --ansi --force

 INFO  No publishable resources for tag [laravel-assets].

No security vulnerability advisories found.
Using version ^1.3 for inertiajs/inertia-laravel
```

Je tape ensuite **npm install** dans le terminal pour finaliser l'installation

```
joakmann@MBP-de-Joakmann arcadia-app % npm install

added 1 package, and audited 243 packages in 582ms

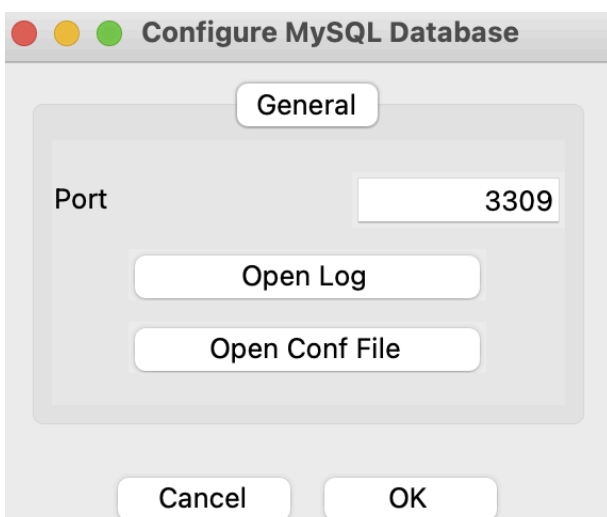
53 packages are looking for funding
run `npm fund` for details

found 0 vulnerabilities
joakmann@MBP-de-Joakmann arcadia-app %
```

8. Configuration de l'Environnement du Projet

Dans le fichier .env, configurer la base de données MySQL :

Mon serveur local se situe sur le port 3309 je dois donc modifier le fichier **.env** pour faire correspondre les ports car le port initial se trouve sur 3306.



```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3309
DB_DATABASE=arcadia_zoo
DB_USERNAME=root
DB_PASSWORD=
```

Fichier **.env** modifié

Je créer une première migration avec la commande :

php artisan make:migration create_roles_table

et en effectuant la commande **php artisan migrate**

```
joakmann@MBP-de-Joakmann arcadia-app % php artisan migrate
WARN The database 'arcadia_app' does not exist on the 'mysql' connection.
Would you like to create it? ☒ Yes / ☐ No
```

Je crée ma base de données relationnelle.

Cependant le standard en matière de pratique impose la création de la bdd en sql je peux donc effectuer le processus suivant :

Je tape dans le terminal :

mysql

puis, j'effectue la commande :

CREATE DATABASE arcadia_app;

Une fois la base de données crée je peux voir les tables présentes dans MySql avec les commandes suivantes :

USE arcadia_app;

Le terminal affiche alors *"database changed"*

SHOW tables;

Mysql affiche les tables présentes dans la base de donnée arcadia-app.

Une fois ce processus effectué, mon environnement de travail comprend à ce stade un environnement php mis à jour, un projet laravel/react initialisé avec breeze et livewire ainsi qu'une base de donnée relationnelle avec **mysql**.

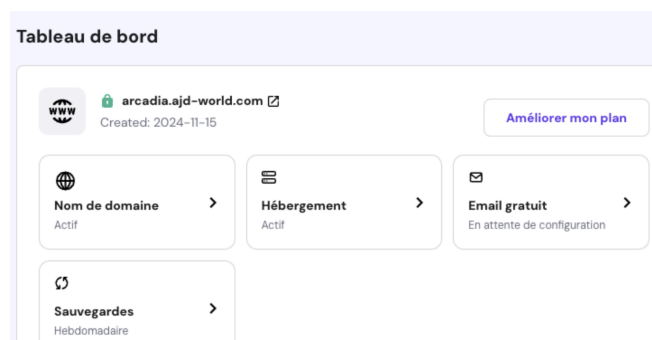
Guide de Déploiement et Configuration

Configuration Hostinger

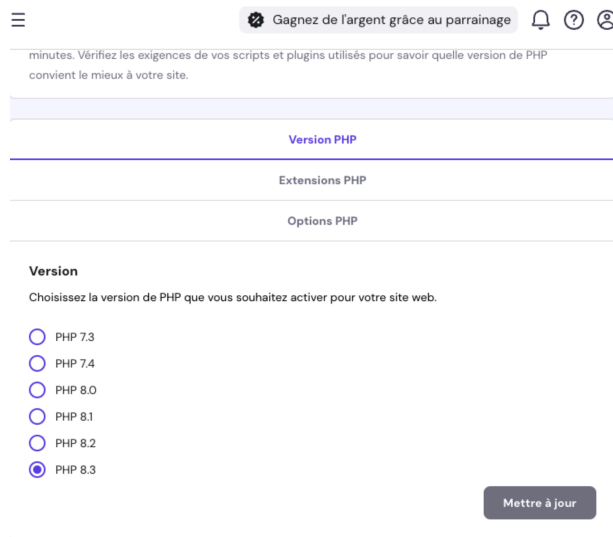
Je me connecte à mon compte Hostinger et accède à la section **Domaines**.

Je configure un domaine principal, par exemple `ajd-world.com`, en suivant les instructions d'achat et d'installation fournies par Hostinger. Ensuite, je crée un sous-domaine pour mon application, tel que `arcadia.ajd-world.com`.

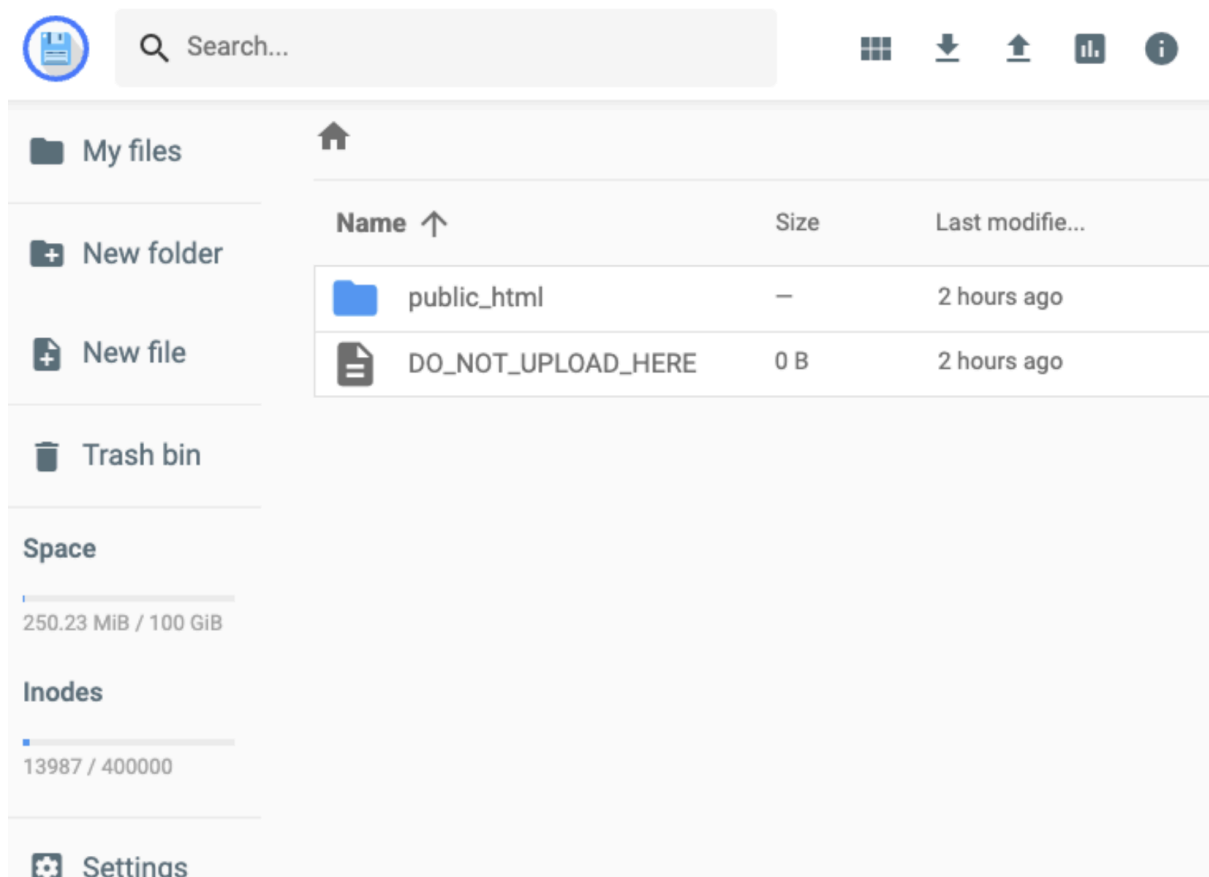
Dans les paramètres du domaine, je m'assure que les certificats SSL sont actifs pour sécuriser les connexions. Une fois le sous-domaine configuré, un dossier `public_html/arcadia` est automatiquement créé. C'est ici que je vais téléverser les fichiers de mon projet.



Je m'assure également que la version de PHP utilisée est compatible avec Laravel. Je vérifie cela dans **Hébergements > Paramètres PHP**, et j'active la dernière version disponible si nécessaire.



Pour téléverser mes fichiers, j'utilise le gestionnaire de fichiers intégré à Hostinger,



Tous les fichiers de mon projet local, précompilés en production avec la commande `npm run build` dans mon éditeur de code, sont copiés dans le dossier correspondant au sous-domaine.

Configuration Atlas

Je me connecte à MongoDB Atlas pour créer une base de données cloud.

Dans Atlas, je crée un cluster MongoDB et configure une base de données nommée `arcadia_zoo_mongo`.

Dans les paramètres du cluster, je m'assure que l'adresse IP de mon serveur Hostinger (92.113.24.65) est autorisée à accéder à la base de données.

Je fais cela dans **Network Access > IP Whitelist**, où j'ajoute cette adresse IP avec le label correspondant. Je crée ensuite un utilisateur pour la base de données avec les identifiants suivants :

- **Nom d'utilisateur** : josearcadia
- **Mot de passe** : arcadiazoo

Je copie l'URI de connexion fourni par Atlas et le personnalise pour inclure mes informations d'utilisateur.

L'URI final est :

<mongodb+srv://josearcadia:arcadiazoo@cluster0.0croj.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0>

Configuration du Fichier .env

Je modifie mon fichier .env pour refléter les paramètres de production et les connexions aux bases de données.

```
.env
X

.env

1  # Paramètres généraux de l'application
2  APP_NAME=Arcadia
3  APP_ENV=production
4  APP_KEY=base64:fPn99C4Idz7ogTDQy8MZBW5Xd1HvFpcMrI5QX24v2HI=
5  APP_DEBUG=false
6  APP_URL=https://arcadia.ajd-world.com/
7
8  # Localisation
9  APP_LOCALE=en
10 APP_FALLBACK_LOCALE=en
11 APP_FAKER_LOCALE=en_US
12
13 # Connexion à la base de données MySQL
14 DB_CONNECTION=mysql
15 DB_HOST=193.203.168.141
16 DB_PORT=3306
17 DB_DATABASE=u724324663_arcadia_zoo
18 DB_USERNAME=u724324663_adminjose # Nom d'utilisateur MySQL
19 DB_PASSWORD=Arcadiazoo13 # Mot de passe MySQL
20
21 # Connexion à la base de données MongoDB
22 DB_CONNECTION_MONGODB=mongodb
23 MONGODB_URI=mongodb+srv://josearcadia:arcadiazoo@cluster0.0croj.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0
24
25 # Autres configurations (Logs, sessions, etc.)
26 LOG_CHANNEL=stack
27 SESSION_DRIVER=database
28 SESSION_LIFETIME=120
29
30 # Paramètres de cache, messagerie et autres services
31 CACHE_STORE=database
32 MAIL_MAILER=log
33 MAIL_HOST=127.0.0.1
34 MAIL_PORT=2525
35 MAIL_USERNAME=null
36 MAIL_PASSWORD=null
37 MAIL_ENCRYPTION=null
38 MAIL_FROM_ADDRESS="hello@example.com"
39 MAIL_FROM_NAME="${APP_NAME}"
```

Pour activer la configuration en production, je nettoie et mets à jour le cache de Laravel :

Pour cela je dois désormais utiliser un terminal ssh

Je me connecte au terminal ssh depuis hostinger

Je colle ssh -p 65002 u724324663@92.113.24.65 dans celui ci

```
Last login: Mon Nov 18 17:39:34 on ttys026
joakmann@MBP-de-Joakmann ~ % ssh -p 65002 u724324663@92.113.24.65
u724324663@92.113.24.65's password:
#####

Vendor: Dell Inc.
Model: PowerEdge R6615

IPMI IP address: 10.4.0.255

IPMI MAC address: 28:00:af:f2:78:90

#####
Last failed login: Mon Nov 18 07:00:41 UTC 2024 from 78.126.52.203 on ssh:notty
There were 3 failed login attempts since the last successful login.

      HHHH          HHHH
      HHHHHHHH      HHHHHHHH
      HHHHHHHH      HHHHHHHH
      HHHHHHHH      HHHHHHHH
      HHHHHHHHHHHHHHHHHHHHH HHHHH
      HHHHHHHHHHHHHHHHHHHHH
      HHHH  HHHHHHHHHHHHHHHHHHH
      HHHHHHHH      HHHHHHHH
      HHHH HHHH      HHHHHHHH
      HHHHHHHH      HHHHHHHH
      HHHH          HHHH

Welcome back! The time now is 16:42 UTC
Server load: 11.79, 12.23, 12.18

Link to hPanel:
https://hpanel.hostinger.com/

[[u724324663@fr-int-web1584 ~]$ cd domains/arcadia.ajd-world.com/public_html
```

Je place tout le répertoire de mon projet avec la commande :

1. Erreur bad auth : authentication failed
2. Organisation des Fichiers sur le Serveur
3. Suppression du Fichier .hot
4. Configuration du Fichier .htaccess à la Racine
5. Configuration du Fichier .htaccess dans le Dossier Public
6. Configuration du Fichier .env
7. Configuration de MongoDB Atlas

1. Erreur bad auth : authentication failed

Description de l'Erreur :

Lors de la tentative de récupération des statistiques depuis MongoDB Atlas, une erreur bad auth : authentication failed a été enregistrée dans les logs. Cette erreur survient lorsque les informations d'identification pour se connecter à MongoDB (URI, utilisateur ou mot de passe) sont incorrectes.

Causes Identifiées :

URI MongoDB incorrecte :

Le fichier .env contenait une mauvaise configuration de l'URI MongoDB.

Problème avec le mot de passe ou l'encodage du caractère spécial.

Adresse IP non autorisée :

L'IP du serveur de production (92.113.24.65) n'avait pas été ajoutée aux règles de pare-feu de MongoDB Atlas.

Absence de l'utilisateur ou rôle insuffisant :

L'utilisateur josearcadia sur MongoDB pouvait manquer ou avoir un rôle inadéquat pour accéder à la base de données arcadia_zoo_mongo.

Solutions Apportées :

Correction de l'URI MongoDB :

Modification du fichier .env pour inclure l'URI corrigée avec les bonnes informations d'identification :

MONGODB_URI=mongodb+srv://josearcadia:arcadiazoo@cluster0.0croj.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0

Ajout de l'IP au pare-feu MongoDB Atlas :

Accès à MongoDB Atlas > Network Access.

Ajout de l'adresse IP 92.113.24.65 pour permettre les connexions depuis le serveur Hostinger.

Vérification de l'utilisateur MongoDB :

Dans MongoDB Atlas > Database Access, vérification que l'utilisateur josearcadia existe avec les permissions adéquates sur la base de données arcadia_zoo_mongo.

Mise à jour des caches Laravel :

Pour appliquer les nouvelles configurations, exécution des commandes suivantes dans le terminal SSH :

```
php artisan config:cache  
php artisan route:cache
```

Liens Utilisés pour Résolution :

<https://www.mongodb.com/docs/atlas/security/ip-access-list/>

<https://laravel.com/docs/11.x/configuration#environment-configuration>

Contexte Suivant : Avec ces corrections, l'application a pu se connecter à MongoDB Atlas, résoudre l'erreur d'authentification, et récupérer les statistiques des animaux et habitats.

2. Erreur : Organisation des fichiers sur le serveur

Problème :

Le dossier `public_html` contenait plusieurs fichiers et dossiers redondants (`arcadiaZoo`, `public_html`, etc.), ce qui compliquait l'organisation et le déploiement.

Action :

Suppression du dossier `public_html` existant, après avoir sauvegardé les fichiers nécessaires.

Renommage du dossier `arcadiaZoo` en `public_html` pour aligner avec la structure standard des serveurs Hostinger.

Moyen Utilisé :

Utilisation du gestionnaire de fichiers intégré à Hostinger pour gérer les fichiers/dossiers.

Commandes SSH pour effectuer des vérifications de permissions et s'assurer que les changements sont pris en compte.

3. Erreur : Suppression du fichier hot.

Le fichier hot est créé lorsque le serveur de développement de Vite est actif (`npm run dev`). Il permet au navigateur de pointer vers ce serveur local pour recharger les modifications en temps réel. En production, cette configuration est remplacée par un fichier build généré par `npm run build`, qui contient les assets compilés et optimisés.

Problème :

Présence du fichier hot inutilisé qui pouvait causer des conflits ou de la confusion.

Action :

Suppression du fichier hot via le gestionnaire de fichiers, car il n'était pas nécessaire pour le déploiement en production.

Moyen Utilisé :

Inspection et nettoyage manuel des fichiers dans `public_html`.

4. Erreur : Configuration du Fichier `.htaccess` à la Racine

Problème :

Le fichier .htaccess dans le dossier public était mal configuré, empêchant le routage correct.

Solution :

Création et ajout des règles standards Laravel pour le fichier .htaccess :

Liens Utilisés :

<https://github.com/laravel/laravel/blob/11.x/public/.htaccess>

5. Erreur : Configuration du Fichier .htaccess dans le Dossier Public

Le fichier .htaccess situé dans le dossier public était mal configuré, ce qui empêchait Laravel de gérer correctement le routage des requêtes en production. Cela se traduisait par une erreur telles que des pages non trouvées (404).

Problème :

Le fichier .htaccess dans le dossier public est mal configuré, empêchant le routage correct.

Solution :

Création et ajout des règles standards Laravel pour le fichier .htaccess :

<https://github.com/laravel/laravel/blob/11.x/public/.htaccess>

Laravel utilise un **Front Controller Pattern**, où toutes les requêtes passent par le fichier index.php. Ce fichier agit comme un point d'entrée unique pour :

- Charger l'application.
- Gérer les routes définies.
- Renvoyer des réponses au client.

Avec cette configuration sur Hostinger, toutes les requêtes passent correctement par public/index.php, et Laravel gère le routage sans problème.

6. Erreur : Configuration du Fichier .env

Problème :

Mauvaise configuration du fichier .env pour la connexion à MySQL et MongoDB, et environnement incorrect (APP_ENV non défini comme production).

Solution :

Mise à jour du fichier .env avec des paramètres corrects :

```
public_html > .env
1 # Paramètres généraux de l'application
2 APP_NAME=Arcadia
3 APP_ENV=production
4 APP_KEY=base64:fPn99C4Idz7ogTDQy8MZBW5XdlHvFpcMrISQXZ4v2HI=
5 APP_DEBUG=true
6 APP_URL=https://arcadia.ajd-world.com/
7
8 # Localisation
9 APP_LOCALE=en
10 APP_FALLBACK_LOCALE=en
11 APP_FAKER_LOCALE=en_US
12
13 # Connexion à la base de données MySQL
14 DB_CONNECTION=mysql
15 DB_HOST=193.203.168.141
16 DB_PORT=3306
17 DB_DATABASE=u724324663_arcadia_zoo
18 DB_USERNAME=u724324663_adminjose # Nom d'utilisateur MySQL
19 DB_PASSWORD=Arcadiazoo13 # Mot de passe MySQL
20
21 # Connexion à la base de données MongoDB
22 DB_CONNECTION_MONGODB=mongodb
23 MONGODB_URI=mongodb+srv://josearcadia:arcadiazoo@cluster0.0croj.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0
24
```

Liens Utilisés :

<https://laravel.com/docs/11.x/configuration#environment-configuration>

7. Erreur : Configuration de MongoDB Atlas

Problème :

L'IP du serveur de production n'était pas autorisée dans les règles de pare-feu de MongoDB Atlas, ce qui provoquait des erreurs bad auth : authentication failed.

Solution :

Ajout de l'IP du serveur 92.113.24.65 dans les règles de sécurité de MongoDB Atlas :

1. Accéder à MongoDB Atlas > Network Access.
2. Ajouter l'adresse IP de production.
3. Tester la connexion.