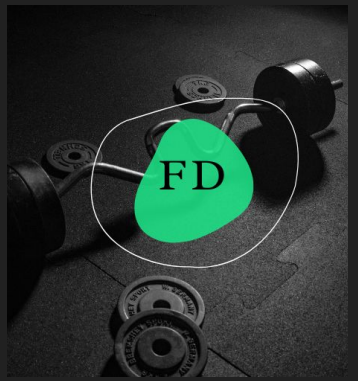
Présentation Oral
pour le Titre Professionnel

Développeur Web et Web Mobile

GIACOMETTI Virginie

Plan

Introduction



FitnessDrive est une grande marque de salle de sport et souhaite la création d'une interface simple à destination de ses équipes. Eux seuls pourront gérer les droits d'accès de ses franchisés et leurs structures.

Lorsqu'une salle de sport ouvre et prend la franchise de cette marque, on lui donne accès à un outil de gestion en ligne.

Depuis cette interface les franchisés pourront visualiser leurs options activées et les structures qu'ils possèdent.

Depuis leur espace, les employés de la marque peuvent ajouter ou modifier des franchisés, leurs structures et leurs différentes options proposées par la société.

L'objectif de ce projet est d'analyser les besoins, les fonctionnalités et de développer une application web avec une interface cohérente et ergonomique.

2. Maquettage

a. Cibles

b. Besoins / Fonctionnalités

c. Users Stories

e. Charte Graphique

a. Les Cibles

Trois utilisateurs :

Client => équipe technique qui ont un accès au panel d'administrateur

Le franchisé => accès son compte et sa/ses structures en lecture

La structure => accès à son compte en lecture

Afin de mieux comprendre les attentes du client, j'ai créé des personas.

Administrateur de Fitness Drive

Nom : FitCat
Prénom : Julie
Age: 28 ans
Situation social : Salariée en CDI
Adresse : Montpellier

Personnalité : Sociale, rigoureuse

Profil : Elle travaille dans la Société depuis 8 ans. Apprécie la relation direct avec les utilisateurs ainsi que sa capacité à résoudre leurs problèmes

Utilisation du produit : Elle se charge de créer ou de modifier les nouvelles franchises avec leurs structures et leurs permissions

Objectifs/Problèmes résolus par le produits : Elle apprécie la simplicité de la saisie de données et valorise une ergonomie efficace lui permettant de faire son travail sans perte de temps

Créé sur Canva

Voir annexe p.2

b. Les besoins /Les Fonctionnalités

Gérer les droits d'accès à l'application (Système d'Authentification)

Développer un panel d'administration

Interface simple et facile à prendre en main

Fonctionnalités

Général => Se connecter

Pour une Franchise :

- Visualiser le détail de sa franchise
- Visualiser le détail de sa/ses structures
- Contacter l'équipe avec une page contact

Pour une Structure :

- Visualiser le détail de sa structure
- Contacter l'équipe avec une page contact

Pour le panel d'admin :

- Ajouter un Utilisateur
- Ajouter une Franchise
- Ajouter une Structure
- Ajouter une permission
- Ajouter une permission à une structure
- Désactiver/Activer une Franchise, une structure ou une permission
- Recherche dynamique

c. Les Users Stories

Représentation des fonctionnalités sous forme d'User Story en y intégrant les personas. Elles sont claires et compréhensibles de tous et elles sont réalisables en un seul sprint.

Classer par thèmes:

A. Gestion des Utilisateurs

Description des fonctionnalités qu'un administrateur pourra effectuer.

B. Gestion des Franchises

Description des droits d'accès à application par une franchise

C. Gestion des Structures

Description des droits d'accès à application par une structure

Titre de l'us : US2 – Création de Compte de Connexion
En tant que, Julie **Je souhaite** créer un nouvel utilisateur **Afin que** le nouvel utilisateur accède à son compte
Règles métier 1. L'E-mail de la franchise ou de la structure doit être unique 2. Le mot de passe doit faire plus de 8 caractères
Critères d'acceptation **Scénario 2 :** Création d'un compte de connexion **Étant donné que,** je possède un compte utilisateur pour Julie **Lorsque**, j'accède à la page « Admin » /admin **Et que** je clique sur « Créer un compte utilisateur » **Et que** je saisis son nom dans le champ « Nom du Gérant » **Et que** je saisis son E-mail dans le champ « E-mail » **Et que** je saisis son Mot de Passe dans le champ « Mot de Passe » **Et que** je sélectionne son rôle « Franchise » ou « Structure » **Et que** je clique sur le bouton « Ajouter » **Alors** j'ai créé un nouvel utilisateur **Et** un E-mail est envoyé automatiquement à ce nouvel utilisateur avec ses identifiant de connexion

d. Charte Graphique

Élaboration du squelette de l'application.

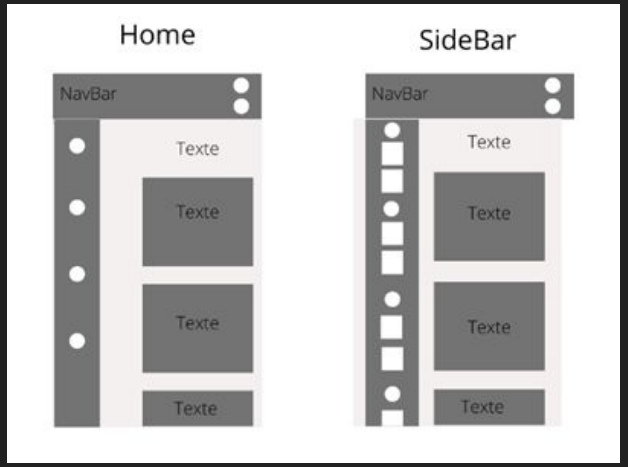
Les Wireframes permettent :

- Définir l'organisation général des blocs
- Matérialiser la navigation
- Orienter ma réflexion sur le design

Pour l'interface, je me suis inspiré du design Dark mode, Pour la navigation avec des icônes pour plus de faciliter.

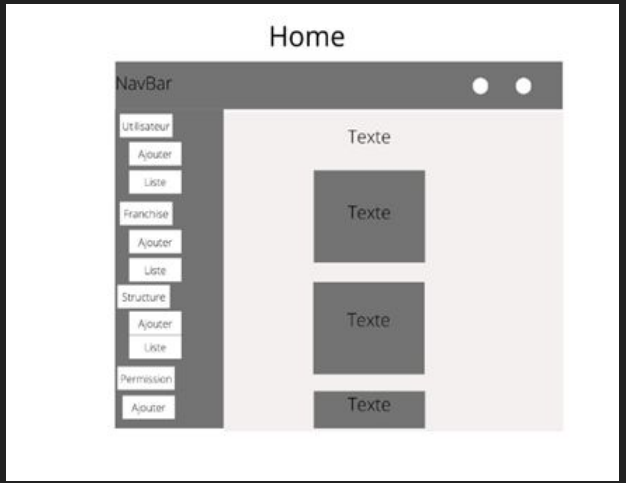
Pour les clients, des images en fond, des cartes pour le contenu avec des couleurs vives.

Pour les admins, le fond est blanc, des cartes pour le contenu avec des couleurs vives.

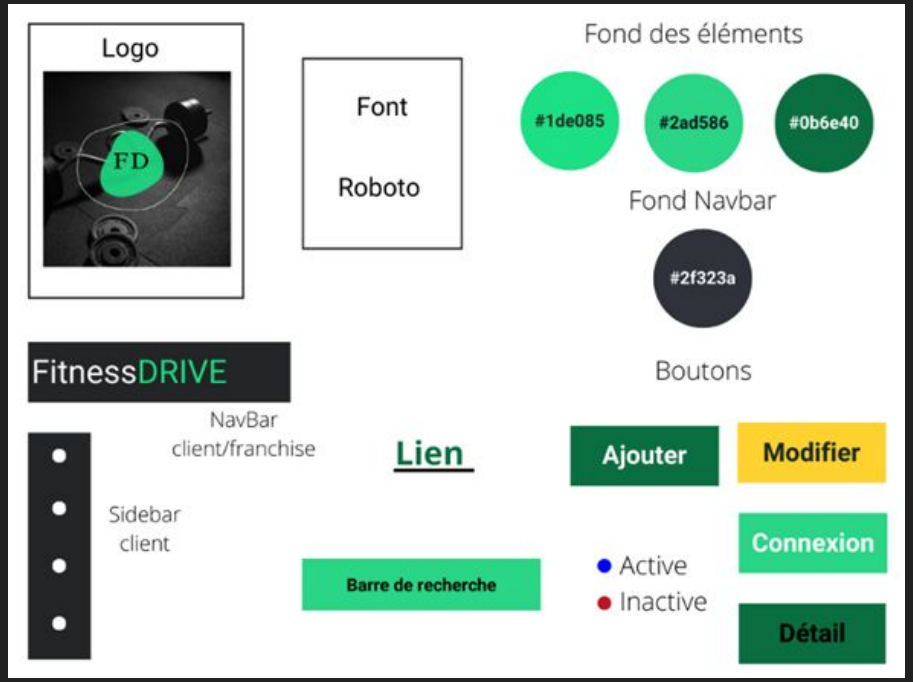


Mobile

Desktop

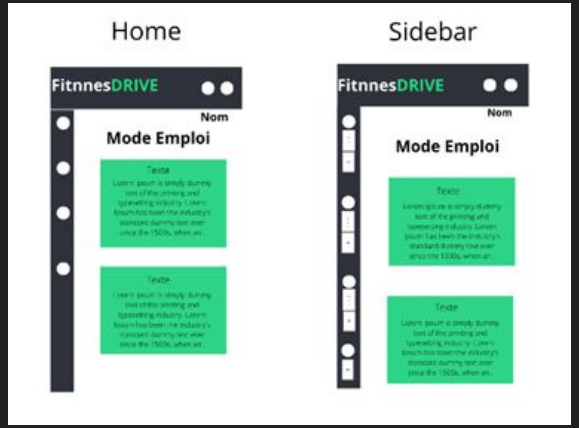


Valider les aspects visuels en créant une planche de style

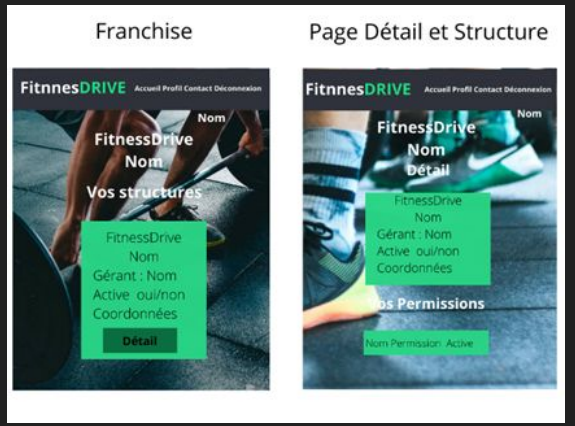


Transformation des Wireframes en maquette graphique les Mockup

Mobile



Desktop



Annexe p.14 à 16

Créé sur Canva

3. Réalisation du projet

a. Diagrammes

b. Méthode Merise

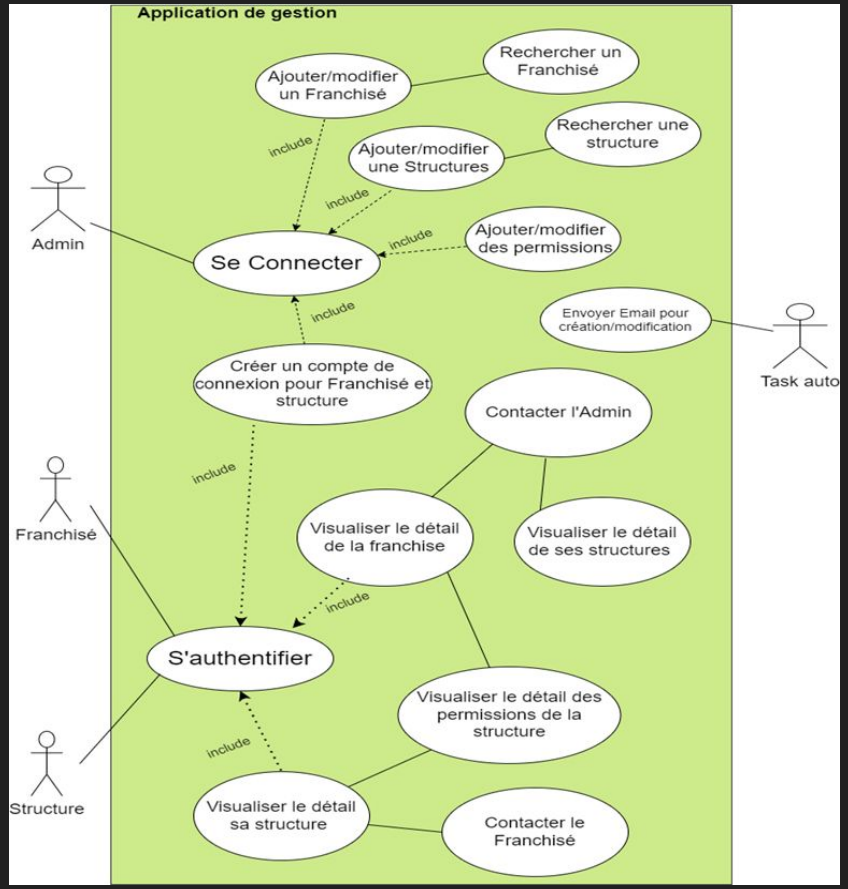
c. Backend

e. FrontEnd

a. Diagrammes

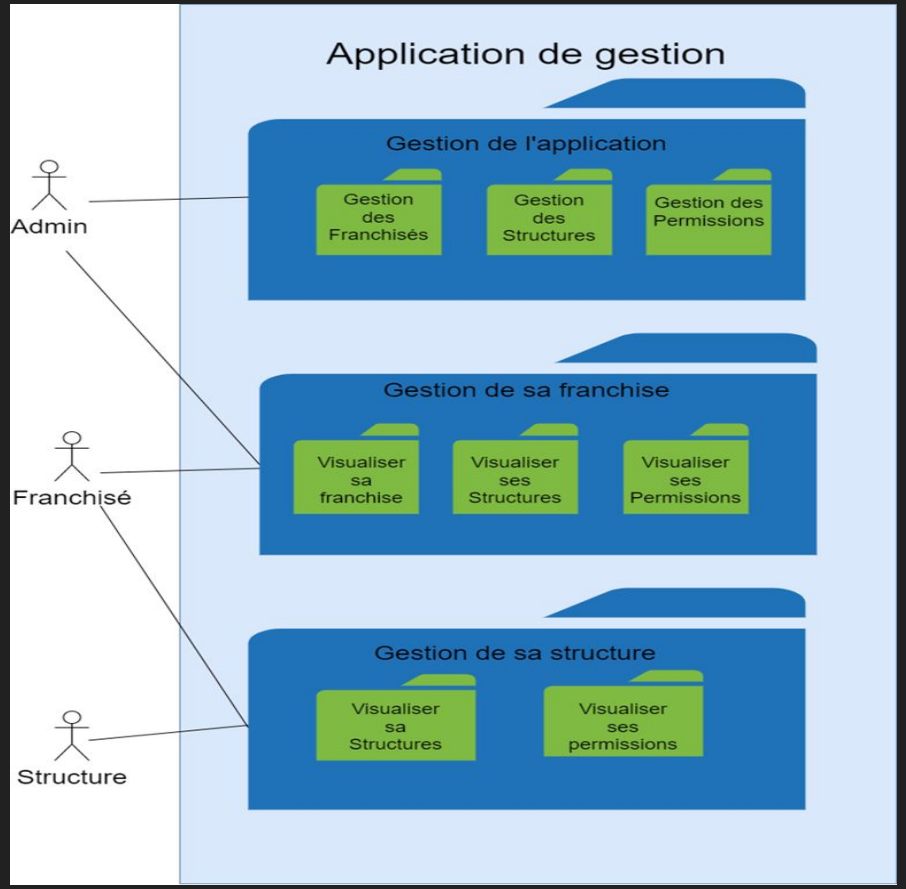
1. Diagrammes

Cas d'utilisation



Créé sur draw.io

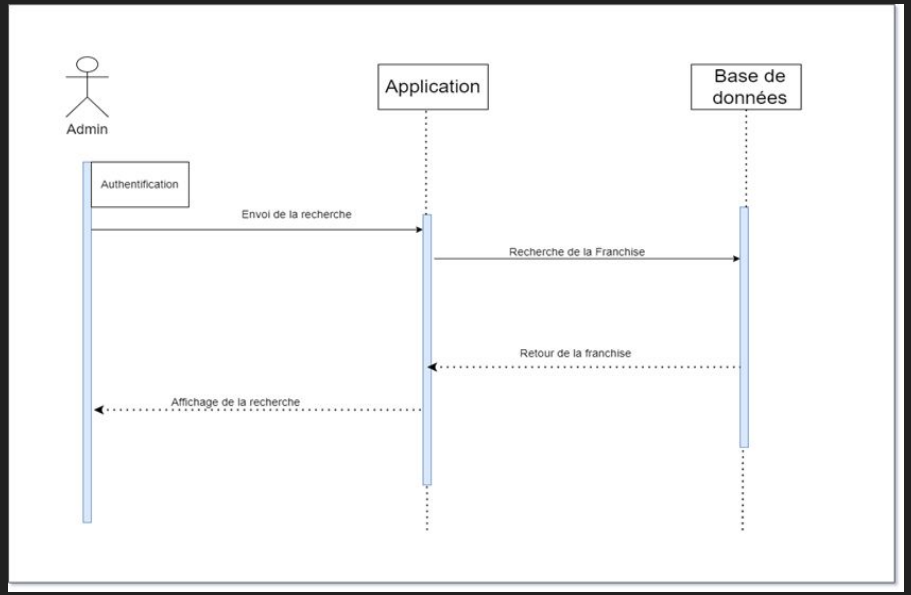
Diagramme de Package



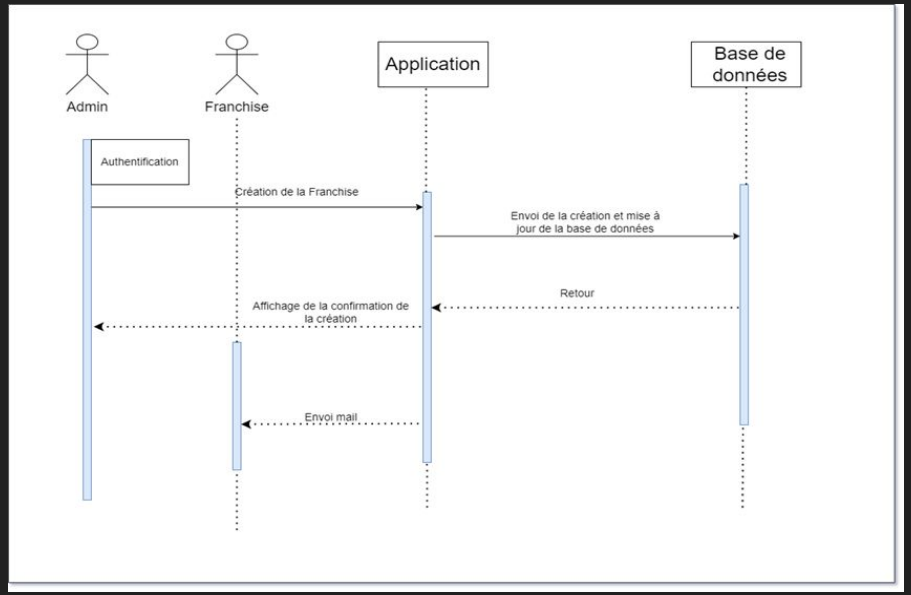
Annexe p.17

Diagrammes de Séquences

US9 – Recherche dynamique de la franchise



US3 – Création de la Franchise



Créé sur draw.io

Annexe p.18

b. Méthode Merise

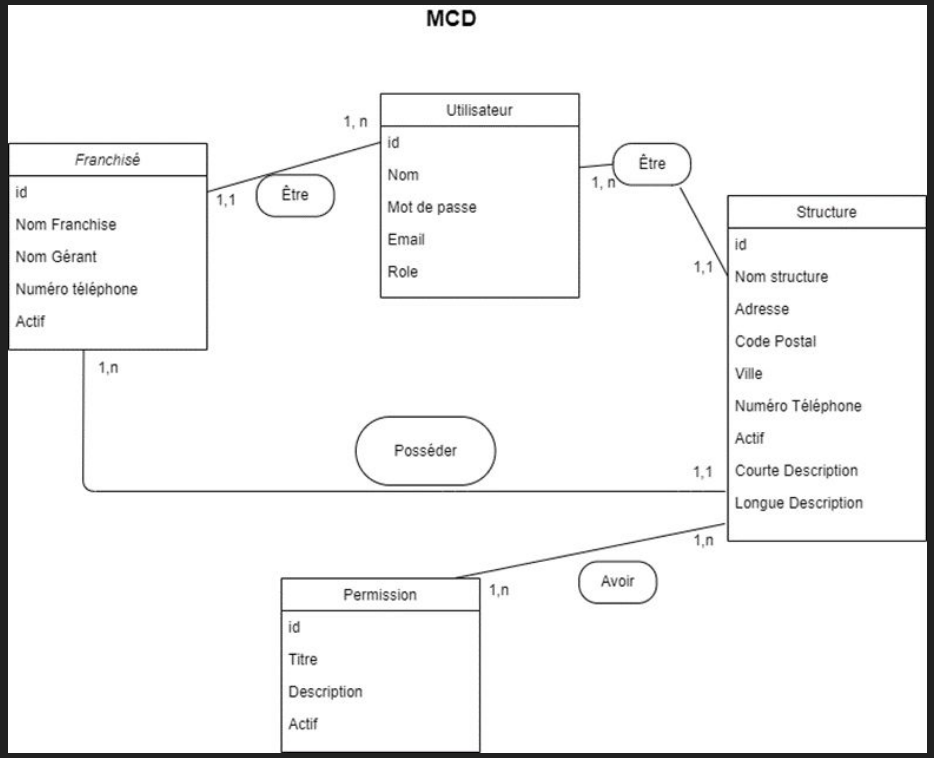
2. Méthode Merise

Permet une analyse fine pour la création de la base de données. Cela se fait par étapes pour obtenir un modèle de données prêt pour la base de données de notre application. Ça permet d'avoir une base de données bien construite et qui ne sera pas ou peu modifiée.

MCD - Modèle Conceptuel de données

Détermine :

- les entités
- les attributs des entités,
- les identifiants des entités,
- les associations des entités,
- les attributs des associations,
- les cardinalités des associations,
- Faire la représentation graphique.



Créé sur draw.io

Annexe p.19

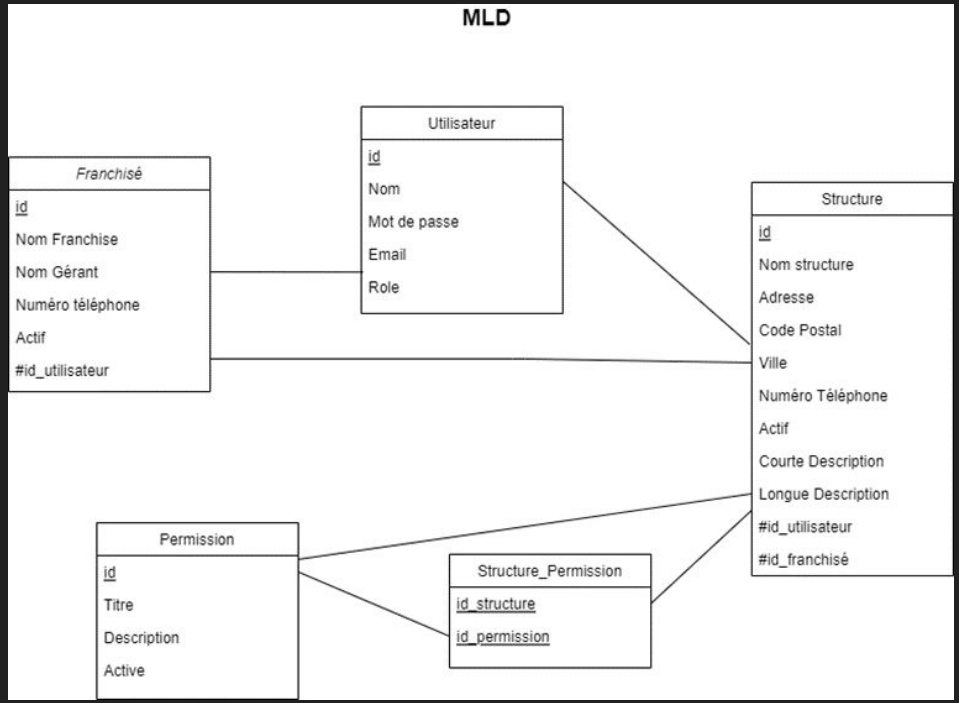
MLD - Modèle Logique de données

Une fois le schéma conceptuel de votre base de données effectué, il faudra établir une logique entre les différentes tables. Le Modèle Logique de Données (MLD) est un modèle extrait à partir du MCD.

Définit : Les Tables

Les clés Primaires

Les clés étrangères

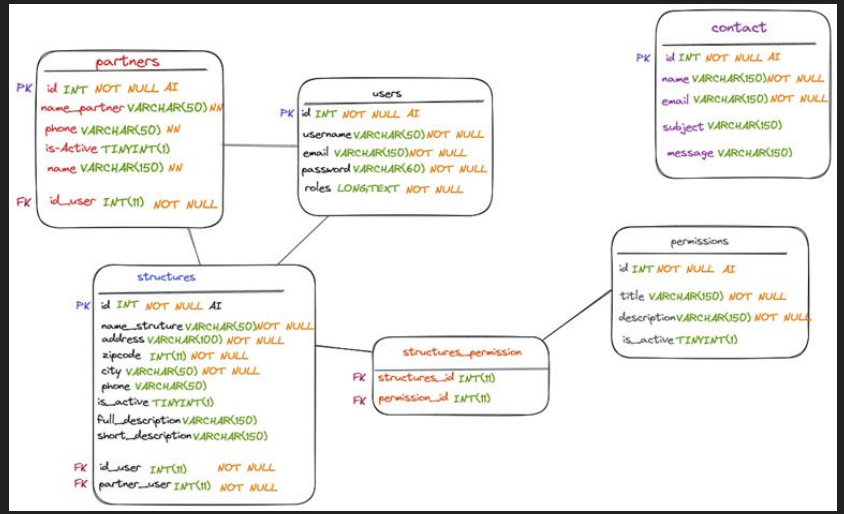


Créé sur draw.io

Annexe p.19

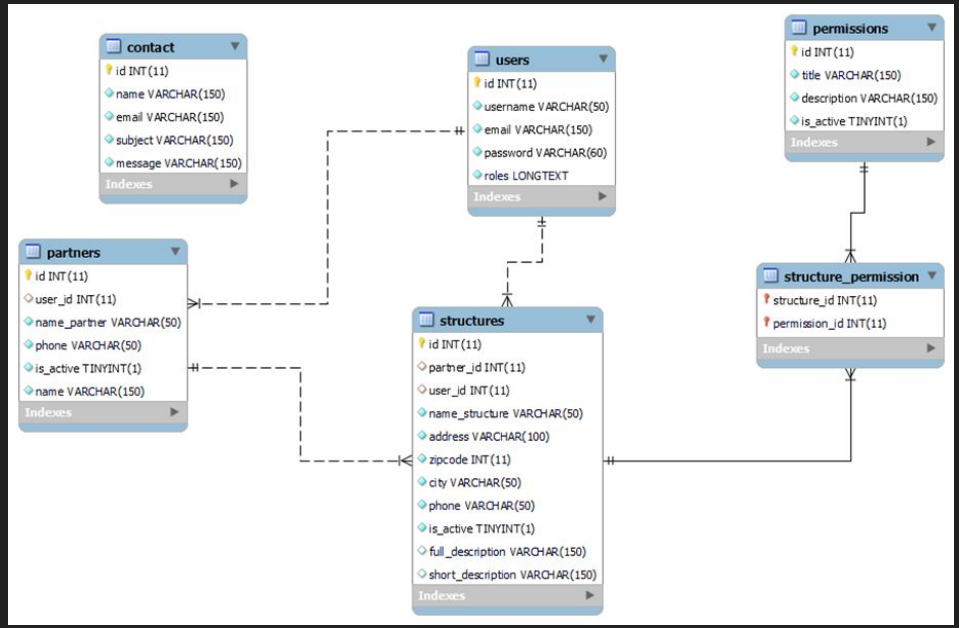
MPD - Modèle Physique de données

Dernière étape avant l'implémentation réelle de la base de données. Le MPD permet d'établir la manière dont le système sera mis en place et permet de préciser les systèmes de stockage employés. Il s'agit donc à cette étape de préparer l'implémentation d'un SGBD.



Créé sur draw.io

Annexe p.20



Export de ma BDD

c. Back-end

a. Installation

b. Base de données

c. Controller

e. Panel d'Admin

f. Service Mail

a. Installation

Ce projet a été développé avec le pattern MVC, qui permet de bien structurer le code source et de le découper en trois parties avec les technologies comme

- Doctrine pour ORM (**M**odel)
- Le moteur de template Twig (**V**iew)
- Le framework Symfony et PHP (**C**ontroller)

Initialisation du Projet

- ★ Installation de symfony
- ★ Initialiser un dépot Github

Configuration du fichier .env.local qui stockera toutes mes variables d'environnement comme :

```
APP_ENV=dev

APP_SECRET=################
```

Vérification:

```
symfony check:security
```

b. Base de Données

Configurer ma variable dans .env.local

DATABASE_URL="mysql://identifiant@localhost/fitness_drive?serverVersion=8&charset=utf8mb4"

Création de ma base de données soit :

- Sur phpMyAdmin en sql.
- Avec la commande :

php bin/console doctrine:database:create

Installation de ORM avec composer

```
-- CREATION dataBase and table

DROP SCHEMA IF EXISTS fitness_drive;

CREATE DATABASE IF NOT EXISTS cinema ;
```

```
DROP TABLE IF EXISTS `users`;

CREATE TABLE users

(

  id int(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,

  email char(150) NOT NULL UNIQUE,

  password char(60) NOT NULL,

  username varchar(50) NOT NULL,

  roles LONGTEXT NOT NULL

) engine=INNODB;
```

Création des entitées avec symfony

php bin/console make:entity qui génère:

- L'entité liée à la table, les getters et les setters pour récupérer ou modifier les données.
- Repository pour effectuer des requêtes liées à l'entité.

La table partners à une relation de OneToMany avec structure

Création de l' entitées User et de l'authentification

php bin/console make:user

symfony console make:auth

Définir les relations

Mettre à jour la base de données :

- **php bin/console make:migration**
- **php bin/console doctrine:migrations:migrate.**

```php
namespace App\Entity;
use Doctrine\ORM\Mapping as ORM;
#[UniqueEntity(fields:'namePartner', message:'Le nom que vous avez indiqué existe déjà')]
class Partner
{
  #[ORM\Column(length: 50, unique:true)]
  private ?string $namePartner = null;
  public function getNamePartner(): ?string
  {... }
  public function setNamePartner(string $namePartner): self
  {... }
```

```php
...
class PartnerRepository ...
public function countPartners()
  {
    return $this->createQueryBuilder('p')
            ->select('COUNT(p.id)')
            ->getQuery()
            ->getSingleScalarResult();
  }
...
```

c. Les controllers

Chaque entité possède son controller

php bin/console make:controller

Contient la logique de notre Application

Reçoit une requête et retourne une réponse

Récupérer des données

Configurer des routes

Transmet à la vue

```php
namespace App\Controller;
use App\Entity\Partner;
use App\Entity\Structure;
use App\Service\MailService;
use App\Repository\PartnerRepository;
use Doctrine\ORM\EntityManagerInterface;
...
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Security;

#[Route('/partner')]
class PartnerController extends AbstractController
{
   #[Route('/show/{id<\d+>}', name: 'app_partner_show')]
   public function index(ManagerRegistry $doctrine, int $id, Partner $partner,
PartnerRepository $partnerRepo): Response {
      $repositoryPartner = $doctrine->getRepository(Partner::class);
      $partner = $repositoryPartner->find($id);
      $structures= $partner->getStructures();
      $repositoryS = $doctrine->getRepository(Structure::class);
      $structure = $repositoryS->find($id);
      $permissions = $structure->getPermissions();
         return $this->render('partner/index.html.twig',[
            'partner'=> $partner,
            'structures' => $structures,
            'permissions'=> $permissions,
         ]);}...
```

e. Le panel d'administrateur

AdminController contient les requêtes pour accéder aux données de chaque entité.

Controller pour l 'ajout ou la modification d'une données

```
Sensio\Bundle\FrameworkExtraBundle\Configuration\Security;
#[Route('/admin')]
#[Security("is_granted('ROLE_ADMIN')", statusCode: 403)]
class AdminController {...
  #[Route('/listPartner', name: 'app_admin_partnerList')]
  public function ListPartner(...): Response
  {
          $partners = $partnerRepo->findAllPartners();
          $structures = $structureRepo->findAllStructures();
          $countPartners = $partnerRepo->countPartners();
    return $this->render('admin/partner/index.html.twig', [
          'titleIndex' => 'Listes des Franchises',
          'countPartners' => $countPartners,
          'partners' => $partners,
          'structures' => $structures,
          'partners' => $partnerRepo->getPaginatedPartner((int)
$request->query->get("page")),
    ]);}
```

```
class NewPartnerController {
  #[Route('/newpartner', name: 'new_partner')]
  public function formNewPartner(..., MailService $mail): Response
  {
      if(!$partner){
        $partner = new Partner();
      }
      $formPartner = $this->createForm(PartnerType::class, $partner);
      $formPartner->handleRequest($request);
      if ($formPartner->isSubmitted() && $formPartner->isValid()){
      if(!$partner->getId()){
      }
      $structures = $partner->getStructures();
      foreach ($structures as $structure) {
        $partner->addStructure($structure);
      }
      $entityManager->persist($partner);
      $entityManager->flush();
      $this->addFlash('success', 'La Franchise a bien été créé');
    ...
```

Formulaire :

- ★ Ajouter
- ★ Modifier

Un utilisateur, une Franchise, une Structure et une Permissions

Création d'un formulaire dans les controllers concernés

```php
$formPartner = $this->createForm(PartnerType::class, $partner);
$formPartner->handleRequest($request);
      if ($formPartner->isSubmitted() && $formPartner->isValid()){
            if(!$partner->getId()){

            }
      $structures = $partner->getStructures();
      foreach ($structures as $structure) {
          $partner->addStructure($structure);
      }

      $entityManager->persist($partner);
      $entityManager->flush();

      $this->addFlash('success', 'La Franchise a bien été créé');
```

```php
class PartnerType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options)
    {
        $builder
                //RECUP LA COLLECTION DE USER
            ->add('user', EntityType::class,[
                'class'=> User::class,
                'label'=> 'Selectionner un compte',
                'placeholder'=>'Choisissez ...',
                'choice_label'=> function (User $user){
                    return $user->getUsername();
                },
                'query_builder' => function (UserRepository $userRepo) {
                    $qb = $userRepo->createQueryBuilder('u');
                    return $qb
                        ->where('u.roles LIKE :role')
                        ->setParameter('role', '%"' . 'ROLE_PARTNER' . '"%')
                        ->orderBy('u.username');}

            ])
```

f. Service Email

Un mail est envoyé pour chaque création ou modification d'un compte

Rajouter la variable dans .env.local

MAILER_DSN=smtp://8d4988e29245f2:61d74c2fc9b32d@smtp.mailtrap.io:2525?encryption=tls&auth_mode=login

```php
namespace App\Service;
use Symfony\Bridge\Twig\Mime\TemplatedEmail;
use Symfony\Component\Mailer\MailerInterface;
use Symfony\Component\Mime\Email;
class MailService
{
    public function __construct(private MailerInterface $mailer){}

    public function sendEmail(string $from, string $to, string $subject,
string $template, array $context ) :void
    {
        $email = (new TemplatedEmail())
            ->from($from)
            ->to($to)
            ->subject($subject)
            ->htmlTemplate("email/$template.html.twig")
            ->context($context);
        $this->mailer->send($email);
    }
}
```

Création

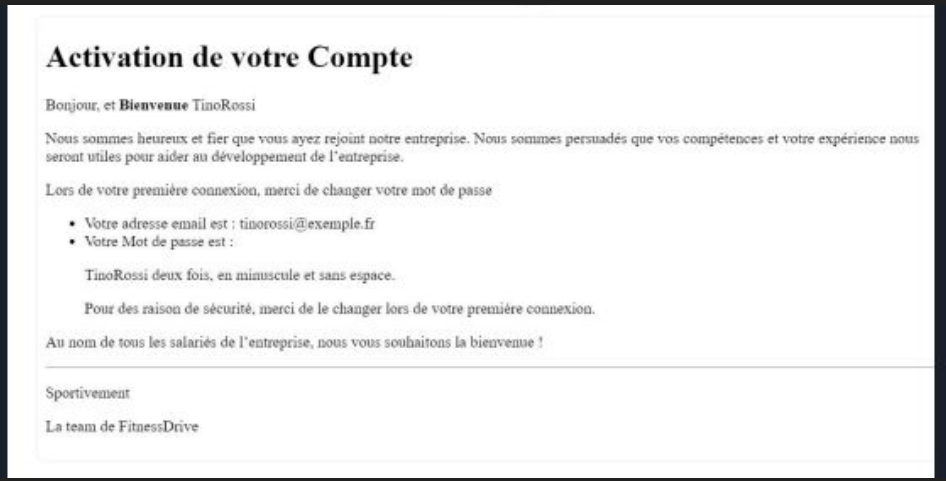
```php
//sendEmail
            $mail->sendEmail(
                'fitnessDrive@outlook.fr',
                $partner->getUser()->getEmail(),
                'Activation de votre compte',
                'registerPartner',
                compact('partner')

            );

            $this->addFlash('send', 'Email d\'Activation a bien été envoyé');
            }
```

Injection dans les controllers concernés

Création de la vue

```
<h1>Modification de votre Compte</h1>
<p>Bonjour,{% if partner.name is defined %}{{ partner.name }}{% endif %}</p>
<p>Nous avons détecter des changements sur votre compte</p>
<p>Vous pouvez vous connecter pour voir ses changements !!</p>
<p> Pour Rappel votre identifiant de connection est : <br>
      {% if partner.name is defined %}{{ partner.user.email }}{% endif %}
</p>
<p>Si vous avez des difficultés à vous connecter, veuillez nous contacter.</p>
<hr>
<p>Sportivement</p>
<p>La team de FitnessDrive</p>
```

Modèle



Activation de votre Compte

Bonjour, et **Bienvenue** TinoRossi

Nous sommes heureux et fier que vous ayez rejoint notre entreprise. Nous sommes persuadés que vos compétences et votre expérience nous seront utiles pour aider au développement de l'entreprise.

Lors de votre première connexion, merci de changer votre mot de passe

- Votre adresse email est : tinorossi@exemple.fr
- Votre Mot de passe est :

  TinoRossi deux fois, en minuscule et sans espace.

  Pour des raison de sécurité, merci de le changer lors de votre première connexion.

Au nom de tous les salariés de l'entreprise, nous vous souhaitons la bienvenue !

Sportivement

La team de FitnessDrive

d. Front-end

a. Twig

b. Styles

c. JavaScript

a. Twig

Twig est un moteur de template de symfony

Création d'une base qui sera étendue aux autres pages.

Personnalisé des blocks

Inclure des bouts de code

Récupération de données du controller

Effectuer des vérifications

Permet de gagner du temps

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <meta name="description" content="Application de gestion des droits
des salles de sport de la marque FitnessDrive"/>
        <title>
            {% block title %}FitnessDrive{% endblock %}
        </title>

        {% block stylesheets %}

        {% endblock %}

        {% block javascripts %}

        {% endblock %}
    </head>
{% block body %}
    <body>
        <header>
            {% block navbar %}    {% endblock %}
        </header>
    </body>
{% endblock %}
</html>
```

```twig
{% extends 'base.html.twig' %}

{% block title %}Franchise | {{ parent() }}{% endblock %}

{% block body %}
<body class="bgPartner">
    {% block navbar %}
        {% include "_partials/_navbarClient.html.twig" %}
    {% endblock %}
```

Vérification

```twig
{% if app.user %}
...
{% else%}
...
{% endif %}
```

```twig
<div class="d-flex justify-content-center flex-wrap container">
{% for structure in structures %}
    <div class="card" >
        <div class="card-body">
            <h5 class="card-title text-center mb-3">FitnessDrive -
                {{structure.partner.namePartner}}
                -
                {{structure.nameStructure}}
            </h5>
            <div class="d-block d-sm-flex justify-content-sm-between">
                <p class="text-muted">Gérant
                    {{structure.user.username}}
                </p>
                    <!-- Structure Active ? -->
                <p>{{ structure.isActive ? '<span class="p-
2">Active</span…</svg>' : '<span class="p-2">InActive</span…'}}</p>
            </div>

            …

            <div class="card-footer d-flex justify-content-center">
                <a href=" {{ path( 'app_partner_structure_show', {'id':
structure.id}) }} " type="button" class="btn-sm btnDetail">Détail</a>
            </div>
        </div>
    </div>
{% else %}
    <div class="alert alert-success text-center">
        <p>Vous n'avez aucune structure !</p>
    </div>
{% endfor %}
</div>
```

b. Styles

Mobile First / Sass / Bootstrap

Responsive :

MediaQueries

Système de grille de Bootstrap

Class personnalisé

```
.btnHome
    background-color: $green
    color: #fff
    transition: 0.3s
    transition-property: color
    &:hover
        color: #000
```

```
//taille ecran
$xs: 0,
$sx2: 400px,
$sx: 460px,
$sm: 576px,
$md: 768px,
$lg: 992px,
$xl: 1200px,
$xxl: 1400px
```

```
@mixin media($size)
    @media screen and (min-width: $size)
        @content
```

```
@include media($sm)
    .main-container
        & .count
            font-size: 0.8em
        & .search
            & .inputSearch
                width: 450px
```

c. JavaScript

La recherche dynamique :

- Barre de recherche


```
{% for partner in partners %}
    <div class="partner d-flex  partner-all {{
partner.isActive ? 'partner-active'  : 'partner-inActive' }}" >
        <div class="card "   >
            <div class="card-body ">...
{% else%}
…
{% endfor %}...
```

```
 //BarSearch
 const inputsearch =
 document.getElementById("searchbar");
 const partners =
 document.getElementsByClassName('partner');


// function barsearch
const mySearch = () => {
    let valueInput = inputsearch.value;
      for( i=0; i< partners.length; i++) {
      if(partners[i].innerHTML.toLowerCase().indexOf(v
      alueInput) != -1) {
      partners[i].classList.remove("d-none");
            partners[i].classList.add("d-flex");
                } else {
            partners[i].classList.remove("d-flex");
            partners[i].classList.add("d-none");
                }
        }
};
inputsearch.addEventListener("change", mySearch);
```

e. La sécurité

Partie Sécurité

L'authentification

Installation du bundle de sécurité

Création de Authentification :

- ★ **symfony console make:auth**
- ★ LoginForm => redirection à la connexion
- ★ SécurityController
  - Login
  - Logout

Template pour le formulaire de login

Entité User - Insertion de contrainte

- ★ Email unique
- ★ LoginUserType => contraintes

Création d'un Compte de connexion

- ★ Hasher le Mot de passe

```
class LoginFormAuthenticator
public function onAuthenticationSuccess(...): ?Response
    {...
        $roles = $token->getRoleNames();
        if (in_array('ROLE_ADMIN', $roles, true)) {
            $user = $token->getUser();
            $id = $user->getId();
            return new
RedirectResponse($this->router->generate("app_admin", ['id' => $id]));
        }
        return new
RedirectResponse($this->urlGenerator->generate('app_home'));
```

```
#[UniqueEntity(fields:'email', message:'L\'email que vous avez indiqué est déjà utiliser')]
class User
 #[ORM\Column(length: 50, unique:true)]
  private ?string $username = null;
```

```
if ($form->isSubmitted() && $form->isValid()){
      $user->setPassword(
        $userPasswordHasher->hashPassword(
          $user,
        $form->get('password')->getData()
        )
```

Sécuriser les routes

Configuration :

- ★ Page Admin
- ★ Accès aux routes
- ★ Twig

```
access_control:
    - { path: ^/admin, roles: ROLE_ADMIN }
    - { path: ^/partner, roles: ROLE_PARTNER}
    - { path: ^/structure, roles: ROLE_STRUCTURE }
    - { path: ^/home, roles: [ROLE_ADMIN, ROLE_PARTNER, ROLE_STRUCTURE] }
    - { path: ^/contact, roles: [ROLE_ADMIN, ROLE_PARTNER, ROLE_STRUCTURE] }
```

```
{% endif app.user %}
  {% if is granted('ROLE_PARTNER') %}...
{% else %}...
{% endif %}
```

Accès au site

```
$roles = $token->getRoleNames();
        if (in_array('ROLE_ADMIN', $roles, true)) {
            /**
             * @var User $user
             */
            $user = $token->getUser();
            $id = $user->getId();
            return new RedirectResponse(
               $this->router->generate(
                "app_admin", ['id' => $id]));
        }
             return new RedirectResponse(
               $this->urlGenerator->generate(
                'app_home'));
```

```
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Security;
#[Route('/admin')]
#[Security("is_granted('ROLE_ADMIN')", statusCode: 403)]
class AdminController extends AbstractController
```

4. Conclusion

Concernant le projet, les fonctionnalités de base sont développées. Il a été le premier projet développé avec le langage PHP et le framework Symfony. J'ai beaucoup appris lors de son développement. Mon projet est mis en ligne sur heroku, puis il a migré vers Fly.io.

Cette application est fonctionnelle.

Le développement est un défi de tous les jours, car il faut toujours apprendre, à l'aide des documentations officielles, des forums dédiés et de tutoriels. Je pense que c'est la meilleure méthode de travail afin de progresser rapidement, et une bonne façon d'entrevoir ce qu'est le métier de développeur.