



DOSSIER PROFESSIONNEL (DP)

Nom de naissance - Arfi
Nom d'usage - Arfi
Prénom - Benjamin
Adresse - 5 bd icard , 13010, Marseille

Titre professionnel visé

Concepteur(trice) Développeur(se) Informatique

MODALITÉ D'ACCÈS :

- Parcours de formation
- Validation des Acquis de l'Expérience (VAE)

DOSSIER PROFESSIONNEL (DP)

Présentation du dossier

Le dossier professionnel (DP) constitue un élément du système de validation du titre professionnel.
Ce titre est délivré par le Ministère chargé de l'emploi.

Le DP appartient au candidat. Il le conserve, l'actualise durant son parcours et le présente **obligatoirement à chaque session d'examen.**

Pour rédiger le DP, le candidat peut être aidé par un formateur ou par un accompagnateur VAE.

Il est consulté par le jury au moment de la session d'examen.

Pour prendre sa décision, le jury dispose :

1. des résultats de la mise en situation professionnelle complétés, éventuellement, du questionnaire professionnel ou de l'entretien professionnel ou de l'entretien technique ou du questionnement à partir de productions.
2. du Dossier Professionnel (DP) dans lequel le candidat a consigné les preuves de sa pratique professionnelle.
3. des résultats des évaluations passées en cours de formation lorsque le candidat évalué est issu d'un parcours de formation
4. de l'entretien final (dans le cadre de la session titre).

[Arrêté du 22 décembre 2015, relatif aux conditions de délivrance des titres professionnels du ministère chargé de l'Emploi]

Ce dossier comporte :

- pour chaque activité-type du titre visé, un à trois exemples de pratique professionnelle ;
- un tableau à renseigner si le candidat souhaite porter à la connaissance du jury la détention d'un titre, d'un diplôme, d'un certificat de qualification professionnelle (CQP) ou des attestations de formation ;
- une déclaration sur l'honneur à compléter et à signer ;
- des documents illustrant la pratique professionnelle du candidat (facultatif)
- des annexes, si nécessaire.

DOSSIER PROFESSIONNEL (DP)

Pour compléter ce dossier, le candidat dispose d'un site web en accès libre sur le site.

► <http://travail-emploi.gouv.fr/titres-professionnels>

Sommaire

Exemples de pratique professionnelle

Activité-type n° 1 : Développement d'un jeu en python	p.	5
- MySokoban - cahier des charges et conception	p.	5
- MySokoban - librairies et map	p.	11
- MySokoban - création et intégration des différentes classes	p	15
Activité-type n° 2 : Développer le back-end avec Symfony - Api Platform (API REST)	p.	23
- Modélisation d'une base de donnée	p.	23
- Symfony et Api Platform	p.	30
- <i>Création et paramétrage de mes entités</i>	p	37
Activité-type n° 3 : Développer le front-end avec Expo - React Native (App mobile)	p.	46
- Conception et maquettage d'une application	p.	46
- React Native - Expo	p.	49
- Création de composants et hooks	p	56
Titres, diplômes, CQP, attestations de formation (facultatif)	p.	65
Déclaration sur l'honneur	p.	66
Documents illustrant la pratique professionnelle (facultatif)	p.	67
Annexes (Si le RC le prévoit)	p.	68

EXEMPLES DE PRATIQUE PROFESSIONNELLE

DOSSIER PROFESSIONNEL (DP)

Activité-type 1 Développement d'un jeu en python

Exemple n°1 - mySokoban - cahier des charges et conception

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

mySokoban est un jeu vidéo avec comme idée de départ de faire un jeu sokoban, mais au fil de la rédaction du cahier des charges, j'ai décidé de partir sur un RPG.

Il a pour but de donner la possibilité à tous les joueurs de contrôlé un personnage et d'évoluer dans une plaine tout en esquivant/tuant tous les monstres présents dans cette dernière, mais il faut faire attention à ne pas se faire toucher par un monstre sinon c'est le "Game Over", notre héros pourra également taper les ennemis pour les éliminer.

Après le recueil des besoins de mon jeu, j'ai défini un ensemble de besoin technique à développer en décrivant en détail ce que les joueurs pourraient faire sur le jeu et pourquoi ils le font. Qui m'ont servi de base pour mettre en place mon cahier des charges :

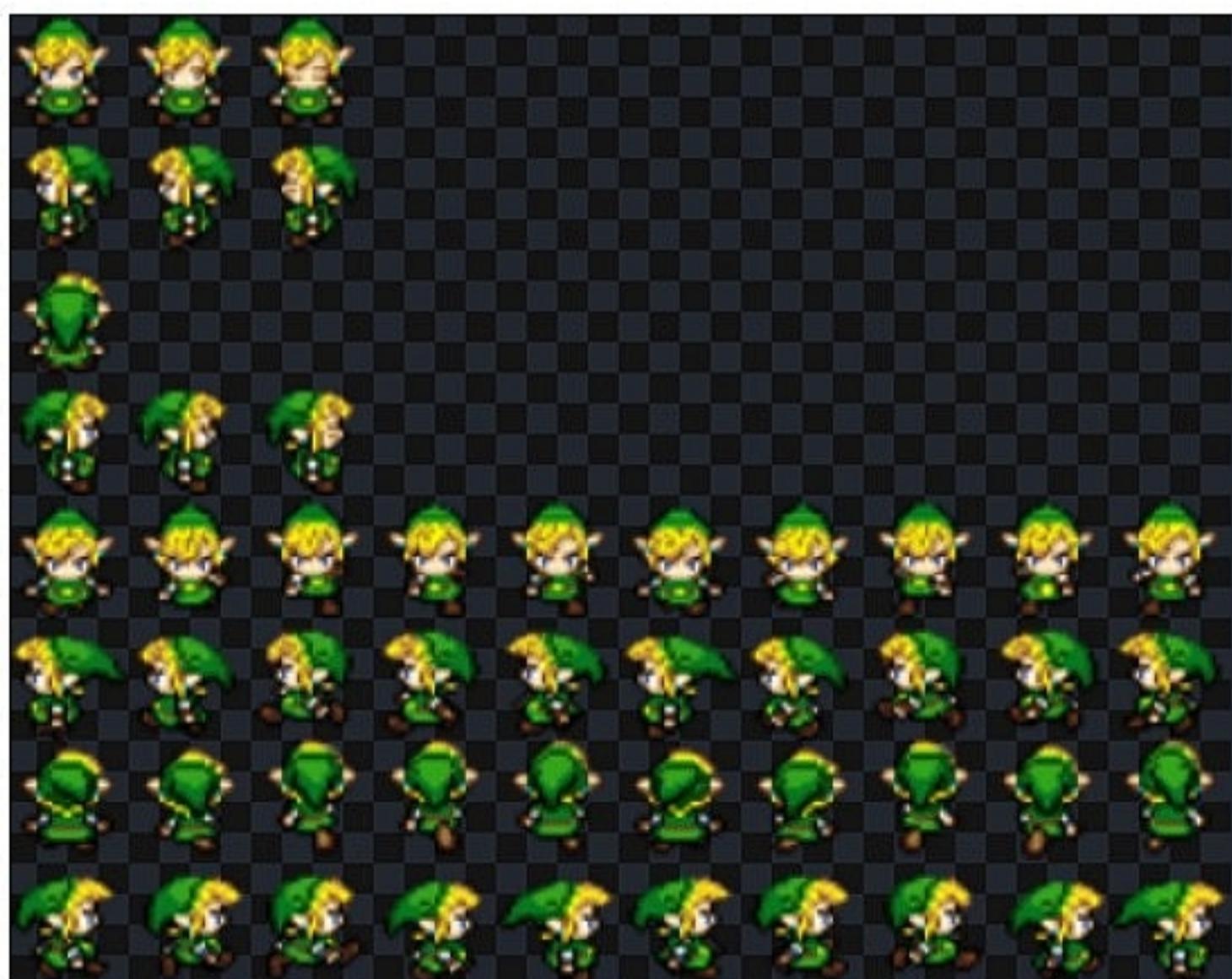
- Sur l'écran d'accueil : une musique avec le nom le logo du jeu et un bouton ou cliquer pour lancer la partie.
- définir mon univers graphique, le game design, le sound design et les sprites utilisés.
- Le joueur a une hitbox, peut taper les ennemis, ce mouvoir et doit être animé dans ces mouvements.
- Quand le joueur frappe un ennemi, l'ennemi meurt et disparaît.
- Les ennemis auront une hitbox aussi, un pattern de mouvement de gauche à droite.
- Quand le joueur rentre en collision avec un ennemi, cela déclenche l'écran de

DOSSIER PROFESSIONNEL (DP)

Game Over

Juste après avoir rédigé mon cahier des charges, je me suis d'abord renseigné sur comment faire un jeu en python avec la librairie PyGame en suivant la documentation et après ça j'ai commencé à définir la charte graphique de mon jeu, j'ai décidé de partir sur un fan game The Legend Of Zelda, ce choix s'est orienté au vu de la richesse en terme d'univers graphique, musical et la magie qui se dégage de cette licence.

pour mon personnage, j'ai utilisé une spritesheet de link que j'ai trouvé sur internet :



Avec ces sprites, je peux même animer tous les mouvements vu qu'ils sont tous présents.

DOSSIER PROFESSIONNEL (DP)

voici ceux des ennemis :

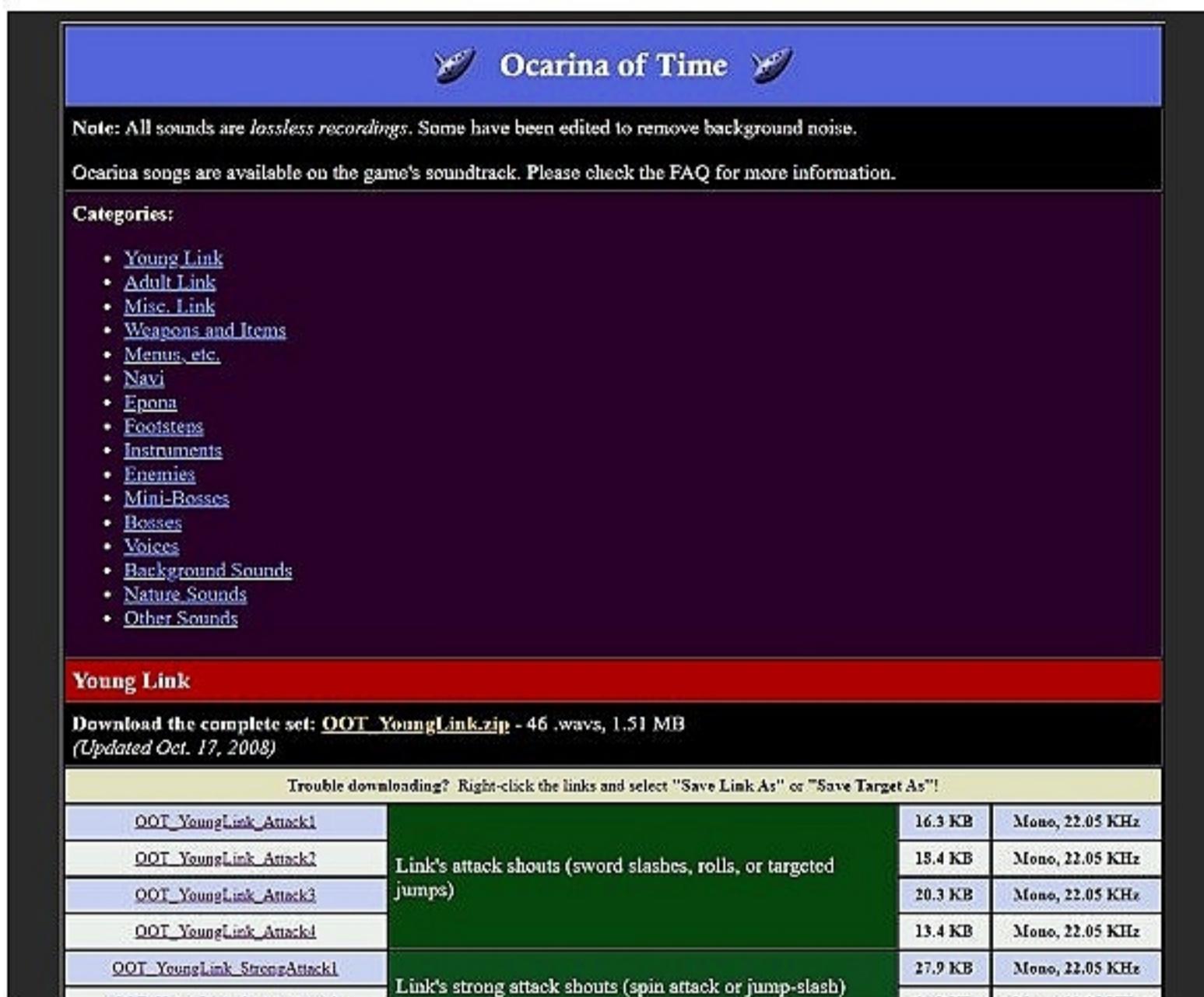


et concernant l'environnement :



DOSSIER PROFESSIONNEL (DP)

Maintenant que mes sprites et mes assets sont défini, je défini maintenant le sound design, j'ai trouvé une bibliothèque avec tous les effets sonore :



après cela, il ne me reste plus qu'à définir une matrice pour définir comment sera dessiné ma map, j'ai décidé de faire un couloir de jeu qui ressemble à celà :

```
[ 'BBBBBBBBBBBBBBBBBBBBBBB',
  'B..E.....B....E...B',
  'B.....B.....B',
  'B....BBBBBBBBBB....B',
  'B.....',
  'B.....P.....B',
  'B.....',
  'B..E...BBBBBBBBBBBB',
  'B.....',
  'B.....B',
  'B.....B....E.....B',
  'B.....B.....B',
  'B.....B....E.....B',
  'B.....B.....B',
  'B.....BB.....B',
```

DOSSIER PROFESSIONNEL (DP)

```
'B..E.....E.....B',
'B.....B.....B',
'B....E.....B',
'B.....B.....B',
'B..E...BBBBBBBBBBBB',
'B.....B.....B',
'B.....B.....B',
'B....B....E.....B',
'B....B.....B',
'B....BB.....B',
'B.....B.....B',
'B.....E.....B',
'BBBBBBBBBBBB....BBB',
'B....B....E.....B',
'B....B.....B',
'B....BB.....B',
'B..E.....E.....B',
'B.....B.....B',
'B....E.....B',
'BBBBBBBBBBBBBBBBBBB', ]
```

2. Précisez les moyens utilisés :

Les moyens utilisés pour ce projet :

Un ordinateur avec une connexion internet.

Comme éditeur de texte J'ai utilisé Visual studio code, qui est un environnement de développement intégré (IDE).

Pour la recherche d'assets je me suis servi d'internet ainsi que de fansite zelda qui répertorie tous les éléments dont je peux sensiblement avoir besoin.

Pour la création de ma TileMap, je me suis renseigné sur StackOverFlow ainsi que sur la documentation de la librairie PyGame.

3. Avec qui avez-vous travaillé ?

Dossier Professionnel (DP)

Sur ce projet, j'ai travaillé seul.

4. Contexte

Nom de l'entreprise, organisme ou association

La Plateforme_

Chantier, atelier, service

Dans le cadre de la formation concepteur, développeur d'application

Période d'exercice

Du : 09/05/2025

au : 13/05/2025

5. Informations complémentaires(*facultatif*)

DOSSIER PROFESSIONNEL (DP)

Activité-type 1 Développement d'un jeu en python

Exemple n° 2 - mySokoban - librairies et map

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Pygame est un ensemble de modules python conçus pour développer des jeux vidéo. Pygame ajoute des fonctionnalités à l'excellente bibliothèque SDL (simple DirectMedia Layer). Cela permet de créer des jeux complets et des programmes multimédias en langage python.

Pygame est hautement portable et fonctionne sur presque toutes les plateformes et systèmes d'exploitation.

Pygame lui-même a été téléchargé des millions de fois et est une référence de choix, si ce n'est la référence par excellence pour en développer avec python.

Pygame est gratuit. Publié sous la licence LGPL, on peut créer avec des jeux open source, freeware, shareware et commerciaux avec.

La meilleure façon d'installer pygame est avec l'outil pip (qui est l'outil que python utilise pour installer les packages).

Pour mon cas, l'installation sous Windows ce fait avec la ligne de commande suivante :

```
py -m pip install -U pygame --user
```

et ensuite pour vérifier si j'ai bien la librairie d'installer sur mon ordinateur, je peux lancer cette commande qui lance un jeu vidéo déjà inclus avec la librairie PyGame :

```
py - m pygame . exemples . extraterrestres
```

maintenant je n'ai plus qu'à importer la librairie sur mon projet, voici mes imports :

```
from re import T
import pygame
from sprites import *
from config import *
import sys
from pygame import mixer
```

DOSSIER PROFESSIONNEL (DP)

après cela je crée une class Game ou je commence à inclure dans le constructeur pygame.init(), la création de la fenêtre de jeu avec ses dimension que j'ai défini dans un fichier de configuration ainsi que tous mes assets, mes sprites et mes effets sonores pour faciliter leurs appelles par la suite :

```
class Game:  
    def __init__(self):  
        pygame.init()  
        self.screen = pygame.display.set_mode((WIN_WIDTH, WIN_HEIGHT))  
        self.clock = pygame.time.Clock()  
        self.font = pygame.font.Font('ARCADECLASSIC.TTF', 32)  
        self.font1 = pygame.font.Font('ARCADECLASSIC.TTF', 10)  
        self.running = True
```

je commence avec la création de ma map :

```
def createTilemap(self):  
    self.gameoversound.stop()  
    self.accueil.stop()  
    self.gerudo.stop()  
    self.gerudo.play(-1)  
    for i, row in enumerate(tilemap):  
        for j, column in enumerate(row):  
            Ground(self, j, i)  
            if column == "B":  
                Block(self, j, i)  
            if column == "E":  
                Enemy(self, j, i)  
            if column == "P":  
                self.player = Player(self, j, i)
```

comme on peut le voir je boucle sur ma variable tilemap qui correspond à cela :

DOSSIER PROFESSIONNEL (DP)

```
tilemap = [
    'BBBBBBBBBBBBBBBBBBBB',
    'B..E.....B....E...B',
    'B.....B.....B.....B',
    'B....BBBBBBBBBB....B',
    'B.....B.....B.....B',
    'B.....P.....B.....B',
    'B.....B.....B.....B',
    'B..E...BBBBBBBBBBBB',
    'B.....B.....B.....B',
    'B.....B.....B.....B',
    'B....B....E.....B',
    'B....B.....B.....B',
```

Je boucle d'abord sur chaque ligne de ma map et ensuite sur chaque colonne de chaque ligne et je fais des conditions ou je remplace chaque lettre par l'asset qui le correspond, dans l'exemple ci dessous, "B" correspond à un rocher immobile, "E" par les ennemis, "P" par le sprite de mon joueur et les "." par de la pelouse.

ensuite je rappel cette fonction `createTileMap()` dans la fonction de début de jeu qui est la suivante :

```
def new(self):
    # une nouvelle partie à commencé

    self.playing = True

    self.all_sprites = pygame.sprite.LayeredUpdates()
    self.blocks = pygame.sprite.LayeredUpdates()
    self.enemis = pygame.sprite.LayeredUpdates()
    self.attacks = pygame.sprite.LayeredUpdates()

    self.createTilemap()
```

et pour finir je boucle sur ma classe game pour que le jeu se lance lorsque je l'execute:

Dossier Professionnel (DP)

```
g = Game()
g.intro_screen()
g.new()

while g.running:
    g.main()
    g.game_over()

pygame.quit()
sys.exit()
```

2. Précisez les moyens utilisés :

Un ordinateur avec une connexion internet.

Comme éditeur de texte J'ai utilisé Visual studio code, qui est un environnement de développement intégré (IDE).

Pour la recherche d'éléments technique je me suis servi de la documentation de pygame ainsi que de stackOverFlow et de certain tutoriel sur youtube

3. Avec qui avez-vous travaillé ?

j'ai travaillé seul sur ce projet.

4. Contexte

Nom de l'entreprise, organisme ou association

La Plateforme_

Chantier, atelier, service

Dans le cadre de la formation concepteur / développeur d'application.

Période d'exercice

Du : 09/05/2025 au : 13/05/2025

5. Informations complémentaires(facultatif)

DOSSIER PROFESSIONNEL (DP)

Activité-type 1 Développement d'un jeu en python

Exemple n° 3 - création et intégration des différentes classes

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Concernant ce jeu, je l'ai réalisé entièrement en orienté objet, avec des classes des fonctions, des attributs...

Il faut savoir que chaque éléments de mon jeu est régi par une classe qui lui est propre, prenant l'exemple du personnage que nous jouons :

elle est régi par la classe player qui est la suivante :

```
class Player(pygame.sprite.Sprite):

    def __init__(self, game, x, y):

        self.game = game
        self._layer = PLAYER_LAYER
        self.groups = self.game.all_sprites
        pygame.sprite.Sprite.__init__(self, self.groups)

        self.x = x * TILESIZE
        self.y = y * TILESIZE
        self.width = TILESIZE
        self.height = TILESIZE

        self.x_change = 0
        self.y_change = 0

        self.facing = 'down'
        self.animation_loop = 1

        # image_to_load = pygame.image.load("./assets/bas-link.png")
        self.image = self.game.character_spritesheet.get_sprite(0, 10,
self.width, self.height)
        self.image.set_colorkey(BLACK)
        # self.image.blit(image_to_load, (0, 0))
```

DOSSIER PROFESSIONNEL (DP)

```
self.rect = self.image.get_rect()
self.rect.x = self.x
self.rect.y = self.y

def update(self):
    self.movement()
    self.animate()
    self.collide_enemy()

    self.rect.x += self.x_change
    self.collide_blocks('x')
    self.rect.y += self.y_change
    self.collide_blocks('y')

    self.x_change = 0
    self.y_change = 0

def movement(self):
    keys = pygame.key.get_pressed()
    if keys[pygame.K_LEFT]:
        for sprite in self.game.all_sprites:
            sprite.rect.x += PLAYER_SPEED
        self.x_change -= PLAYER_SPEED
        self.facing = 'left'
    if keys[pygame.K_RIGHT]:
        for sprite in self.game.all_sprites:
            sprite.rect.x -= PLAYER_SPEED
        self.x_change += PLAYER_SPEED
        self.facing = 'right'
    if keys[pygame.K_UP]:
        for sprite in self.game.all_sprites:
            sprite.rect.y += PLAYER_SPEED
        self.y_change -= PLAYER_SPEED
        self.facing = 'up'
    if keys[pygame.K_DOWN]:
        for sprite in self.game.all_sprites:
            sprite.rect.y -= PLAYER_SPEED
        self.y_change += PLAYER_SPEED
```

DOSSIER PROFESSIONNEL (DP)

```
self.facing = 'down'

def collide_enemy(self):
    hits = pygame.sprite.spritecollide(self, self.game.ennemis, False)
    if hits:
        self.kill()
        self.game.playing = False

def collide_blocks(self, direction):
    if direction == "x":
        hits = pygame.sprite.spritecollide(self, self.game.blocks, False)
        if hits:
            if self.x_change > 0:
                self.rect.x = hits[0].rect.left - self.rect.width
                for sprite in self.game.all_sprites:
                    sprite.rect.x += PLAYER_SPEED
            if self.x_change < 0:
                self.rect.x = hits[0].rect.right
                for sprite in self.game.all_sprites:
                    sprite.rect.x -= PLAYER_SPEED

    if direction == "y":
        hits = pygame.sprite.spritecollide(self, self.game.blocks, False)
        if hits:
            if self.y_change > 0:
                self.rect.y = hits[0].rect.top - self.rect.height
                for sprite in self.game.all_sprites:
                    sprite.rect.y += PLAYER_SPEED
            if self.y_change < 0:
                self.rect.y = hits[0].rect.bottom
                for sprite in self.game.all_sprites:
                    sprite.rect.y -= PLAYER_SPEED

def animate(self):
    down_animations = [self.game.character_spritesheet.get_sprite(128, 0,
self.width, self.height),
                       self.game.character_spritesheet.get_sprite(128, 128,
self.width, self.height),
                       self.game.character_spritesheet.get_sprite(288, 128,
```

DOSSIER PROFESSIONNEL (DP)

```
self.width, self.height)]  
  
    up_animations = [self.game.character_spritesheet.get_sprite(0, 192,  
self.width, self.height),  
                      self.game.character_spritesheet.get_sprite(128, 192,  
self.width, self.height),  
                      self.game.character_spritesheet.get_sprite(288, 192,  
self.width, self.height)]  
  
    left_animations = [self.game.character_spritesheet.get_sprite(0, 160,  
self.width, self.height),  
                       self.game.character_spritesheet.get_sprite(128, 160,  
self.width, self.height),  
                       self.game.character_spritesheet.get_sprite(288, 160,  
self.width, self.height)]  
  
    right_animations = [self.game.character_spritesheet.get_sprite(0, 224,  
self.width, self.height),  
                         self.game.character_spritesheet.get_sprite(128, 224,  
self.width, self.height),  
                         self.game.character_spritesheet.get_sprite(256, 224,  
self.width, self.height)]  
  
    if self.facing == "down":  
        if self.y_change == 0:  
            self.image = self.game.character_spritesheet.get_sprite(0, 0,  
self.width, self.height)  
        else:  
            self.image = down_animations[math.floor(self.animation_loop)]  
            self.animation_loop += 0.1  
            if self.animation_loop >= 3:  
                self.animation_loop = 1  
  
    if self.facing == "up":  
        if self.y_change == 0:  
            self.image = self.game.character_spritesheet.get_sprite(0, 192,  
self.width, self.height)  
        else:  
            self.image = up_animations[math.floor(self.animation_loop)]
```

DOSSIER PROFESSIONNEL (DP)

```
self.animation_loop += 0.1
if self.animation_loop >= 3:
    self.animation_loop = 1

if self.facing == "left":
    if self.x_change == 0:
        self.image = self.game.character_spritesheet.get_sprite(0, 160,
self.width, self.height)
    else:
        self.image = left_animations[math.floor(self.animation_loop)]
        self.animation_loop += 0.1
        if self.animation_loop >= 3:
            self.animation_loop = 1

if self.facing == "right":
    if self.x_change == 0:
        self.image = self.game.character_spritesheet.get_sprite(0, 224,
self.width, self.height)
    else:
        self.image = right_animations[math.floor(self.animation_loop)]
        self.animation_loop += 0.1
        if self.animation_loop >= 3:
            self.animation_loop = 1
```

On peut voir dans la classe suivante que toutes les actions et les différentes mécaniques de gameplay lié au joueur y sont. On peut y voir les mouvements et les animations quand il se déplace, le rafraîchissement, la collision avec les blocs et les ennemis, quand le joueur attaque... et par la suite il suffit simplement de les implémentés dans les différentes classes et fonctions que j'appel quand je lance ma boucle de jeu.

DOSSIER PROFESSIONNEL (DP)

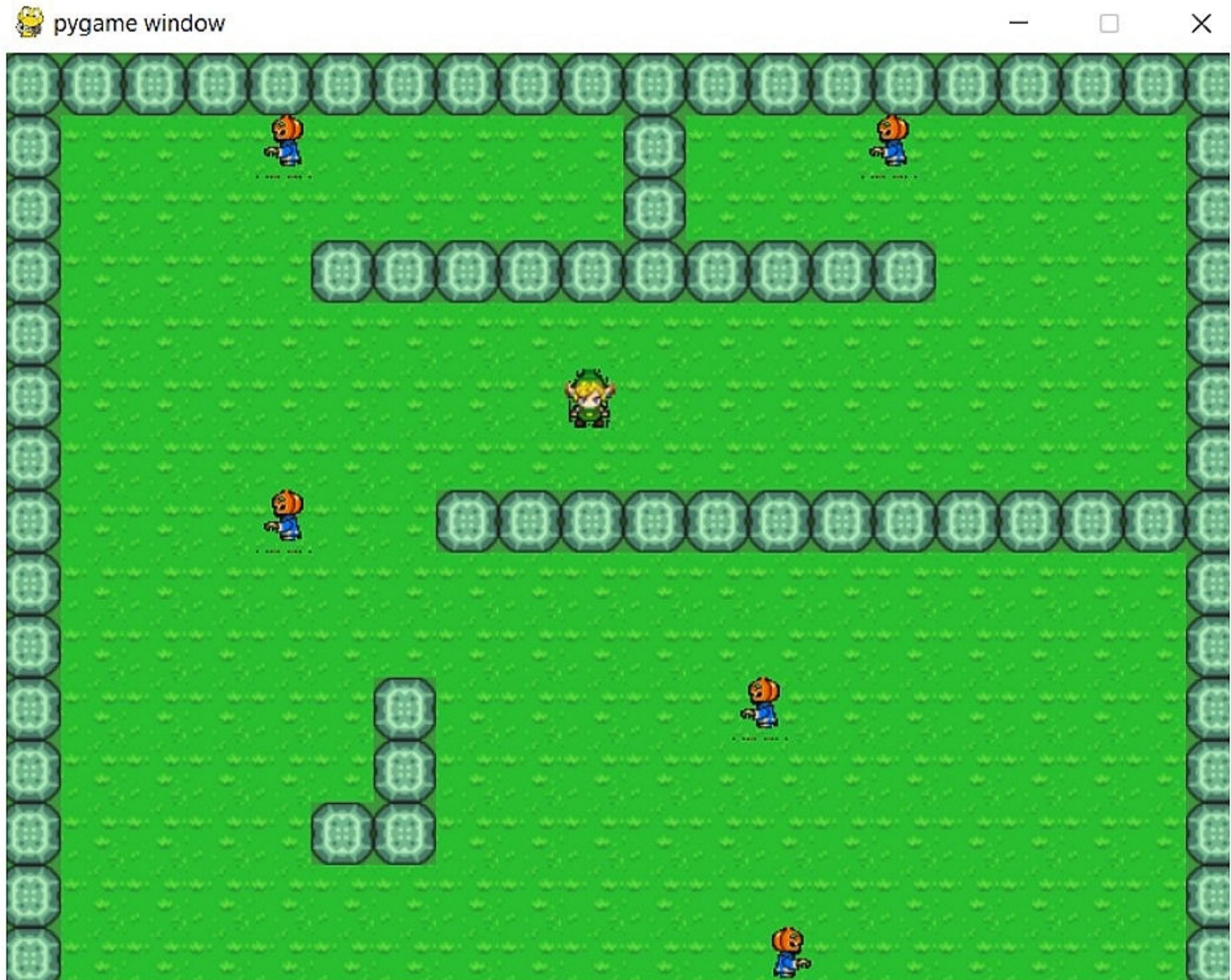
Voici le résultat final :

menu du début de jeu :



DOSSIER PROFESSIONNEL (DP)

le jeu final :



DOSSIER PROFESSIONNEL (DP)

le Game-Over :



Dossier Professionnel (DP)

2. Précisez les moyens utilisés :

Les moyens utilisés pour ce projet :

Un ordinateur avec une connexion internet.

Comme éditeur de texte J'ai utilisé Visual studio code, qui est un environnement de développement intégré (IDE).

Pour la rédactions des différentes classes et comment m'y prendre pour intégré les différentes logique de gameplay, je me suis servi de la documentation de PyGame, ainsi que de tutoriel sur youtube.

3. Avec qui avez-vous travaillé ?

j'ai travaillé seul sur ce projet.

4. Contexte

Nom de l'entreprise, organisme ou association

La Plateforme_

Chantier, atelier, service

Dans le cadre de la formation concepteur /développeur d'application.

Période d'exercice

Du : 09/05/2025 au : 13/05/2025

5. Informations complémentaires(facultatif)

La suite de tutoriel youtube dont je me suis servi :

 Pygame RPG Tutorial #1 - Pygame Tutorial

la documentation PyGame dont je me suis servi :

<https://www.pygame.org/docs/>

DOSSIER PROFESSIONNEL (DP)

Activité-type 2 Développer le back-end avec Symfony - Api Platform (API REST)

Exemple n° 1 - modélisation d'une base de donnée

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Méthode MERISE

MERISE (Méthode d'Etude et de Réalisation Informatique pour les Systèmes d'Entreprise), c'est une méthode française qui a émergée dans les années 70 en France qui permet la modélisation et la conception de S.I. Parmi les ressources informatiques de ces S.I., il y a en particulier les fichiers de données, bases de données et système de gestion de bases de données (S.G.B.D.).

C'est sur ce dernier point que la méthode MERISE m'a été utile, car c'est en se basant sur ses principes de modélisation que j'ai conçu la base de données de Côté Pote.

Modèle conceptuel de données (MCD)

c'est un modèle simplifié de la base de données, où les tables sont seulement au stade d'entités (c'est l'équivalent de la table en MCD), entre les différentes entités il y a des liaisons que l'on appelle association, qui est le verbe d'action qui décrit l'opération qui se fait entre les deux entités, donc nous distinguons principalement les entités et les association, d'où son second nom de schéma Entité/Association.

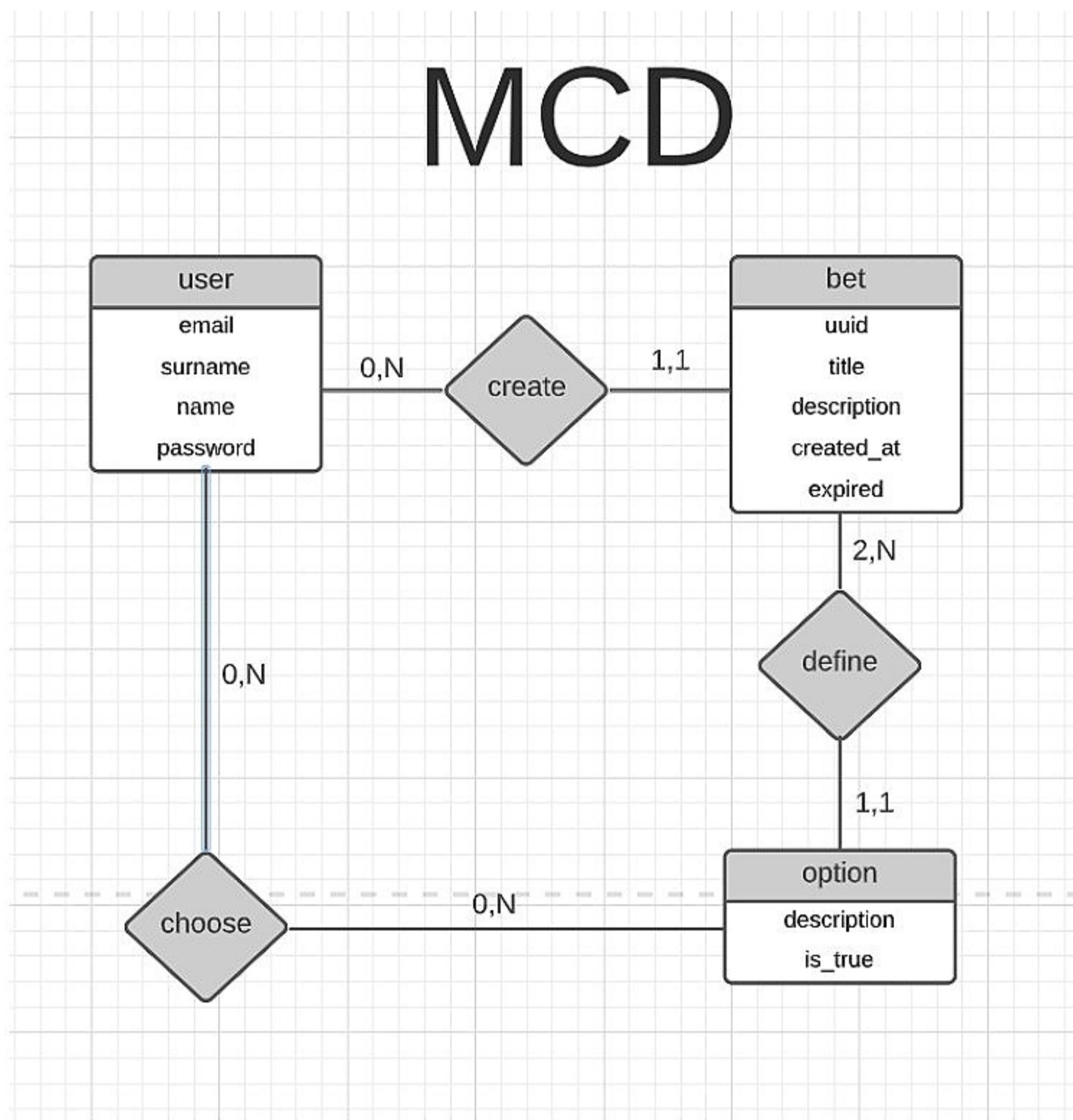
D'abord il a fallu imaginer tous les besoins des futurs utilisateurs de Côté Pote. A partir de ces besoins, j'ai été en mesure d'établir les règles de gestion des données à conserver.

Ensuite il faut définir le dictionnaire des données, c'est-à-dire toutes les données élémentaires qui vont être conservées en base de données et définir certaines caractéristiques qui figureront dans le MCD. Parmi ces caractéristiques, on retrouve par exemple la référence d'une donnée et notamment un identifiant unique, sa désignation, son type etc...

Étape finale à sa conception, on a pu à partir des informations précédemment recueillies, créer chaque entité, unique et décrite par un

DOSSIER PROFESSIONNEL (DP)

ensemble de propriétés et leurs associations permettant de définir les liens et cardinalités entre les entités.



Modèle Logique de données (MLD)

Le modèle logique de données (MLD), est une étape intermédiaire entre le modèle conceptuel de données et le modèle physique de données.

Le passage d'un MCD en MLD s'effectue selon quelques règles de conversion précise :

Une entité du MCD devient une table. Dans un SGBD de type relationnel, une table est une structure tabulaire dont chaque ligne correspond aux données d'un objet enregistré et où chaque colonne correspond à une propriété de cet objet.

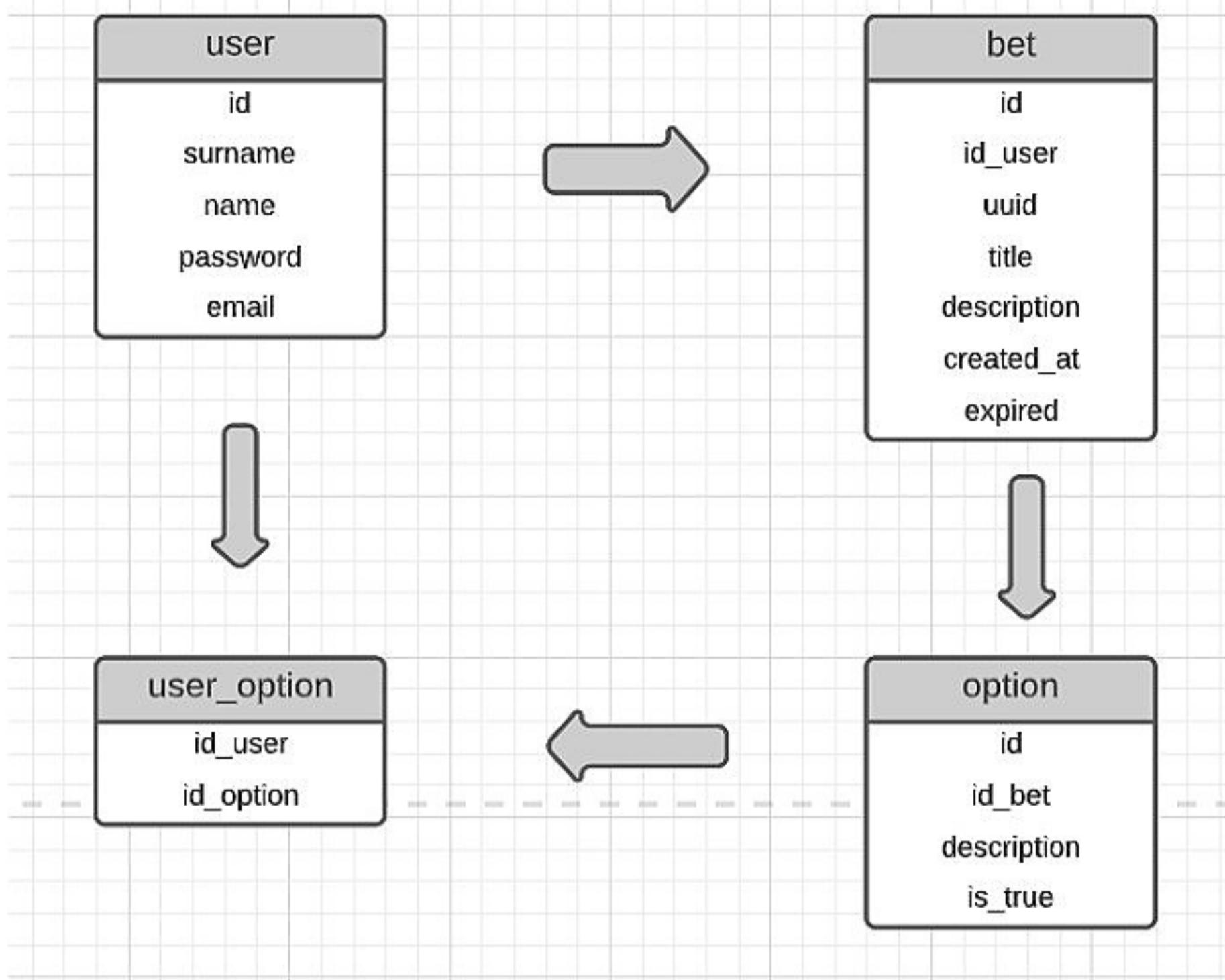
Ces colonnes font notamment référence aux caractéristiques définies dans le dictionnaire de données du MCD.

Les identifiants respectifs de chaque entité deviennent des clefs primaires et toutes les autres propriétés définies dans le MCD deviennent des attributs. Les clefs primaires permettent d'identifier de façon unique chaque enregistrement dans une table et ne peuvent pas avoir de valeur nulle.

Les cardinalités de type "0:n" / "1:n" sont représentées à travers le référencement de la clef primaire de la table qui la possède, en clef étrangère au sein de la table auquelle elle est liée. Si deux tables liées possèdent une cardinalité de type "0:n / 1:n", la relation sera traduite par la création d'une table de jonction, dont la clef primaire de chaque table deviendra une clef étrangère au sein de la table de jonction.

DOSSIER PROFESSIONNEL (DP)

MLD



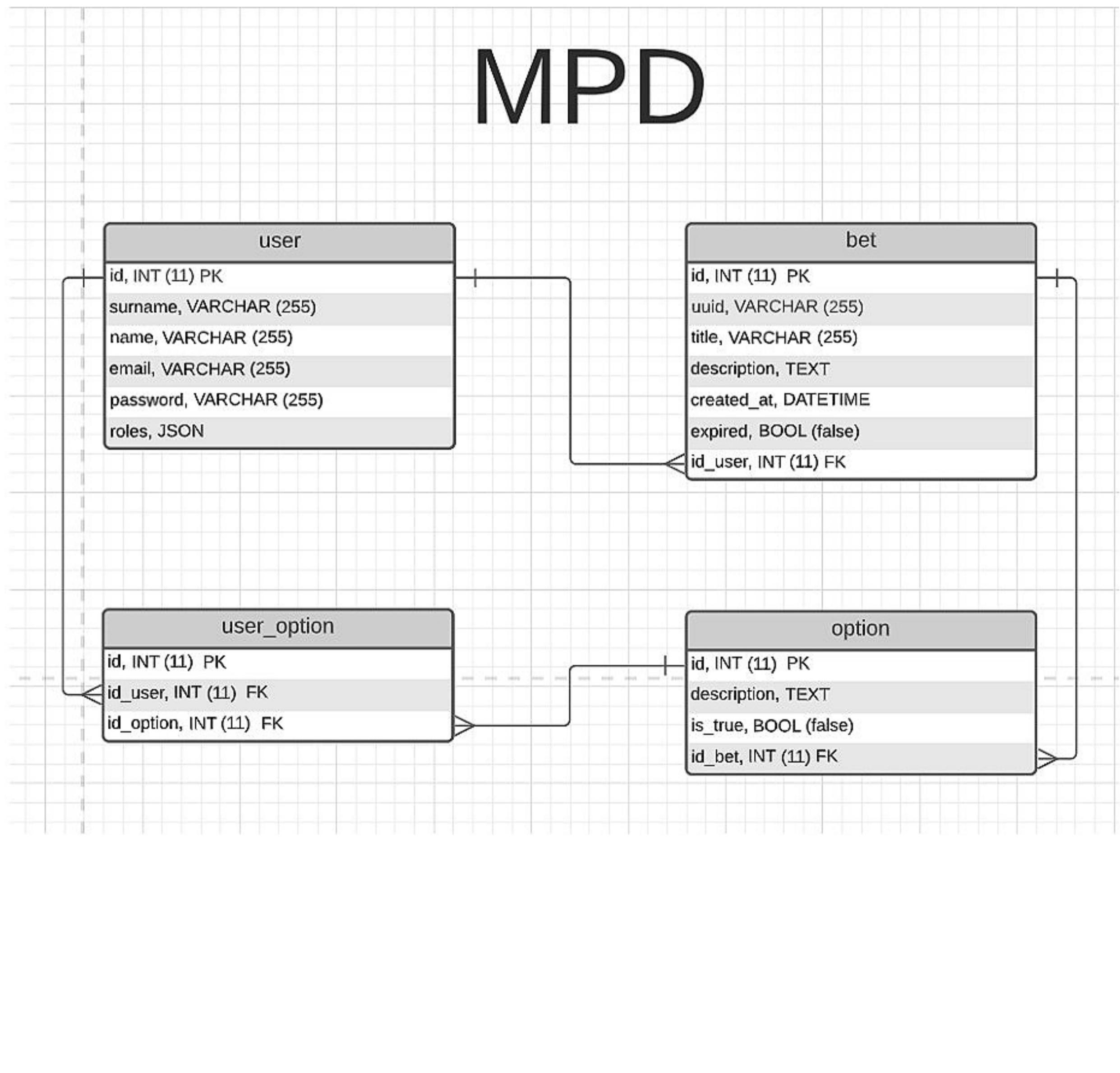
Ici, nous voyons bien que les clés primaires de chaque table associée deviennent des clé étrangère dans chacune des tables qui reçoivent les données.

DOSSIER PROFESSIONNEL (DP)

Modèle physique de données (MPD)

Étant presque une formalité, le MPD consiste à l'implémentation des modèles générés précédemment dans le Système de Gestion de Base de Données (SGBD) utilisé.

Dans notre cas, ce sera le SGBD MySQL couplé au moteur de stockage InnoDB qui permet la gestion des contraintes des clefs étrangères en base de données.



2. Précisez les moyens utilisés :

Les moyens utilisés pour ce projet :

Un ordinateur avec une connexion internet.

Pour faire la conception de ma base de données, j'ai utilisé Lucid App qui est une application intelligente de création de diagrammes et conception de S.I, et qui facilite grandement le travail collaboratif, entre autres je peux travailler en temps réel avec mes collègues de travail.

Pour la création de ma BDD j'ai utilisé une documentation sur Merise.

3. Avec qui avez-vous travaillé ?

Sur ce projet j'ai travaillé seul.

4. Contexte

Nom de l'entreprise, organisme ou association

La Plateforme_

Chantier, atelier, service

Période d'exercice

Du : 10/12/2023

25/01/2024

5. Informations complémentaires(*facultatif*)

pour la conception de ma base de données, j'ai utilisé la documentation suivante :

<https://ineumann.developpez.com/tutoriels/merise/initiation-merise/>

DOSSIER PROFESSIONNEL (DP)

Activité-type 2 Développer le back-end avec Symfony - Api Platform (API REST)

Exemple n° 2 - Symfony et Api Platform

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Tout d'abord,

API Platform est un framework web utilisé pour générer des API REST, se basant sur le patron de conception MVC (Modèle, Vue, contrôleur). La partie serveur du framework est écrite en PHP et basée sur le framework Symfony, tandis que la partie client est écrite en JavaScript et TypeScript

Doctrine ORM qui est un système de persistance est automatiquement livré dans la distribution API Platform.

Entre autres, Doctrine ORM est le moyen le plus simple de persister et d'interroger des données dans un projet API Platform grâce au pont fourni avec la distribution.

Doctrine Bridge est optimisé pour la performance et la commodité du développement. Par exemple, lors de l'utilisation de Doctrine, API Platform est capable d'optimiser automatiquement les requêtes SQL générées en ajoutant les JOIN clauses appropriées. Il fournit également de nombreux filtres intégrés puissants.

Doctrine ORM et son pont prennent en charge les SGBD les plus populaires, notamment PostgreSQL, MySQL, MariaDB, SQL Server, Oracle, SQLite et MongoDB ODM.

API Platform a une recette officielle Symfony Flex. Cela signifie que vous pouvez facilement l'installer depuis n'importe quelle application Symfony en utilisant le binaire Symfony :

Pour créer un nouveau projet Symfony :

```
symfony new nom-du-projet
```

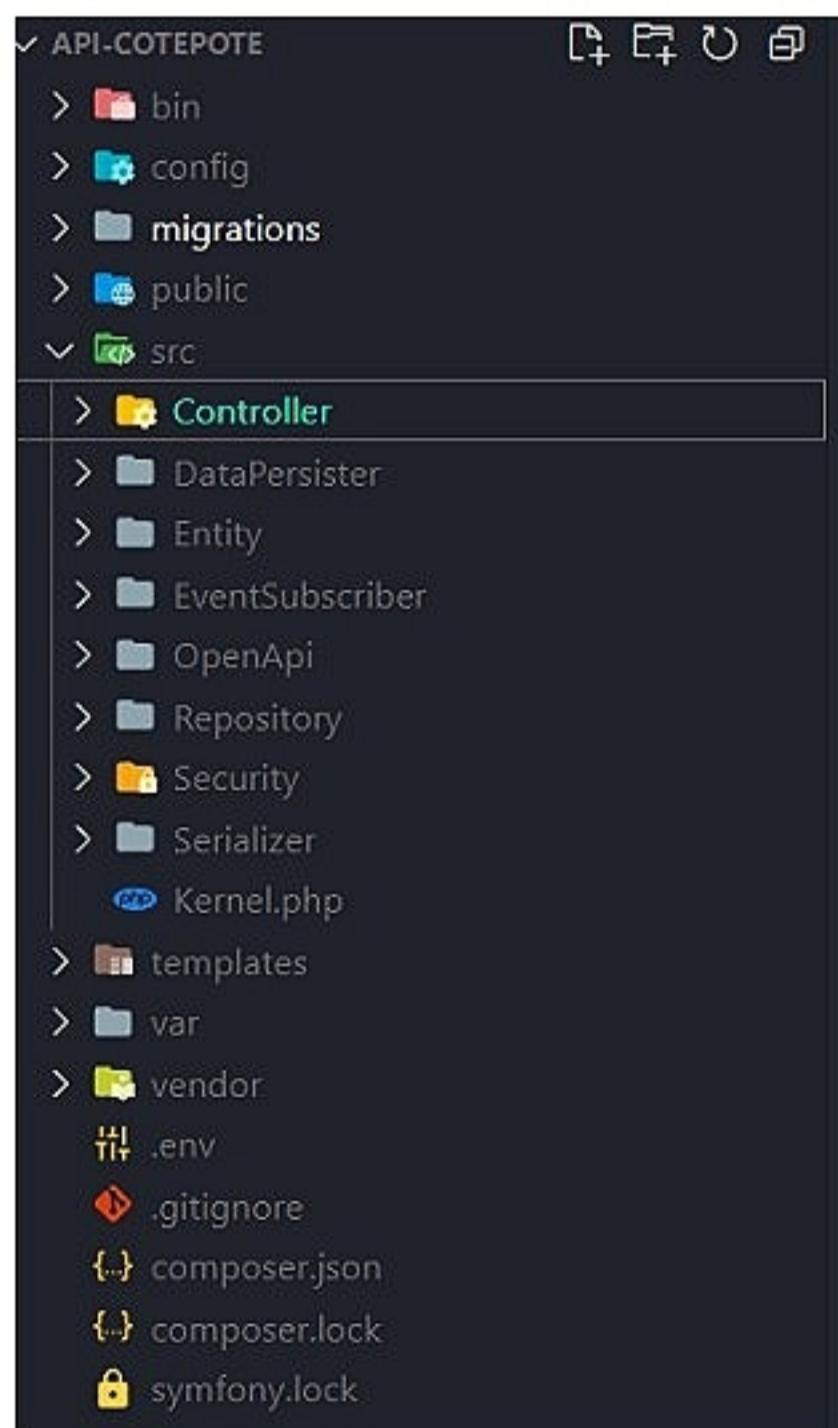
DOSSIER PROFESSIONNEL (DP)

Il faut d'abord s'assurer d'avoir l'environnement adéquat sur sa machine, entre autres WAMP Server et Composer pour ma part.

Pour Installer le composant serveur de la Api Platform dans ce squelette :

```
symfony composer require api
```

une fois ces deux commandes lancés, nous nous retrouvons avec l'architecture suivante :



DOSSIER PROFESSIONNEL (DP)

Ce qui est pratique avec ce type de framework, on peut aisément utiliser et télécharger des librairies pour pouvoir s'en servir par la suite sur mon projet, je peut retrouver simplement dans le fichier "composer.json" toutes les librairies et dépendances que j'ai sur mon projet :

```
{} composer.json > ...
1  {
2      "type": "project",
3      "license": "proprietary",
4      "minimum-stability": "stable",
5      "prefer-stable": true,
6      "require": {
7          "php": ">=8.0",
8          "ext-ctype": "*",
9          "ext-iconv": "*",
10         "api-platform/core": "^2.6",
11         "doctrine/annotations": "^1.0",
12         "doctrine/doctrine-bundle": "^2.5",
13         "doctrine/doctrine-migrations-bundle": "^3.2",
14         "doctrine/orm": "^2.11",
15         "gesdinet/jwt-refresh-token-bundle": "*",
16         "lexik/jwt-authentication-bundle": "^2.14",
17         "nelmio/cors-bundle": "^2.2",
18         "phpdocumentor/reflection-docblock": "^5.3",
19         "phpstan/phpdoc-parser": "^1.2",
20         "ramsey/uuid": "^4.2",
21         "symfony/asset": "5.4.*",
22         "symfony/console": "5.4.*",
23         "symfony/dotenv": "5.4.*",
24         "symfony/expression-language": "5.4.*",
25         "symfony/flex": "^1.17|^2",
26         "symfony/framework-bundle": "5.4.*",
27         "symfony/property-access": "5.4.*",
28         "symfony/property-info": "5.4.*",
29         "symfony/proxy-manager-bridge": "5.4.*",
30         "symfony/runtime": "5.4.*",
31         "symfony/security-bundle": "5.4.*",
32         "symfony/serializer": "5.4.*",
33         "symfony/twig-bundle": "5.4.*",
34         "symfony/validator": "5.4.*",
35         "symfony/yaml": "5.4.*",
36         "vich/uploader-bundle": "^1.19"
37     },
38     "config": {
39         "allow-plugins": {
40             "composer/package-versions-deprecated": true,
41             "symfony/flex": true,
42             "symfony/runtime": true
43         }
44     }
45 }
```

je peux très facilement en télécharger avec composer qui est un logiciel gestionnaire de dépendances libre écrit en PHP. Il permet à ses utilisateurs de

DOSSIER PROFESSIONNEL (DP)

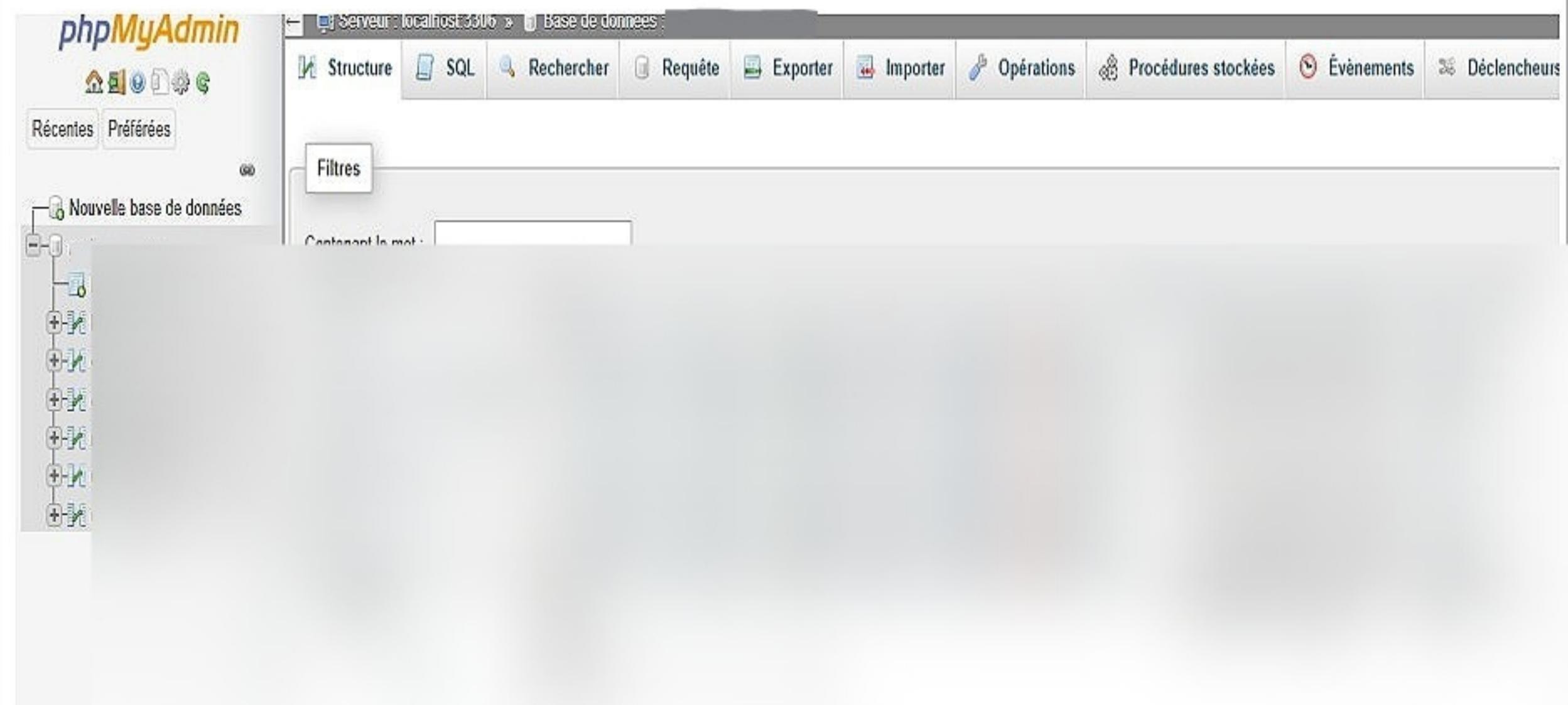
déclarer et d'installer les bibliothèques dont le projet principal a besoin. pour la suite de la création de notre api, on se rend dans le fichier ".env" de notre projet pour modifier la ligne " DATABASE_URL", cela nous permet de dire où est ce que doctrine va créer notre base de données ainsi que l'emplacement où sera faites les migrations :

```
DATABASE_URL="mysql://root:@127.0.0.1:3306/apicotepte?serverVersion=5.7"
```

Une fois cela fait, je n'ai plus qu'à créer ma base de données dans le SGBD et son schéma dans mon projet avec les lignes de commandes suivantes :

```
symfony console doctrine:database:create  
symfony console doctrine:schema:create
```

a là suite de ces lignes de commandes, je me retrouve avec ma base de données créé dans mon SGBD :



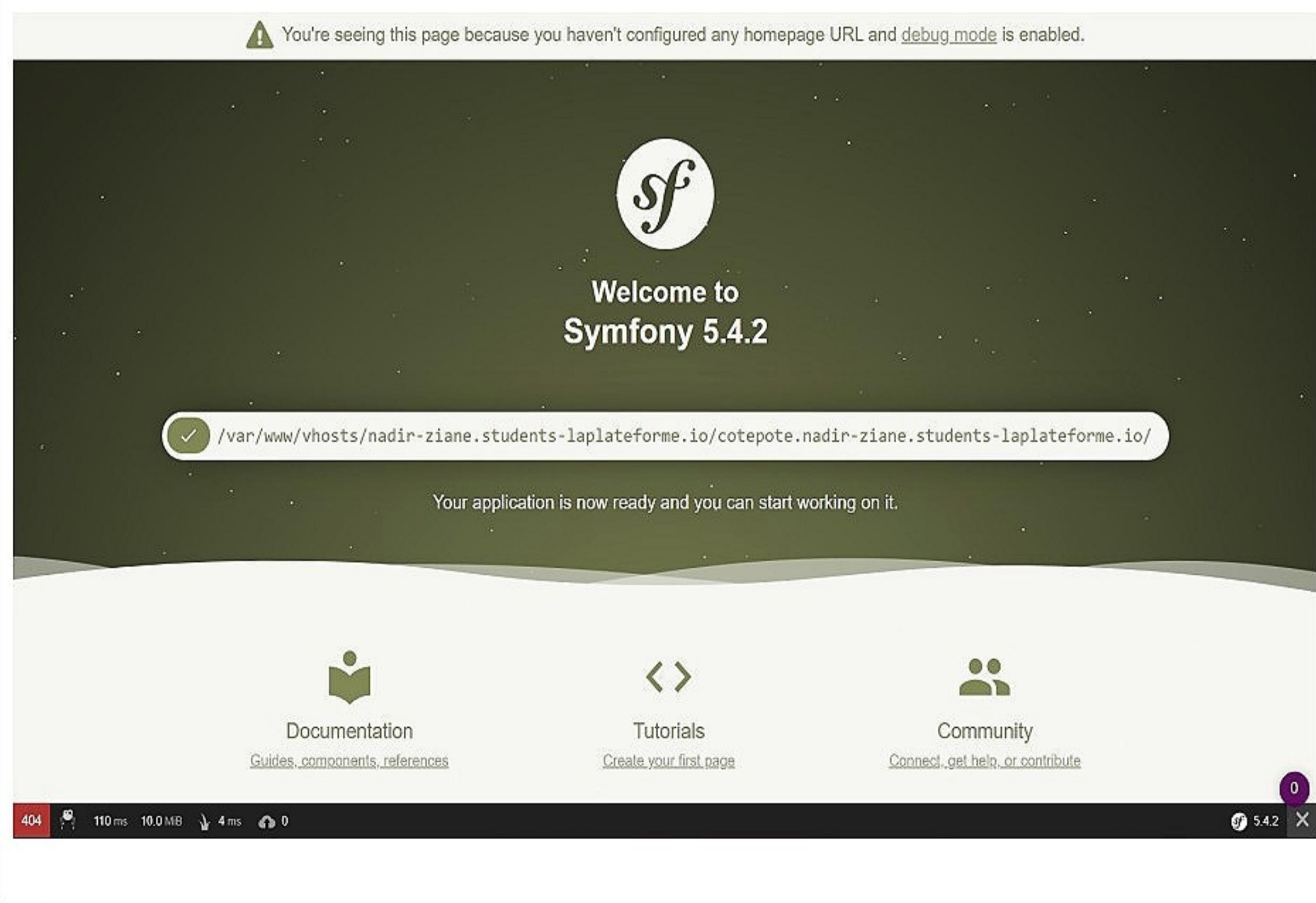
DOSSIER PROFESSIONNEL (DP)

Pour vérifier si toutes les actions que j'ai effectués ont bien fonctionnés, je démarre le serveur PHP intégré avec la ligne de commande suivante :

```
symfony serve
```

Pour vérifier si cela à bien fonctionné, j'ouvre <https://localhost> Dans mon navigateur.

Si cela fonctionne bien, il y aura ça sur le navigateur :



DOSSIER PROFESSIONNEL (DP)

Lors de l'installation de la Api Platform de cette manière, l'API sera exposée en tant que "/api/".

Donc j'ouvre <http://localhost:8000/api/> pour voir la documentation de l'API.

The screenshot shows the API Platform documentation interface. At the top, there's a header with the API PLATFORM logo and a "0.0.0 OAS3" button. Below the header, there's a "Servers" dropdown set to "/" and an "Authorize" button. The main content area is titled "Bet" and lists various HTTP methods and their corresponding URLs:

Method	URL	Description	Actions
GET	/api/bets	Retrieves the collection of Bet resources.	▼ 🔒
POST	/api/bets	Creates a Bet resource.	▼
GET	/api/bets/{id}	Retrieves a Bet resource.	▼
DELETE	/api/bets/{id}	Removes the Bet resource.	▼
PATCH	/api/bets/{id}	Updates the Bet resource.	▼
POST	/api/bets/{id}/image	Creates a Bet resource.	▼
POST	/api/bets/{id}/setExpired	Permet de set un pari en expired	▼

Below the Bet section, there's a "Dependency" section with a single GET operation:

Method	URL	Description	Actions
GET	/api/dependencies	Retrieves the collection of Dependency resources.	▼

At the bottom of the interface, there's a performance metrics bar showing 200 requests, @api_entrypoint, 311 ms, 14.0 MiB, 244 in 44.36 ms, n/a, 8 ms, 0 errors, and a version 5.4.2.

API Platform expose une description de l'API au format OpenAPI (anciennement connu sous le nom de Swagger). Il intègre également une version personnalisée de Swagger UI , une belle interface restituant la documentation OpenAPI. Il faut cliquer sur une opération pour afficher ses détails. et je peux également envoyer des requêtes à l'API directement depuis l'interface utilisateur.

Mon projet API Platform est désormais 100% fonctionnel, il ne reste plus qu'à implémenter mes différentes entités et y mettre les annotations nécessaires pour le bon fonctionnement de mon api

Dossier Professionnel (DP)

2. Précisez les moyens utilisés :

Les moyens utilisés pour ce projet :

Un ordinateur avec une connexion internet.

Comme IDE j'utilise Visual Studio Code avec son terminal intégré pour taper toutes mes lignes de commandes.

Pour faire la création de mon API, j'ai utilisé Symfony ainsi que API Platform, tout simplement en suivant la documentation de API Platform ainsi que des tutoriels youtube.

3. Avec qui avez-vous travaillé ?

J'ai travaillé seul sur ce projet.

4. Contexte

Nom de l'entreprise, organisme ou association

(DP)

Chantier, atelier, service

Dans le cadre de la formation concepteur /développeur d'application

Période d'exercice

Du : 10/12/2023

au : 25/01/2024

5. Informations complémentaires(facultatif)

suite de tutoriel youtube pour la création complète d'une API REST avec API Platform :

 Découverte d'API Platform : Qu'est ce qu'API Platform

la documentation dont je me suis servi :

<https://api-platform.com/docs/distribution/>

DOSSIER PROFESSIONNEL (DP)

Activité-type 2 Développer le back-end avec Symfony - Api Platform (API REST)

Exemple n° 3 - création et paramétrage de mes entités

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Une fois l'étape de la mise en place de mon projet en Symfony - Api Platform réalisé, la première étape que j'ai effectué est la création de mes entités, c'est là où Doctrine ORM rentre en jeu :

Doctrine facilite grandement la création de mes entités ainsi que mes champs, il prend tout en compte pour la création / modélisation de ma base de données. A l'aide de quelques lignes de commande je crée mon modèle de données, gère la persistance (la sauvegarde) dans une table.

Voici quelques commandes utiles pour rendre la création d'entité plus fluide (on dit streamline en anglais), création d'une entité, création de la table correspondante opération nommée migration et pour accepter une valeur par défaut proposée appuyer sur Entrée.

Tout d'abord :

```
php bin/console make:entity
```

Cette ligne de commande me sert à créer une entité, l'étape suivante est de choisir le nom de cette entité (ou le nom de l'entité que je souhaite modifier).

```
Class name of the entity to create or update (e.g.  
AgreeableJellybean):  
> Product  
created: src/Entity/Product.php  
created: src/Repository/ProductRepository.php  
Entity generated! Now let's add some fields!  
You can always add more fields later manually or by re-running this  
command.
```

DOSSIER PROFESSIONNEL (DP)

l'étape qui suit est la création de mes champs, cela se fait en quelques étapes :

```
New property name (press <return> to stop adding fields):
> name
Field type (enter ? to see all types) [string]:
> string
Field length [255]:
>
Can this field be null in the database (nullable) (yes/no) [no]:
>
updated: src/Entity/Product.php
```

tout d'abord le type (si c'est une chaîne, un entier, un booléen, un datetime)

dans mon cas c'est une string et par là suite je dois choisir la longueur de la chaîne de caractère et enfin si elle peut être nul.

je n'ai plus qu'à répéter l'opération pour chaque champ de mon entité.

une fois la création de mon entité effectué je n'ai plus qu'à faire une migration pour que ma base de données dans mon SGBD (Système de Gestion de Base de Données) se met à jour :

cela se fait en une ligne de commande :

```
php bin/console make:migration
```

Maintenant je fais la migration proprement dite, cette migration est possible que si dans le fichier .env, il y a la chaîne de connexion bien remplie avec les identifiant et mot de passe, nom de la base de donnée et du host.

DOSSIER PROFESSIONNEL (DP)

Il suffit de lancer la ligne de commande suivante :

```
php bin/console doctrine:migrations:migrate
WARNING! You are about to execute a database migration that could
result in schema changes and data loss. Are you sure you wish to
continue? (y/n)y
Migrating up to 20190612091941 from 0

++ migrating 20190612091941

    -> CREATE TABLE product (id INT AUTO_INCREMENT NOT NULL, name
VARCHAR(255) NOT NULL, price INT NOT NULL, PRIMARY KEY(id)) DEFAULT
CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci ENGINE = InnoDB

++ migrated (took 128.9ms, used 12M memory)

-----
++ finished in 133.7ms
++ used 12M memory
++ 1 migrations executed
++ 1 sql queries
```

maintenant je me retrouve avec ma table en base de données :

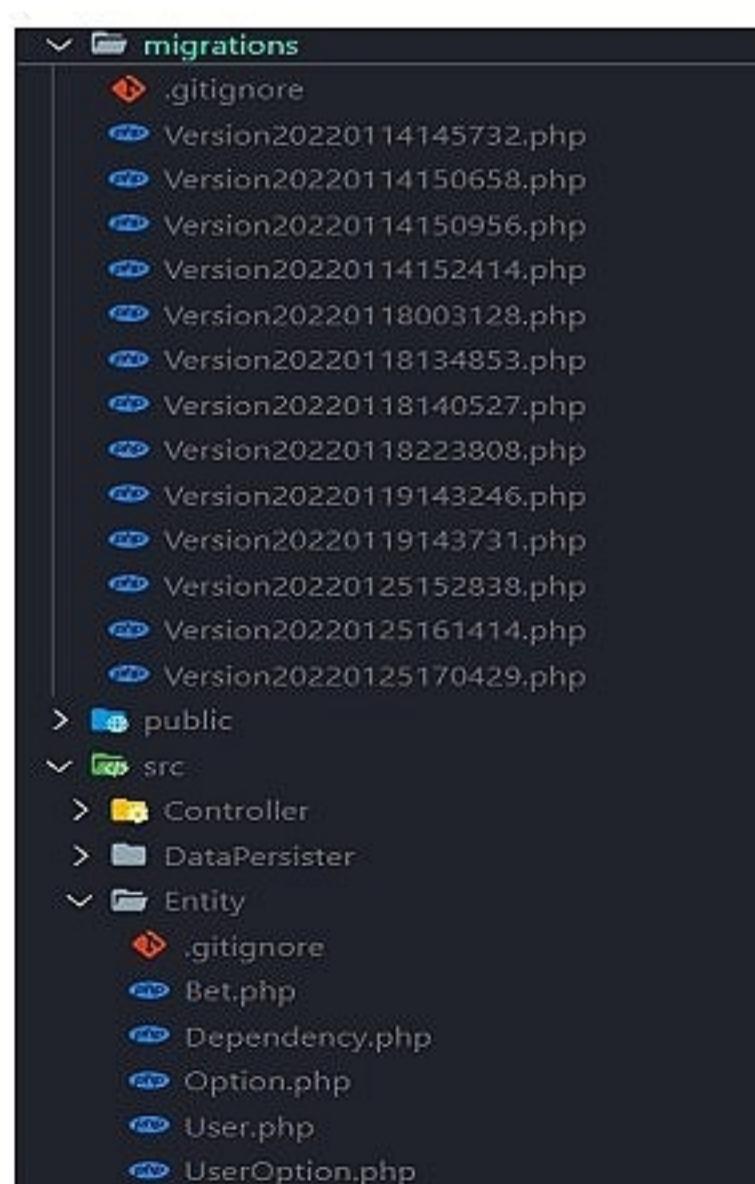
The screenshot shows the phpMyAdmin interface with the 'user' table selected. The left sidebar shows a tree view of databases and tables, with 'Nouvelle base de données' expanded. The main area displays the 'Structure de table' tab for the 'user' table, which has 8 columns: id, email, roles, password, name, surname, file_path, and updated_at. Each column has its type, length, and constraints defined.

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action
1	id	int(11)			Non	Aucun(e)		AUTO_INCREMENT	Modifier Supprimer Plus
2	email	varchar(180)	utf8mb4_unicode_ci		Non	Aucun(e)			Modifier Supprimer Plus
3	roles	longtext	utf8mb4_unicode_ci		Non	Aucun(e)			Modifier Supprimer Plus
4	password	longtext	utf8mb4_unicode_ci		Non	Aucun(e)			Modifier Supprimer Plus
5	name	varchar(255)	utf8mb4_unicode_ci		Non	Aucun(e)			Modifier Supprimer Plus
6	surname	varchar(255)	utf8mb4_unicode_ci		Non	Aucun(e)			Modifier Supprimer Plus
7	file_path	longtext	utf8mb4_unicode_ci		Oui	NULL			Modifier Supprimer Plus
8	updated_at	datetime			Oui	NULL			Modifier Supprimer Plus

DOSSIER PROFESSIONNEL (DP)

et je vois également que dans ma table "doctrine_migration_versions" une ligne est ajouté à chaque migration :

une fois toutes ces étapes réalisées, je vois que dans l'architecture de mes fichiers est mis également à jour, un fichier s'ajoute dans le dossier migrations et également un fichier sera ajouter dans le dossier pour les entités :



DOSSIER PROFESSIONNEL (DP)

Également dans l'interface swagger de mon API je vois qu'une nouvelle collection de routes API concordante à l'entité que je viens de créer et de migrer avec ces méthodes (GET, POST, DELETE, PUT, PATCH), le CRUD de base entre autres, avec la possibilité de tester mes routes, voir quelles informations sont renvoyées après l'appel API que je fais :

The screenshot shows the API Platform Swagger interface. At the top, there's a teal header bar with the API PLATFORM logo. Below it, a navigation bar includes a 'Servers' dropdown set to '/', an 'Authorize' button with a lock icon, and a search bar. The main content area is titled 'Bet' and lists seven endpoints:

Method	Path	Description	Lock
GET	/api/bets	Retrieves the collection of Bet resources.	🔒
POST	/api/bets	Creates a Bet resource.	🔓
GET	/api/bets/{id}	Retrieves a Bet resource.	🔓
DELETE	/api/bets/{id}	Removes the Bet resource.	🔓
PATCH	/api/bets/{id}	Updates the Bet resource.	🔓
POST	/api/bets/{id}/image	Creates a Bet resource.	🔓
POST	/api/bets/{id}/setExpired	Permet de set un pari en expiré	🔓

DOSSIER PROFESSIONNEL (DP)

POST /api/bets Creates a Bet resource.

Creates a Bet resource.

Parameters

Try it out

No parameters

Request body required

application/json

The new Bet resource

Example Value | Schema

```
{  
  "title": "string",  
  "description": "string",  
  "user": "string",  
  "options": [  
    {  
      "description": "string"  
    }  
  ]  
}
```

Responses

Code	Description	Links
201	Bet resource created	<p>GetBetItem</p> <p>The <code>id</code> value returned in the response can be used as the <code>id</code> parameter in <code>GET /api/bets/{id}</code>.</p> <p>Operation 'getBetItem'</p> <p>Parameters { "id": "\$response.body#/id" }</p>

DOSSIER PROFESSIONNEL (DP)

Globalement, ce qui va nous intéresser c'est le fichier qui sera ajouter dans le dossier entité :

c'est tout simplement la classe qui va gérer toute notre table et depuis laquelle je vais pouvoir ajouter des annotations pour configurer mes routes API

Cela va comprendre majoritairement le type de sérialisation et désérialisation que je vais choisir :

La sérialisation est ce que mon api va me retourner, et la désérialisation est ce que j'envoie à mon API.

cela est gérée dans le contexte de normalisation et de dénormalisation :

En ce qui concerne le contexte de dénormalisation, cela va définir quelle données je souhaite intercepter de mon API, cela va se faire à travers les annotations dans ma classe :

via ce type d'écriture commenter, API Platform va tout simplement lire les annotations au sein de "@ApiResource(...)" pour voir les règles concernant les routes API, ce qui va principalement définir les règles de normalisation ainsi que de dénormalisation de mes routes sont les deux suivantes :

```
/**
 * @ORM\Entity(repositoryClass=BetRepository::class)
 * @ApiResource(
 *     normalizationContext = {"groups" = {"read:bet"}},
 *     denormalizationContext = {"groups" = {"create:bet"}},
 *     itemOperations = {"get", "put", "patch", "delete"},  
     collectionOperations = {"post", "get"}  
 )
```

À l'intérieur, je vais tout simplement dire quel groupe de lecture j'affecte à l'un et à l'autre pour savoir quels éléments sont concernés dans ma base de données.

DOSSIER PROFESSIONNEL (DP)

Ces noms de groupes, je n'ai plus qu'à les rappeler au dessus des attributs dans ma classe pour que je décide lesquels seront concernées, la manière est la suivante :

```
class Bet
{
    /**
     * @ORM\Id
     * @ORM\GeneratedValue
     * @ORM\Column(type="integer")
     * @Groups({"read:bet"})
     */
    private $id;

    /**
     * @ORM\Column(type="string", length=255)
     * @Groups({"read:bet"})
     */
    private $uuid;

    /**
     * @ORM\Column(type="string", length=255)
     * @Groups({"read:bet", "create:bet"})
     * @Assert\Length(
     *     min = 5,
     *     max = 255
     * )
     */
    private $title;

    /**
     * @ORM\Column(type="text")
     * @Groups({"read:bet", "create:bet"})
     * @Assert\Length(
     *     min = 5,
     *     max = 255
     * )
     */
}
```

je met dans l'annotation “@Groups()” l'attribut concerné pour que je puisse le retrouver dans mon contexte de normalisation, ou celui de dénormalisation.

Dossier Professionnel (DP)

2. Précisez les moyens utilisés :

Les moyens utilisés pour ce projet :

Un ordinateur avec une connexion internet.

Comme IDE j'utilise Visual Studio Code avec son terminal intégré pour taper toutes mes lignes de commandes.

Pour faire la création de mon API, j'ai utilisé Symfony ainsi que API Platform, tout simplement en suivant la documentation de API Platform ainsi que des tutoriels youtube.

3. Avec qui avez-vous travaillé ?

J'ai travaillé seul sur ce projet.

4. Contexte

Nom de l'entreprise, organisme ou association

La Plateforme

Chantier, atelier, service

Dans le cadre de ma formation de concepteur /développeur d'application

Période d'exercice

Du : 10/12/2023

au : 25/01/2024

5. Informations complémentaires(facultatif)

suite de tutoriel youtube pour la création complète d'une API REST avec API Platform :

 Découverte d'API Platform : Qu'est ce qu'API Platform

la documentation dont je me suis servi :

<https://api-platform.com/docs/distribution/>

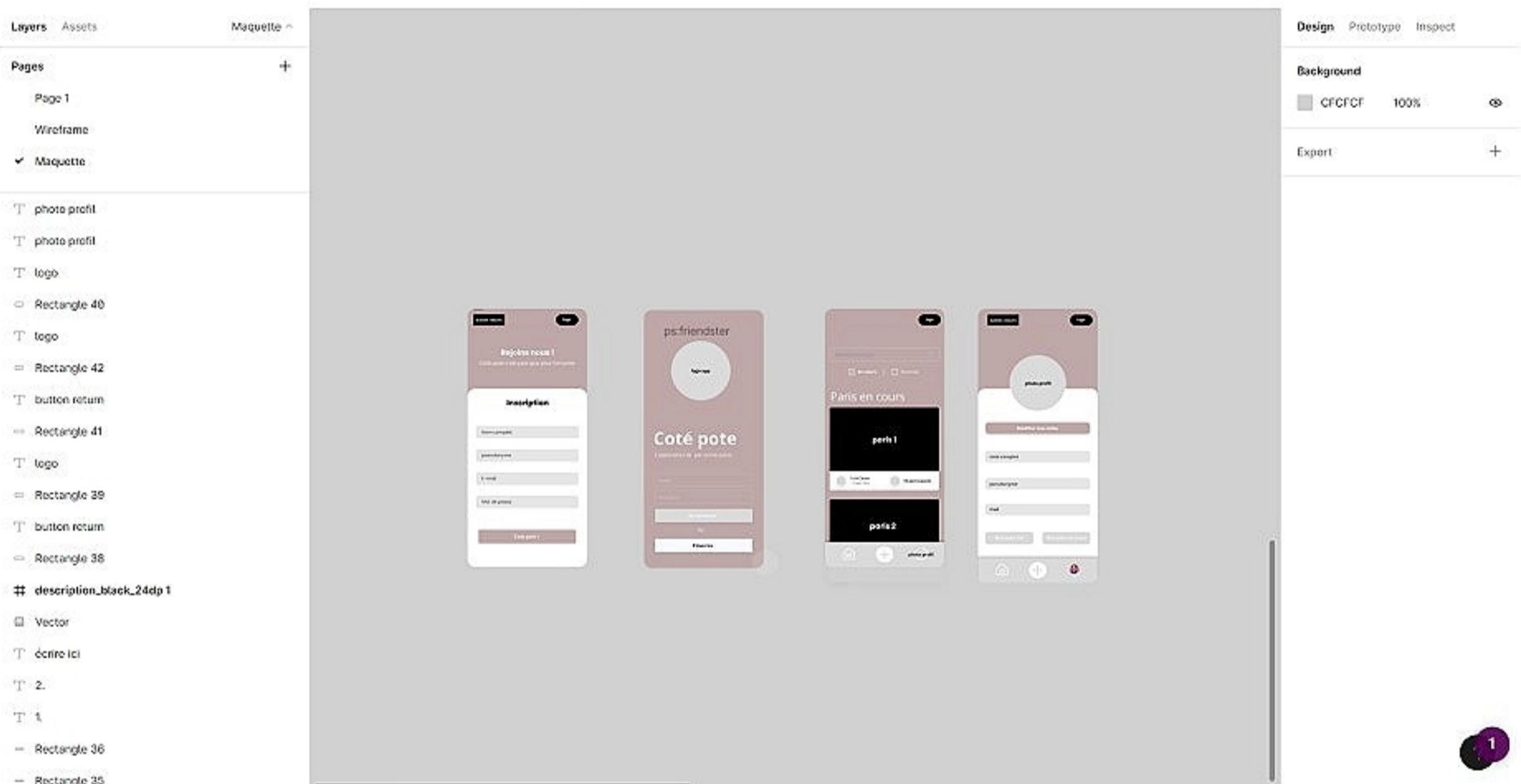
DOSSIER PROFESSIONNEL (DP)

Activité-type 3 Développer le front-end avec Expo - React Native (App mobile)

Exemple n° 1 - Conception et maquettage d'une application

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Dans un premier temps, je commence par rédiger une **maquette design**. j'ai appris à faire un **wireframe** pour visualiser comment seront agencer les éléments sur les différentes vues, pour la création de ce wireframe je me sers de Figma qui est un éditeur de graphiques vectoriels et un outil de prototypage. Il est principalement basé sur le web :

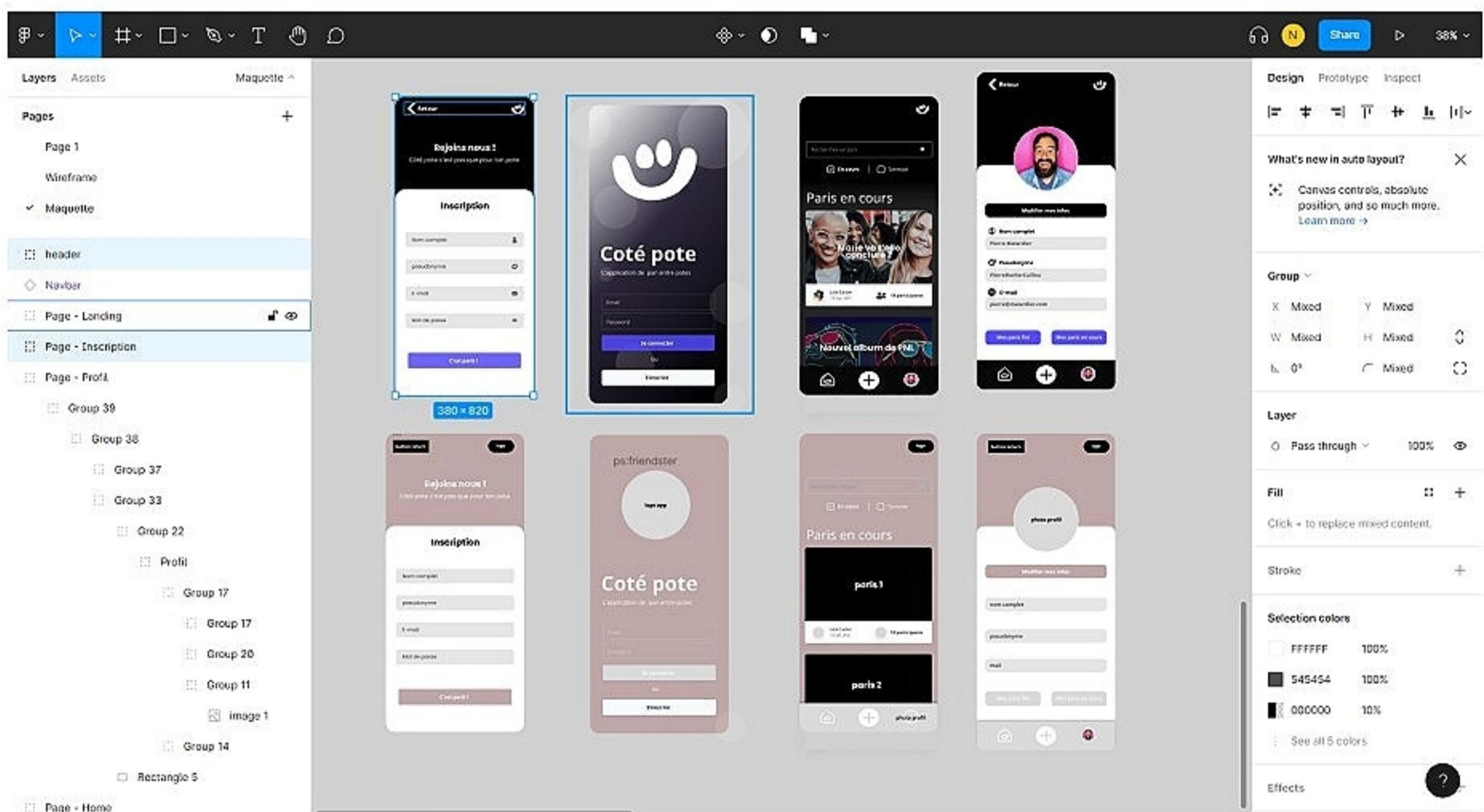


j'ai choisi un **design minimaliste**, qui est très en vogue actuellement dans l'univers du web.

DOSSIER PROFESSIONNEL (DP)

Ensute je fais une maquette où je défini la **charte graphique**. Entre autres **les couleurs, les logos et les polices utilisées**, pour que l'application ait une identité visuelle, qu'elle soit **attrayante visuellement pour l'utilisateur**. Puisque, l'utilisateur ne voit et ne se fie qu'à l'apparence de l'application et ne peut pas voir le Back-end.

j'ai ici l'exemple de la maquette que j'ai réalisé pour la même application que je présente pour l'exemple précédent du wireframe :



DOSSIER PROFESSIONNEL (DP)

2. Précisez les moyens utilisés :

j'ai utilisé pour la création de la maquette une application web nommée figma, qui est très utile pour le travail collaboratif, j'aurai pu utiliser aussi Adobe XD, j'ai également utilisé shutter stock ainsi que Flaticon pour trouver des logos et icônes.

Nous avons travaillé via Trello qui est un outil de gestion de projet en ligne, il repose sur une organisation des projets en planches listant des cartes, chacune représentant des tâches, et d'autres supports de type Google Suit.

3. Avec qui avez-vous travaillé ?

J'ai travaillé seul sur ce projet.

4. Contexte

Nom de l'entreprise, organisme ou association

Cliquez ici pour taper du texte.

Chantier, atelier, service

Dans le cadre de ma formation de concepteur / développeur d'application

Période d'exercice

Du : 25/01/2024

au : 27/05/2024

5. Informations complémentaires(facultatif)

DOSSIER PROFESSIONNEL (DP)

Activité-type 3 Développer le front-end avec Expo - React Native (App mobile)

Exemple n° 2 - React Native - Expo

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Tout d'abord,

React Native est un framework d'applications mobiles open source créé par Facebook. Il est utilisé pour développer des applications pour Android, iOS et UWP en permettant aux développeurs d'utiliser React avec les fonctionnalités natives de ces plateformes.

Pour pouvoir mettre en place l'environnement de développement pour React Native où plus généralement pour la grande majorité des Framework Front-End, je sais que j'ai besoin de NodeJS qui est une plateforme logicielle libre en JavaScript, orientée vers les applications réseau événementielles.

Il y a également NPM ou Node Packet Manager, écrit en grande partie en JavaScript, est indissociable du succès de node.

Il permet de gérer ou bien de publier de nouveaux logiciels au sein de l'écosystème NodeJS.

maintenant que nous avons vu ces quelques principes, on va s'intéresser à Expo :

Expo est tout simplement un ensemble d'outils gratuit et open source développé autour de React Native qui permet de gagner du temps dans le développement d'applications iOS et Android.

Cela facilite grandement l'émulation de mon application sur tout type d'émulateur, entre autres sur Navigateur, sur Android Studio, sur Xcode (à condition d'être sur MacOS) et même directement sur son téléphone en téléchargeant l'application Expo sur les stores.

DOSSIER PROFESSIONNEL (DP)

Pour créer un nouveau projet Expo React Native :

Il faut d'abord s'assurer d'avoir EXPO-CLI installé dans mes variables d'environnement système, donc je commence par l'installer avec npm :

```
npm install --global expo-cli
```

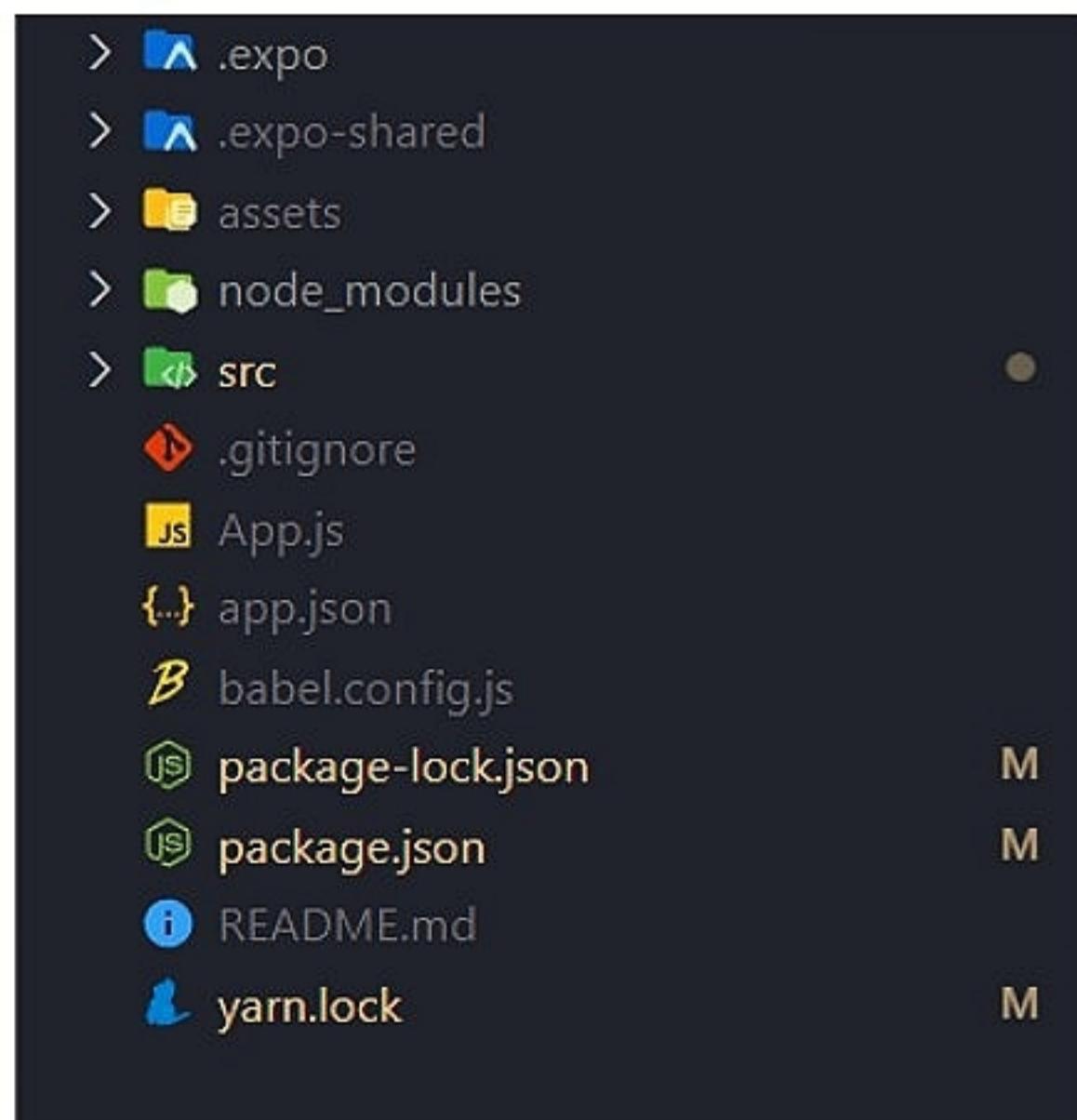
Expo CLI est une application en ligne de commande qui est l'interface principale entre un développeur et les outils Expo.

Expo CLI dispose également d'une interface graphique Web qui apparaît dans mon navigateur Web lorsque je démarre mon projet.

Maintenant que j'ai Expo CLI installer de manière global sur ma machine, je crée un nouveau projet :

```
expo init my-app
```

Une fois cette opération faite, je me retrouve avec une architecture de ce genre :



DOSSIER PROFESSIONNEL (DP)

Ce qui est pratique une fois de plus avec ce type de framework, je peut aisément utiliser et télécharger des librairies pour pouvoir s'en servir par la suite sur mon projet, je peut retrouver simplement dans le fichier "composer.json" toutes les librairies et dépendances que j'ai sur mon projet :

```
package.json > private
1  {
2    "name": "app-cotepote",
3    "version": "1.0.0",
4    "main": "node_modules/expo/AppEntry.js",
5    "scripts": {
6      "start": "expo start",
7      "android": "expo start --android",
8      "ios": "expo start --ios",
9      "web": "expo start --web",
10     "eject": "expo eject"
11   },
12   "dependencies": {
13     "@expo-google-fonts/dev": "^0.2.2",
14     "@expo-google-fonts/poppins": "^0.2.2",
15     "@react-native-async-storage/async-storage": "^1.15.17",
16     "@react-native-community/masked-view": "^0.1.11",
17     "@react-navigation/bottom-tabs": "^6.0.9",
18     "@react-navigation/drawer": "^6.3.1",
19     "@react-navigation/native": "^6.0.6",
20     "@react-navigation/stack": "^6.0.11",
21     "axios": "^0.25.0",
22     "buffer": "^6.0.3",
23     "expo": "~44.0.0",
24     "expo-app-loading": "~1.3.0",
25     "expo-constants": "~13.0.1",
26     "expo-font": "~10.0.4",
27     "expo-google-fonts": "^0.0.0",
28     "expo-image-picker": "~12.0.1",
29     "expo-secure-store": "~11.1.0",
30     "expo-status-bar": "~1.2.0",
31     "react": "17.0.1",
32     "react-dom": "17.0.1",
33     "react-native": "0.64.3",
34     "react-native-gesture-handler": "~2.1.0",
35     "react-native-keyboard-aware-scroll-view": "^0.9.5",
36     "react-native-reanimated": "2.3.1",
37     "react-native-safe-area-context": "^3.3.2",
38     "react-native-screens": "~3.10.1",
39     "react-native-web": "0.17.1",
40     "save": "^2.4.0"
41   }
}
```

je peux très facilement en télécharger avec npm qui est un logiciel gestionnaire de dépendances comme par exemple axios dont je me sers comme client HTTP, ce qui me permet de communiquer avec des API en utilisant des requêtes.

DOSSIER PROFESSIONNEL (DP)

une fois la mise en place de l'architecture de mon projet réalisé, si je veux le lancer je n'ai plus qu'à lancer le serveur de la manière suivante :

expo start

une fois cela fait je n'ai plus qu'à me rendre sur le lien localhost :

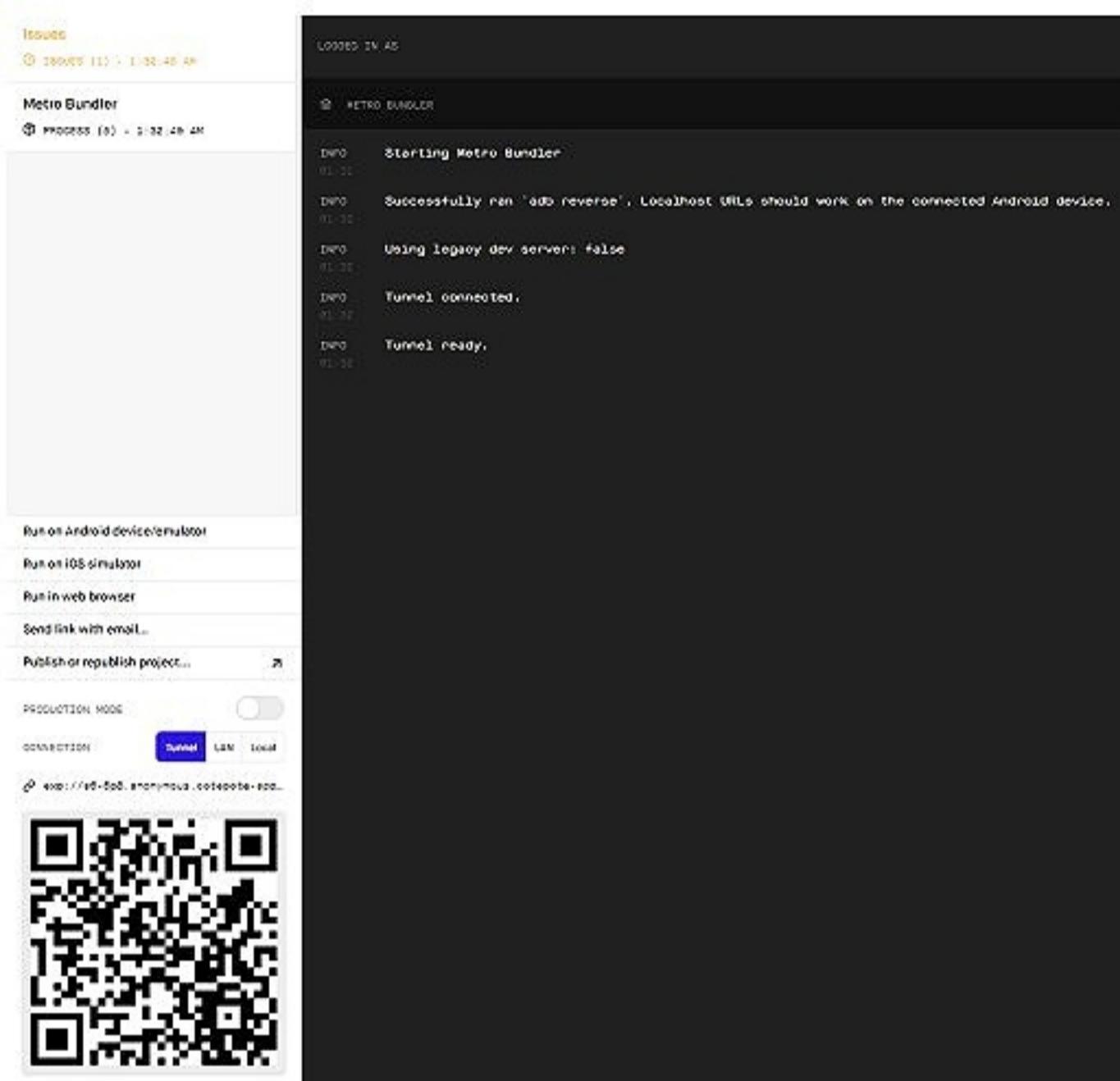
```
> expo start

There is a new version of expo-cli available (5.4.11).
You are currently using expo-cli 5.4.8
Install expo-cli globally using the package manager of your choice;
for example: `npm install -g expo-cli` to get the latest version

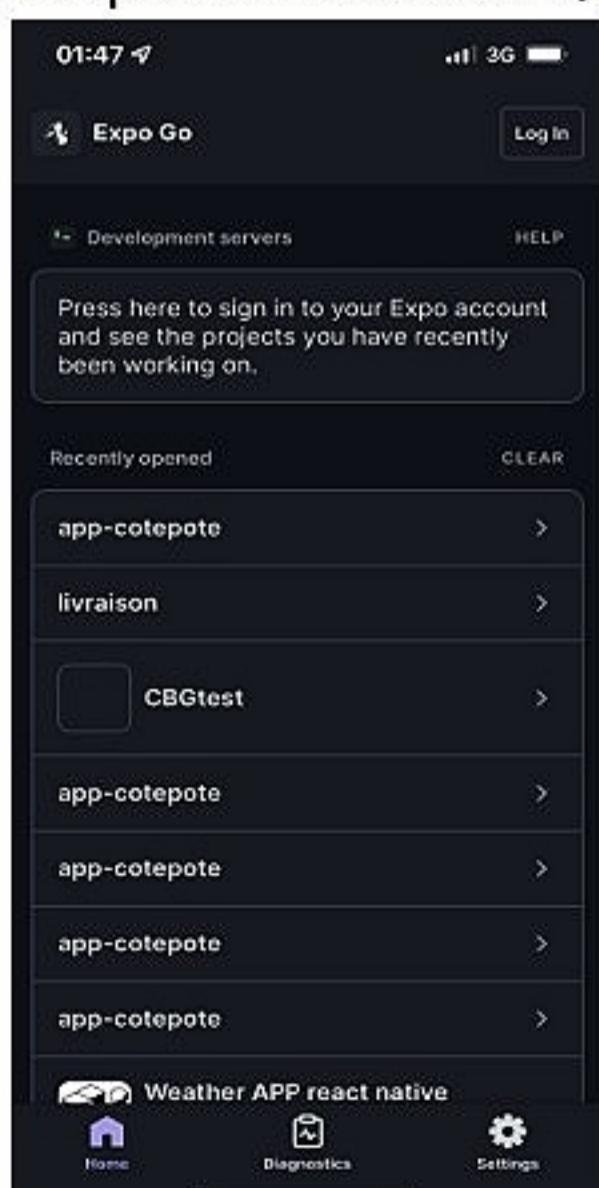
Developer tools running on http://localhost:19002
Starting Metro Bundler


```

DOSSIER PROFESSIONNEL (DP)



je décide de lancer la connexion par tunnel pour pouvoir émuler l'application sur mon mobile et je dois m'assurer que ma machine et mon smartphone sont bien sûr sur le même réseau, et je n'ai plus qu'à scanner le QR code et j'ai accès à mon appli



DOSSIER PROFESSIONNEL (DP)

et je lance mon appli qui porte le nom app-cotepote, et je me retrouve avec l'application lancer sur mon smartphone :



Dossier Professionnel (DP)

2. Précisez les moyens utilisés :

j'ai utilisé pour la création de mon application mobile NodeJS ainsi que NPM.

Comme IDE j'utilise Visual Studio Code.

Comme logiciel de versionning j'utilise Git Kraken couplé à GitHub pour mettre à jour mon repository.

Comme application de gestion de projet j'utilise Trello.

Comme FrameWork, j'ai utilisé React Native Expo.

j'utilise mon SmartPhone avec l'application Expo Go.

3. Avec qui avez-vous travaillé ?

j'ai travaillé seul sur ce projet.

4. Contexte

Nom de l'entreprise, organisme ou association

La Plateforme

Chantier, atelier, service

Dans le cadre de la formation concepteur / développeur d'application

Période d'exercice

Du : **25/01/2024**

au : **27/05/2024**

5. Informations complémentaires (facultatif)

DOSSIER PROFESSIONNEL (DP)

Activité-type 3 Développer le front-end avec Expo - React Native (App mobile)

Exemple n° 3 - *Création de composants et hooks*

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Tout d'abord il faut savoir que toutes les vues seront gérées à partir de mon fichier "App.js".

C'est donc à partir d'ici que je commence :

The screenshot shows a code editor with a dark theme. On the left, there is a sidebar titled 'EXPLORATEUR' which lists the project structure:

- ✓ COTEPOTE-APP-MOBILE
 - > .expo
 - > .expo-shared
 - > assets
 - > node_modules
 - > src
 - ↳ .gitignore
 - ↳ App.js
 - ↳ app.json
 - ↳ babel.config.js
 - ↳ package-lock.json
 - ↳ package.json
 - ❶ README.md
 - ↳ yarn.lock

On the right, the main editor window is titled 'App.js'. It contains the following code:

```
App.js
104     dispatch({ type: "SIGN_IN", token: token });
105 },
106
107     signOut: async () => {
108         await userInfoService.logout();
109         dispatch({ type: "SIGN_OUT" });
110     },
111 },
112 []
113 );
114
115 return (
116     <AuthContext.Provider value={authContext}>
117         {state.userToken !== null ? (
118             <>
119                 <NavigationContainer>
120                     <Navigator screenOptions={{ headerShown: false }}>
121                         <Screen name="Tab" component={Tabs} />
122                     </Navigator>
123                 </NavigationContainer>
124             </>
125         ) : (
126             <>
127                 <NavigationContainer>
128                     <Navigator screenOptions={{ headerShown: false }}>
129                         <Screen name="Connexion" component={ConnexionScreen} />
130                         <Screen name="Inscription" component={InscriptionScreen} />
131                         <Screen name="Loading" component={LoadingScreen} />
132                     </Navigator>
133                 </NavigationContainer>
134             </>
135         )
136     </AuthContext.Provider>
137 );
138 }
139
140 const styles = StyleSheet.create({
141     container: {
142         flex: 1,
143     },
144 });
145
```

DOSSIER PROFESSIONNEL (DP)

je commence par importer toutes les librairies et les méthodes react dont je sais que je vais avoir besoins, cela se fait de la manière suivante :

```
1 import React from "react";
2 import { StyleSheet } from "react-native";
3 import ConnexionScreen from "./src/screens/ConnexionScreen";
4 import InscriptionScreen from "./src/screens/InscriptionScreen";
5 import LoadingScreen from "./src/screens>LoadingScreen";
6 import Tabs from "./src/navigation/Tabs";
7 import "react-native-gesture-handler";
8 import { NavigationContainer } from "@react-navigation/native";
9 import { createStackNavigator } from "@react-navigation/stack";
10 import * as SecureStore from "expo-secure-store";
11 import UserService from "./src/services/UserService";
12 import AuthContext from "./src/context/Context";
13 import axios from "axios";
14 import UserInfoService from "./src/services/UserInfoService";
15
16 const { Navigator, Screen } = createStackNavigator();
```

Le découpage des applications JavaScript en modules permet d'avoir un code bien structuré.

ensuite pour l'utilisation de ces librairies, je n'ai qu'à consulter la documentation et appliquer comme il est décrit, comme exemple je peux parler de la documentation d'Axios qui est un client HTTP :

The screenshot shows the Openbase.js library documentation for the react-native-axios package. The left sidebar has navigation links for Categories, Discussions, JS, and Sign Up / Log in. The main content area shows the react-native-axios library with its npm version listed as 1.0.0. Below the library name, there's a search bar with the placeholder 'Search for any JavaScript Library or category'. A 'Save' button is also present. The 'DOCUMENTATION' tab is selected, showing an 'Example' section with code for performing a GET request using axios. The code example is:

```
// Make a request for a user with a given ID
axios.get('/user?ID=12345')
  .then(function (response) {
    console.log(response);
  })
  .catch(function (error) {
    console.log(error);
  });

// Optionally the request above could also be done as
axios.get('/user', {
  params: {
    ID: 12345
  }
})
  .then(function (response) {
    console.log(response);
  })
  .catch(function (error) {
```

To the right of the code, there's a 'JUMP TO' sidebar with a list of links related to the Axios API, including Features, Browser Support, Installing, Example, axios API, Request method aliases, Concurrency, Creating an instance, Instance methods, Request Config, Response Schema, Config Defaults, Global axios defaults, Custom instance defaults, and Config order of precedence. The 'Example' link is currently highlighted.

DOSSIER PROFESSIONNEL (DP)

Concernant la mise en place de ma user interface, il ne s'agit pas de HTML à proprement dit mais de JSX, cela ressemble à du HTML. :

```
<KeyboardAwareScrollView showsVerticalScrollIndicator={false}>
  <View style={styles.container}>
    <Text style={styles.title}>Côté Pote</Text>
    <Text style={styles.txt}>L'application de pari entre potes</Text>
    <TextInput
      style={styles.ipt}
      onChangeText={setLogin}
      value={login}
      placeholder="Email"
      placeholderTextColor="#CFCFCF"
    />
    <TextInput
      style={styles.ipt2}
      onChangeText={setPassword}
      secureTextEntry={true}
      value={password}
      placeholder="Mot de passe"
      placeholderTextColor="#CFCFCF"
    />
    <Pressable
      onPress={() => {
        connexion();
      }}
      style={styles.button}
    >
```

JSX (JavaScript Extension), est une extension de React qui permet d'écrire du code JavaScript qui ressemble à du HTML. En d'autres termes, JSX est une syntaxe de type HTML utilisée par React.

Il faut savoir que les balises ne sont pas les mêmes, mais la documentation de react native est très bien faite à ce niveau et l'on peut même chercher sur google l'équivalent de tel ou tel balise html en React Native.

Pour le style, ce n'est toujours pas du CSS :

Avec React Native, je stylise mon application en utilisant JavaScript. Tous les composants de base acceptent un accessoire nommé style. Les noms et les valeurs de style correspondent généralement au fonctionnement de CSS sur le Web, sauf que les noms sont écrits en utilisant la camelCase, par exemple backgroundColor plutôt que background-color.

DOSSIER PROFESSIONNEL (DP)

I faut commencer par importer "StyleSheet" depuis la librairie react-native et ensuite déclaré une constante où l'on va stocker la méthode "StyleSheet.Create({})" et à l'intérieur déclarer toute les classes dont j'ai besoins :

```
const styles = StyleSheet.create({
  container: {
    marginLeft: 20,
    marginTop: 250,
  },
  title: {
    fontFamily: "Poppins_700Bold",
    fontSize: 45,
    fontWeight: "bold",
    color: "white",
  },
  txt: {
    fontFamily: "Poppins_200ExtraLight",
    fontSize: 15,
    fontWeight: "bold",
    color: "white",
  },
  ipt: {
    height: 50,
    width: 335,
    borderWidth: 0.25,
    borderColor: "white",
    borderRadius: 5,
    padding: 10,
    marginTop: 50,
    color: "white",
  },
})
```

et ensuite pour attribuer du style à un élément dans mon code je le fais de la manière suivante :

```
<Text style={styles.txt}>S'inscrire</Text>
```

Maintenant nous allons voir les différents hooks que l'on a avec React Native.

Mais les Hooks qu'est ce que c'est ?

Les Hooks permettent de bénéficier d'un état local et d'autres fonctionnalités de React sans avoir à écrire une classe.

Je vais présenter les deux que j'ai le plus utilisé :

En premier, l'un des hooks les plus utiles et qui revient le plus dans mon code est le useState(). Ce hook me permet d'avoir un state dans mes fonctions composants et me débloque donc beaucoup plus de possibilités !

```
const [password, setPassword] = useState("");
```

Je commence tout d'abord par appeler la fonction useState() qui me permet de créer une nouvelle variable de state. Cette fonction me retourne un tableau de 2 éléments que j'assigne à des variables. La première variable renvoyée (password) correspond à notre nouvelle variable de state créée ; la deuxième variable renvoyée (ici setPassword) correspond elle, à la fonction à utiliser pour mettre à jour password. Cette fonction prend en paramètre la nouvelle valeur que l'on souhaite assigner à notre variable. Pour simplifier, c'est simplement un setState, mais juste sur le scope de ma variable.

DOSSIER PROFESSIONNEL (DP)

Le deuxième hook important est `useEffect()`. Celui-là me permet d'injecter dans mes fonctions composants la notion de cycle de vie.

Je me sert principalement de ces fonctions pour récupérer des données (call Api), mettre à jour des variables ou encore faire des modifications sur le DOM, principalement je sais que ça va être pour les composants dit dynamique.

Maintenant un exemple, disons que je souhaite mettre à jour le titre de mon document à chaque fois qu'une variable est mise à jour :

```
1 const App = (props) => {
2     const [items, setItems] = useState<Array<CartItem>">([]);
3
4     function addItem(item: CartItem) {
5         setItems(items.concat(item));
6     }
7
8     function deleteItem(item: CartItem) {
9         setItems(items.filter((i: CartItem) => i.id !== item.id));
10    }
11
12    useEffect(() => {
13        document.title = `${items.length} article(s) dans votre panier`;
14    });
15
16    return (
17        <div className="App">
18            <ShoppingCart items={items} deleteItem={deleteItem} />
19            <Shop addItem={addItem} />
20        </div>
21    )
22 }
```

Avec cette méthode, cela se fait très facilement.

DOSSIER PROFESSIONNEL (DP)

Maintenant il faut savoir que la navigation entre différente vue est très différente de celle du web classique, il n'y a pas de "href" qui ramène d'un lien à l'autre.

Il y a React Navigation qui est une bibliothèque populaire pour le routage et la navigation dans une application React Native. Cette bibliothèque aide à résoudre le problème de la navigation entre plusieurs écrans et du partage des données entre eux.

Pour naviguer entre les écrans, j'utilise StackNavigator, il fonctionne exactement comme une pile d'appels.

Chaque écran vers lequel je navigue est poussé vers le haut de la pile. Chaque fois que j'appuie sur le bouton Retour, les écrans se détachent du haut de la pile.

tout d'abord je commence par installer la librairie avec cette commande sur mon terminal :

```
npm install @react-navigation/native
```

Ensuite, j'installe @react-navigation/stack et ses dépendances

par la suite je m'occupe de les importer :

```
import { NavigationContainer } from "@react-navigation/native";
import { createStackNavigator } from "@react-navigation/stack";

const { Navigator, Screen } = createStackNavigator();
```

par la suite je créer le corps de ma stack de navigation de la façon suivante :

```
<Navigator screenOptions={{ headerShown: false }}>
  <Screen name="CreateBet" component={CreateBetScreen} />
  <Screen name="SearchBetById" component={SearchBetScreen} />
  <Screen name="NewBet" component={NewBetScreen} />
</Navigator>
```

DOSSIER PROFESSIONNEL (DP)

je m'occupe simplement de ranger à l'intérieur de ma stack de navigation toutes mes vues dans la balise Screen, je dois lui donner un nom qui me servira pour ensuite les appeler et ensuite quelle composant je mets pour tel ou tel screen.

Après ça, ma stack de navigation est prête à l'utilisation.

pour naviguer entre 2 vues, je dois d'abord passer dans la props de mon composant "{navigation}" :

```
export default function connexion({ navigation }) {
  const { signIn } = useContext(AuthContext);

  let [fontsLoaded] = useFonts({ Poppins_200ExtraLight });
  const [login, setLogin] = useState("");
  const [password, setPassword] = useState("");

  const goTo = () => {
    navigation.navigate("Inscription");
  };
}
```

et ensuite je stock dans une constante la méthode "navigation.navigate("nom_du_screen")" pour signaler que lorsque j'appellerai cette constante, je veux que la vue qui a pour nom "Inscription" soit afficher.

```
<Pressable
  onPress={() => {
    goTo();
  }}
  style={styles.button1}
>
```

je l'utilise dans une balise Pressable qui est un button et donc lorsque que j'appuierai dessus je serai sur la vue inscription.

Dossier Professionnel (DP)

pour faire une sorte de bouton de retour, il y a une méthode prévue dans navigation, la librairie couvre beaucoup de type de cas :

```
const goTo = () => {
  navigation.goBack();
};
```

2. Précisez les moyens utilisés :

j'ai utilisé pour la création de mon application mobile NodeJS ainsi que NPM.

Comme IDE j'utilise Visual Studio Code.

Comme logiciel de versionning j'utilise Git Kraken couplé à GitHub pour mettre à jour mon repository.

Comme application de gestion de projet j'utilise Trello.

Comme FrameWork, j'ai utilisé React Native Expo.

j'utilise mon SmartPhone avec l'application Expo Go.

3. Avec qui avez-vous travaillé ?

j'ai travaillé seul sur ce projet

4.1 Contexte

La Plateforme

Chantier, atelier, service

Dans le cadre de ma formation de concepteur /développeur d'application

Période d'exercice

Du : 25/01/2024 au : 27/05/2024

5. Informations complémentaires(facultatif)

DOSSIER PROFESSIONNEL (DP)

Titres, diplômes, CQP, attestations de formation

(facultatif)

Intitulé	Autorité ou organisme	Date
Bac Série S spécialité Physique chimie	Ort, Marseille.	06/2017.

DOSSIER PROFESSIONNEL

(DP)

Déclaration sur l'honneur

Je soussigné(e) Benjamin Arfi

déclare sur l'honneur que les renseignements fournis dans ce dossier sont exacts et que je suis
l'auteur(e) des réalisations jointes.

Fait à Marseille

le 26/06/2025

pour faire valoir ce que de droit.

Signature : Benjamin Arfi

Documents illustrant la pratique professionnelle

Alternance

DOSSIER PROFESSIONNEL (DP)

ANNEXES

(Si le RC le prévoit)