Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

MINISTÈRE CHARGÉ
DE L'EMPLOI

DOSSIER PROFESSIONNEL (DP)

Nom de naissance - MACHTELINCKX
Nom d'usage - MACHTELINCKX
Prénom - Clément
Adresse - 100 route du golf
06210
mandelieu la napoule

Titre professionnel visé

RNCP 37873 Concepteur développeur d'applications

MODALITÉ D'ACCÈS :

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

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

EXEMPLES DE PRATIQUE PROFESSIONNELLE

DOSSIER PROFESSIONNEL (DP)

Activité-type 1 Développer une application sécurisée

Exemple n°1 ▸ Développement sécurisé de l'API Symfony pour LDVEH

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

Dans le cadre de mon projet *Livre Dont Vous Êtes le Héros*, j'ai développé une application mobile et une API sécurisée. Ce projet a été réalisé en autonomie, sur une durée étendue, dans un contexte de formation au sein de l'école La Plateforme.
J'ai commencé par installer et configurer mon environnement de développement à l'aide de **VS Code**, Composer, Symfony CLI, Expo CLI et Docker. J'ai ensuite conçu la base de données selon la méthode Merise, puis développé les entités métier (livre, page, choix, aventurier, aventure…) avec Doctrine.
J'ai mis en place une **API REST sécurisée** avec Symfony et le bundle **LexikJWTAuthenticationBundle**, gérant l'authentification par token JWT, et l'intégration de groupes de sérialisation pour contrôler les données exposées.

Côté client, j'ai construit l'application mobile en **React Native avec TypeScript**, en intégrant une logique métier côté frontend (navigation, choix, affichage des pages, redirection après combat, etc.). L'ensemble des appels API est sécurisé et géré via un fichier central *api.ts*, avec injection automatique du token. J'ai également utilisé Zustand pour la gestion d'état.

J'ai porté une attention particulière à la **sécurité de l'accès API**, à la séparation claire entre rôles utilisateur (admin / joueur), et à la protection des routes critiques. J'ai aussi mis en place un **système de tests unitaires** pour valider la cohérence du cœur métier : progression dans l'histoire, blocage sur pages avec monstres, validation des combats.

2. Précisez les moyens utilisés :

Environnement technique : Visual Studio Code, Symfony (PHP), React Native (Expo, TypeScript), MySQL, Doctrine, Docker (optionnel), GitHub.

Sécurité : Authentification JWT (LexikJWT), séparation frontend/backend, token stocké via AsyncStorage

et transmis via **Authorization: Bearer**, absence de refresh token (choix assumé pour simplifier le cycle de vie du token), communication HTTPS.

Tests : PHPUnit pour les tests back, Postman pour les scénarios manuels d'authentification et de navigation.

Organisation du projet : GitHub Projects en mode kanban pour garder une trace claire des tâches effectuées.

3. Avec qui avez-vous travaillé ?

J'ai mené ce projet seul. Cependant, j'ai conservé une organisation de travail professionnelle, en utilisant des outils collaboratifs (GitHub + GitHub Projects), afin de simuler une dynamique d'équipe.
Par ailleurs, j'ai pu m'appuyer sur l'accompagnement pédagogique de l'école, notamment lors de points réguliers avec mes formateurs.

4. Contexte

Nom de l'entreprise, organisme ou association ▸ La Plateforme

Chantier, atelier, service ▸ en formation

Période d'exercice ▸ Du : 01/01/2025 au : 01/08/2025

5. Informations complémentaires *(facultatif)*

Ce projet m'a permis de consolider mes compétences en sécurité, gestion de projet, développement d'API, ainsi qu'en développement mobile. J'ai appliqué une démarche proactive de résolution de problèmes, notamment lors des tests liés à la navigation et aux combats.
Enfin, j'ai veillé à respecter les bonnes pratiques recommandées par l'ANSSI, notamment en matière de gestion des identifiants, de cloisonnement des accès et de séparation des responsabilités.

DOSSIER PROFESSIONNEL (DP)

Activité-type 2

Concevoir et développer une application sécurisée organisée en couches

Exemple n° 1 *Conception et développement de l'architecture multicouche de l'application LDVEH*

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

Dans le cadre du projet LDVEH, j'ai commencé par **analyser les besoins fonctionnels** issus de mon idée de départ : permettre à des utilisateurs de vivre une aventure interactive à travers un livre dont ils sont le héros. Cette analyse m'a permis d'identifier les **entités principales** (livre, page, choix, utilisateur, aventurier, monstre, etc.) et les **flux d'interactions** dans l'application.

J'ai ensuite réalisé **des maquettes de l'interface mobile** à l'aide de Figma, pour proposer une navigation fluide et claire, en m'inspirant des principes UX/UI. Ces maquettes m'ont servi de base pour concevoir **l'architecture logicielle en couches**, séparant la logique métier (backend Symfony), la couche d'accès aux données (Doctrine + MySQL), et l'interface utilisateur (frontend React Native).

Pour modéliser la base de données, j'ai conçu un **MCD, un MLD, puis un MPD** en suivant la méthode Merise, en utilisant dbdiagram.io et dbdesigner.net. Ces schémas m'ont permis de créer une base de données relationnelle cohérente, avec des **relations complexes bien définies**, des **clés étrangères**, et **des contraintes d'unicité métier** (par exemple, une page ne peut exister qu'une seule fois par numéro dans un livre donné).

Enfin, j'ai développé les **composants d'accès aux données** à travers les repositories Doctrine et des services Symfony. J'ai sécurisé tous les accès, validé les entrées, traité les cas d'exception, et mis en place des jeux d'essais et **tests unitaires sur les composants critiques** comme le **AdventureService** et le **CombatService**.

2. Précisez les moyens utilisés :

Symfony pour l'architecture logicielle en couches côté serveur.

Doctrine ORM pour la persistance des données et la création des entités.

React Native + Expo pour l'interface utilisateur mobile.

MySQL comme système de gestion de base de données relationnelle.

VS Code comme environnement principal de développement.

dbdiagram.io et dbdesigner.net pour la modélisation Merise.

PHPUnit pour les tests unitaires.

Figma pour la création des maquettes graphiques.

Postman pour les tests d'API.

3. Avec qui avez-vous travaillé ?

J'ai mené ce projet en autonomie, mais je me suis inspiré de méthodologies professionnelles d'équipe. J'ai notamment utilisé **GitHub Projects** pour organiser les tâches, simuler un backlog, suivre la progression via des tickets, et garder une vision claire de l'avancement, comme on le ferait dans une gestion agile. Cela m'a permis de structurer le projet comme si j'étais intégré à une équipe réelle.

4. Contexte

Nom de l'entreprise, organisme ou association ▸ La Plateforme

Chantier, atelier, service ▸ en formation

Période d'exercice ▸ Du : 01/01/2025 au : 01/08/2025

5. Informations complémentaires *(facultatif)*

Tout au long du projet, j'ai documenté mes choix techniques dans le dossier de conception et dans le code source. J'ai veillé à respecter une **séparation claire des responsabilités**, à sécuriser les couches critiques, et à structurer la logique métier dans des **services dédiés injectés proprement dans les contrôleurs**. Cela m'a permis d'assurer la **cohérence**, la **testabilité**, et la **maintenabilité** du projet.

DOSSIER PROFESSIONNEL (DP)

Activité-type 2

Concevoir et développer une application sécurisée organisée en couches

Exemple n° 2 – *Développement d'un site vitrine en architecture MVC avec Symfony (Global Yachting)*

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

Dans le cadre d'un exercice de renforcement sur Symfony, j'ai développé un site vitrine pour une entreprise fictive de location de bateaux : **Global Yachting**. Ce projet avait pour but de mettre en pratique l'architecture MVC de Symfony, de structurer correctement les différentes couches de l'application (Modèle, Vue, Contrôleur), et d'appliquer les bonnes pratiques de développement web back-end.
J'ai modélisé les entités principales (**Boat**, **Service**, **User**), configuré la base de données avec Doctrine, généré les formulaires, mis en place les routes, et conçu les vues associées en **Twig**. J'ai également mis en œuvre l'enregistrement d'utilisateurs et l'authentification basique via **Symfony Security**, ainsi qu'un système d'administration pour gérer les services.

Ce projet a été réalisé de manière autonome dans un contexte d'entraînement personnel, en parallèle de ma formation principale

2. Précisez les moyens utilisés :

Symfony 6 (framework PHP MVC)

Doctrine ORM pour la gestion des entités et des accès base de données

Twig pour le rendu des vues côté utilisateur

Symfony MakerBundle pour accélérer le développement de code standard

FormBuilder Symfony pour la gestion des formulaires utilisateur

Symfony Security pour l'authentification simple

MySQL comme base de données relationnelle

Visual Studio Code comme éditeur principal

php bin/console make: pour la génération automatisée des classes et des vues

3. Avec qui avez-vous travaillé ?

Ce projet a été réalisé **entièrement en autonomie**, dans le cadre d'un travail personnel d'entraînement. J'ai tout de même appliqué une structure professionnelle dans mon organisation du code et de l'architecture logicielle, afin de rester fidèle aux standards d'un développement en entreprise.

4. Contexte

Nom de l'entreprise, organisme ou association ▸ La Plateforme

Chantier, atelier, service ▸ en formation

Période d'exercice ▸ Du : 01/03/2024 au : 01/06/2024

5. Informations complémentaires *(facultatif)*

Ce projet m'a permis de **consolider ma maîtrise de l'architecture en couches**, d'appliquer la logique MVC dans un contexte concret, et d'améliorer mes compétences en manipulation de formulaires, routing Symfony, et sécurité de base.
Bien qu'il ne comporte pas de front-end avancé, il m'a offert un cadre parfait pour approfondir les fondations de Symfony, et pour mieux structurer les interactions entre base de données, logique métier, et rendu utilisateur.

DOSSIER PROFESSIONNEL (DP)

Activité-type 3 Préparer le déploiement d'une application sécurisée

Exemple n° 1 – *Préparation du déploiement et vérification métier de l'API sécurisée LDVEH*

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

Dans le cadre du projet **LDVEH**, j'ai préparé et réalisé le **déploiement de l'API Symfony sur la plateforme Scalingo**, afin de proposer une version stable et accessible en ligne pour les tests finaux et la démonstration.

J'ai configuré mon environnement de production en respectant les contraintes imposées par l'hébergement cloud : gestion des variables d'environnement, configuration du fichier Procfile, adaptation du **composer.json**, et connexion à la base de données distante fournie par Scalingo.

En parallèle, j'avais mis en place un environnement local de développement avec Docker, afin d'assurer une continuité entre les phases de développement et de production. J'ai également documenté toutes les étapes dans un guide de déploiement versionné sur GitHub.

Enfin, j'ai conçu et réalisé des **tests fonctionnels métiers** simulant des scénarios utilisateur complets, incluant la navigation conditionnelle et les combats bloquants. Ces tests ont été menés à la fois via **PHPUnit (tests automatisés)** et **Postman (tests manuels API)**.

2. Précisez les moyens utilisés :

Scalingo : hébergement cloud PaaS utilisé pour le déploiement de l'API Symfony

Dashboard Scalingo : gestion des environnements, des déploiements, logs et variables d'environnement

Docker / docker-compose : utilisé localement pour le développement et les tests

PHPUnit : pour l'exécution des tests automatisés sur la logique métier (combat, navigation)

Postman : pour les scénarios de test API en manuel

GitHub : gestion de version, suivi des commits, préparation au déploiement (Git push → Scalingo)

README.md + guide de déploiement : documentant le processus complet

Fichier .env.production + Procfile : pour configurer l'environnement Scalingo

Terminal Linux : utilisé pour les dernières vérifications et tests CLI post-déploiement

3. Avec qui avez-vous travaillé ?

Bien que le projet ait été développé en autonomie, j'ai sollicité l'aide ponctuelle de mon formateur cybersécurité/réseaux pour valider la **configuration des variables d'environnement, la sécurité des accès HTTP**, ainsi que la **connexion à la base de données distante**.

Ces échanges m'ont permis de sécuriser le déploiement sur Scalingo tout en assurant un bon niveau d'isolation et de contrôle des services exposés.

4. Contexte

Nom de l'entreprise, organisme ou association ▸ La Plateforme

Chantier, atelier, service ▸ en formation

Période d'exercice ▸ Du : 01/01/2025 au : 01/08/2025

5. Informations complémentaires *(facultatif)*

Ce travail m'a permis de mettre en pratique un **déploiement réel sur une plateforme cloud**, avec des

contraintes proches de celles rencontrées en entreprise.

J'ai pu mieux comprendre les **enjeux de la configuration d'un environnement de production**, l'importance des **variables d'environnement sécurisées**, ainsi que la gestion des logs et du cycle de vie applicatif.

Je n'ai pas encore mis en place de pipeline DevOps complet, mais le déploiement via Git (push vers Scalingo) permet déjà une livraison rapide et maîtrisée. Une **intégration continue via GitHub Actions** est envisagée comme évolution post-projet.

Activité-type 3 — Préparer le déploiement d'une application sécurisée

Exemple n° 2 ▸ *Mise en place d'un pipeline CI/CD avec GitHub Actions sur le projet BDD_Creator*

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

Dans le cadre d'un projet secondaire nommé **BDD_Creator**, j'ai expérimenté la mise en place d'une **chaîne d'intégration et de déploiement continue (CI/CD)**. L'objectif était de fluidifier le cycle de développement, depuis le push du code jusqu'au déploiement sur un serveur de test.

J'ai configuré un workflow **GitHub Actions** pour automatiser le processus de build, exécuter des tests PHP, et préparer le déploiement. J'ai également rédigé les fichiers de configuration nécessaires (.yml) pour déclencher automatiquement le pipeline à chaque push ou pull request sur la branche principale.

Même si ce projet ne respectait pas encore tous les standards de sécurité (notamment en matière d'authentification), il m'a permis de **comprendre concrètement les enjeux d'un déploiement sécurisé et automatisé**, et de les appliquer ensuite à mon projet principal LDVEH.

2. Précisez les moyens utilisés :

GitHub Actions pour l'automatisation du build, des tests et du déploiement

Docker pour simuler un environnement serveur proche de la production

PHPUnit pour l'exécution des tests unitaires pendant le pipeline

GitHub pour le versionnement, la gestion des branches et l'hébergement du pipeline CI

Fichier .yml pour la configuration du workflow CI/CD

3. Avec qui avez-vous travaillé ?

Ce projet a été réalisé **en autonomie**, mais j'ai profité des retours et conseils de mes formateurs, notamment sur les bonnes pratiques d'intégration continue. Cette démarche m'a également permis de consolider ma gestion de projet sur GitHub (branches, pull requests, revue de code simulée).

4. Contexte

Nom de l'entreprise, organisme ou association ▸ La Plateforme

DOSSIER PROFESSIONNEL (DP)

Chantier, atelier, service ▸ en formation

Période d'exercice ▸ Du : 15/02/2025 au : 10/03/2025

5. Informations complémentaires *(facultatif)*

Même si ce projet ne comportait pas encore de couche de sécurité avancée, il m'a permis de **maîtriser les bases de l'automatisation CI/CD**. Ce savoir m'a ensuite été **directement utile pour le projet LDVEH**, dans lequel j'ai préparé une mise en ligne manuelle mais documentée, avec un Docker maîtrisé.
Il s'agit donc d'un **projet tremplin**, qui m'a permis de progresser sur le plan professionnel en découvrant l'importance de la reproductibilité et de la rigueur dans le processus de déploiement.

Titres, diplômes, CQP, attestations de formation

(facultatif)

Intitulé	Autorité ou organisme	Date
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.

Déclaration sur l'honneur

Je soussigné(e) [prénom et nom] **Clément Machtelinckx** ,

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 Cannes le 23/05/2025

pour faire valoir ce que de droit.

Signature :

Documents illustrant la pratique professionnelle

(facultatif)

Intitulé
Cliquez ici pour taper du texte.

DOSSIER PROFESSIONNEL (DP)

ANNEXES

Activité type n° 1: exemple 1

[security.yaml]

```yaml
firewalls:
    dev:
        pattern: ^/(_(profiler|wdt)|css|images|js)/
        security: false

    api:
        pattern: ^/(api|fight|page)
        stateless: true
        provider: app_user_provider
        json_login:
            check_path: /api/login
            success_handler: lexik_jwt_authentication.handler.authentication_success
            failure_handler: lexik_jwt_authentication.handler.authentication_failure
            username_path: email
            password_path: password
        jwt: ~

    main:
        lazy: true
        provider: app_user_provider
        stateless: false
        form_login:
            login_path: app_login
            check_path: app_login
            success_handler: lexik_jwt_authentication.handler.authentication_success
            failure_handler: lexik_jwt_authentication.handler.authentication_failure
            default_target_path: /admin
        logout:
            path: app_logout
            target: /login
        security: true

access_control:
    - { path: ^/api/login, roles: PUBLIC_ACCESS }
    - { path: ^/api/register, roles: PUBLIC_ACCESS }
    - { path: ^/api, roles: IS_AUTHENTICATED_FULLY }
    - { path: ^/api/my-adventurers, roles: IS_AUTHENTICATED_FULLY }
    - { path: ^/fight, roles: IS_AUTHENTICATED_FULLY }
    - { path: ^/page, roles: IS_AUTHENTICATED_FULLY }
    - { path: ^/login, roles: PUBLIC_ACCESS }
    - { path: ^/, roles: ROLE_USER }
```

[auth.ts]

```ts
// services/auth.ts
import AsyncStorage from '@react-native-async-storage/async-storage';

const TOKEN_KEY = 'access_token';

export async function saveToken(token: string) {
  await AsyncStorage.setItem(TOKEN_KEY, token);
}

export async function getToken(): Promise<string | null> {
  return await AsyncStorage.getItem(TOKEN_KEY);
}

export async function clearToken() {
  await AsyncStorage.removeItem(TOKEN_KEY);
}
```

[AdventureTest.php]

```php
0 references | 0 implementations
class AdventureTest extends TestCase
{
    0 references | 0 overrides
    public function testConstructorSetsStartedAt(): void
    {
        $adventure = new Adventure();
        $this->assertInstanceOf(expected: \DateTimeImmutable::class, actual: $adventure->getStartedAt());
    }

    0 references | 0 overrides
    public function testSetAndGetUser(): void
    {
        $user = new User();
        $adventure = new Adventure();
        $adventure->setUser(user: $user);
        $this->assertSame(expected: $user, actual: $adventure->getUser());
    }

    0 references | 0 overrides
    public function testSetAndGetBook(): void
    {
        $book = new Book();
        $adventure = new Adventure();
        $adventure->setBook(book: $book);
        $this->assertSame(expected: $book, actual: $adventure->getBook());
    }

    0 references | 0 overrides
    public function testSetAndGetAdventurer(): void
    {
        $adventurer = new Adventurer();
        $adventure = new Adventure();
        $adventure->setAdventurer(adventurer: $adventurer);
        $this->assertSame(expected: $adventurer, actual: $adventure->getAdventurer());
    }

    0 references | 0 overrides
    public function testSetAndGetCurrentPage(): void
    {
        $page = new Page();
        $adventure = new Adventure();
        $adventure->setCurrentPage(currentPage: $page);
        $this->assertSame(expected: $page, actual: $adventure->getCurrentPage());
    }
```

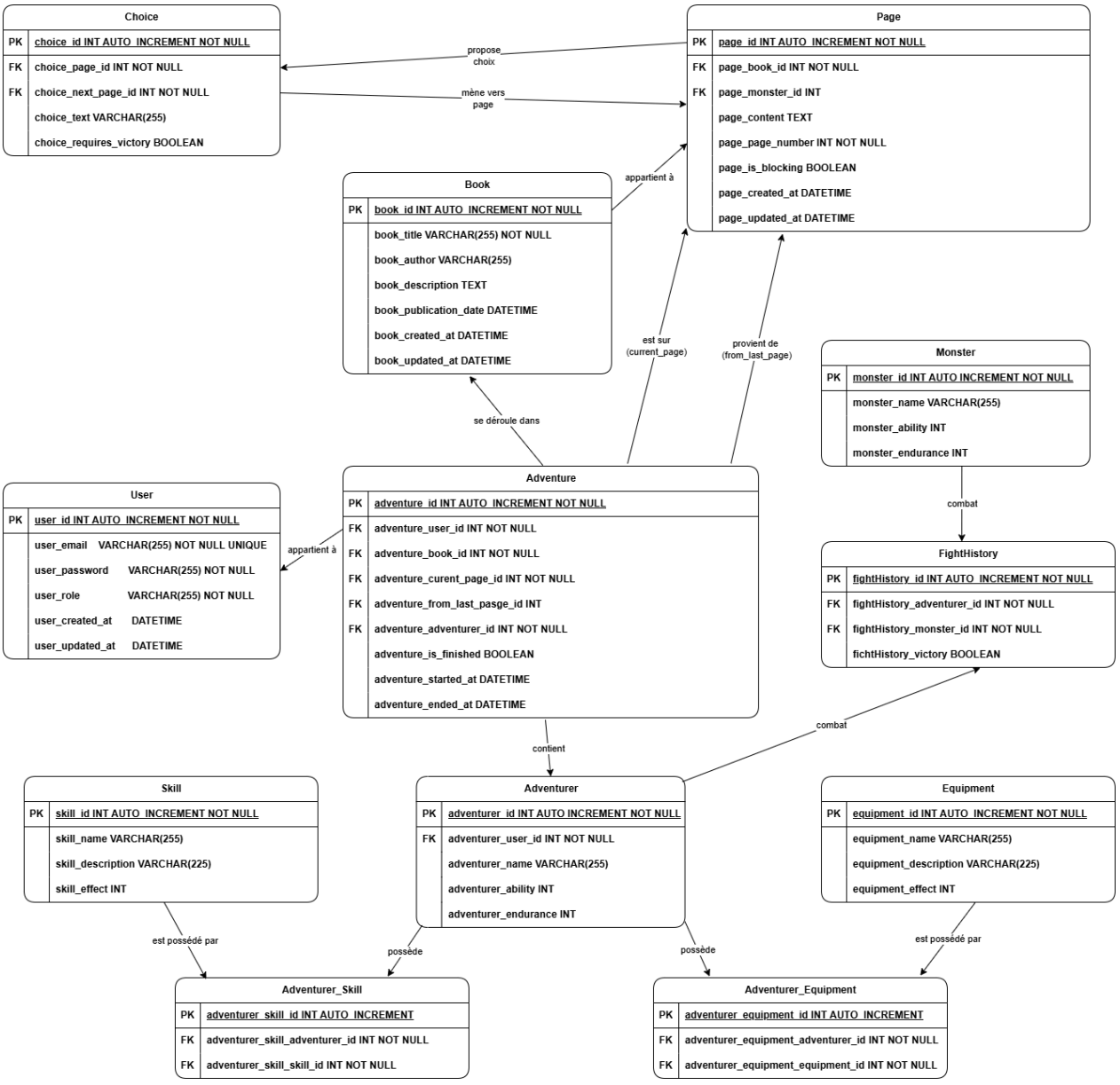
Activité type n° 2 : exemple 1

[AdventureService.php]

```php
12 references | 0 implementations
class AdventureService
{
    4 references | 0 overrides
    public function __construct(
        private AdventureRepository $adventureRepository,
        private EntityManagerInterface $em
    ) {}

    1 reference | 0 overrides
    public function startAdventure(User $user, Book $book, Adventurer $adventurer): Adventure
    {

        $existing = $this->adventureRepository->findOneBy(criteria: ['user' => $user, 'book' => $book, 'isFinished' => false]);

        if ($existing) {
            $this->em->remove(object: $existing);
            $this->em->flush();
        }

        $startPage = $book->getPage()->first();
        if (!$startPage instanceof Page) {
            throw new \LogicException(message: "Le livre n'a pas de page de départ.");
        }
        // 🛠 On crée l'aventure
        $adventure = new Adventure();
        $adventure->setUser(user: $user);
        $adventure->setBook(book: $book);
        $adventure->setAdventurer(adventurer: $adventurer);
        $adventure->setCurrentPage(currentPage: $startPage);
        $adventure->setFromLastPage(fromLastPage: $startPage); // Au début, c'est la même
        $adventure->setIsFinished(isFinished: false);

        $this->em->persist(object: $adventure);
        $this->em->flush();

        return $adventure;
    }
```

DOSSIER PROFESSIONNEL (DP)

[schéma MPD]



Activité type n° 2 : exemple 2

[BoatController.php]

```php
#[Route('/boat/show/{id}', name: 'boat.show', methods: ['GET'])]
public function show(Boat $boat, BoatImage $boatImages, BoatImageRepository $boatImageRepository): Response
{
    $boatImages = $boatImageRepository->findBy(['boat' => $boat]);
    return $this->render('pages/boat/show.html.twig', [
        'boat' => $boat,
        'boatImages' => $boatImages
    ]);
}
```

[BoatRepository.php]

```php
class BoatRepository extends ServiceEntityRepository
{
    public function __construct(ManagerRegistry $registry)
    {
        parent::__construct($registry, Boat::class);
    }


    // BoatRepository.php// BoatRepository.php
// BoatRepository.php
public function findByBrand(?string $brand): array
{
    $query = $this->createQueryBuilder('b')
        ->andWhere('b.brand = :brand')
        ->setParameter('brand', $brand)
        ->orderBy('b.id', 'ASC')
        ->getQuery();

    return $query->getResult();
}
```

[boat/show.html.twig]

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

{% block title %}Global - Yatching - Boat_page{% endblock %}

{% block body %}
<div class="container mt-4">
    <div class="row">
        <div class="col-md-6">
            <h1>{{ boat.name }}</h1>

            <p>{{ boat.description }}</p>

            <h3 class="mt-4">Section</h3>
            <p>Length: {{ boat.loa }}</p>
            <p>Beam: {{ boat.beam }}</p>
            <p>Draft: {{ boat.draft }}</p>
            <p>Year: {{ boat.year ? boat.year.format('d/m/Y') : '' }}</p>


            <h3 class="mt-4">Shipyard</h3>
            <p>Builder: {{ boat.builder }}</p>
            <p>Material: {{ boat.material }}</p>
            <p>Accommodation: {{ boat.accomodation }}</p>

            <h3 class="mt-4">Propulsion</h3>
            <p>Engine: {{ boat.engines }}</p>
            <p>Range: {{ boat.boatRange }}</p>
            <p>Max speed: {{ boat.maxSpeed }}</p>
            <p>Cruising speed: {{ boat.cruiseSpeed }}</p>

            <div class="mt-4">
                <a href="{{ path('boat.show', {'id': boat.id}) }}" class="btn btn-success">Show</a>
                <a href="{{ path('boat.edit', {'id': boat.id}) }}" class="btn btn-primary">Edit</a>
                <a href="{{ path('boat.delete', {'id': boat.id}) }}" class="btn btn-danger">Delete</a>
            </div>
        </div>
```

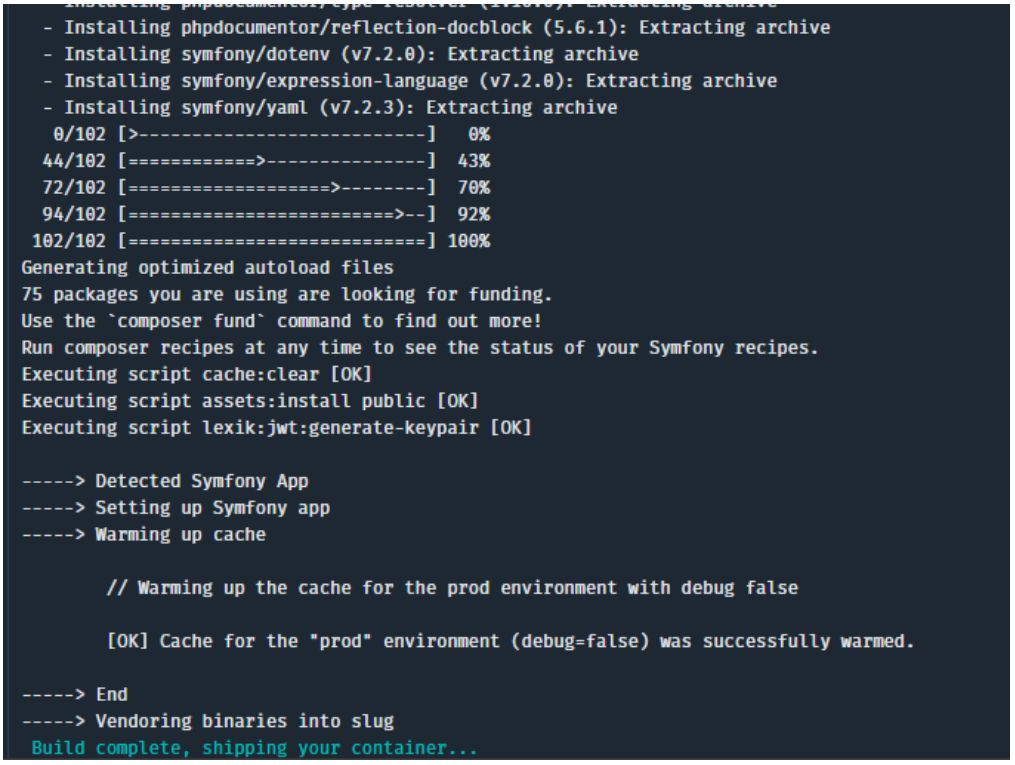
Activité type n° 3 exemple 1:

[variable d'environnement production]

```
APP_DEBUG=0
APP_ENV=prod
DATABASE_URL=$SCALINGO_MYSQL_URL
JWT_PASSPHRASE=5d[redacted]
JWT_PUBLIC_KEY=%kernel.project_dir%/config/jwt/public.pem
JWT_SECRET_KEY=%kernel.project_dir%/config/jwt/private.pem
SCALINGO_MYSQL_URL=mysql://ldveh1_4234:[redacted]
```

[terminal scalingo]

```
 <-- Start deployment of ldveh1 -->
       Fetching source code
       Fetching deployment cache
-----> Bundling Nginx 1.28.0
              Checksums match. Fetching from cache.
-----> Bundling PHP 8.4.10
              Checksums match. Fetching from cache.
-----> Bundling platform default extensions
       apcu
              Checksums match. Fetching from cache.
       phpredis
              Checksums match. Fetching from cache.
       mongodb
              Checksums match. Fetching from cache.
-----> Vendoring Composer 2.8.10
       Checksums match. Fetching Composer from cache.
-----> Bundling additional extensions
       PHP extension ctype is embedded in runtime
       PHP extension iconv is embedded in runtime
              Checksums match. Fetching from cache.
       Installing PHP extension: sodium
       Checksums match. Fetching from cache.
-----> Installing application dependencies with Composer
Installing dependencies from lock file
Verifying lock file contents can be installed on current platform.
Warning: The lock file is not up to date with the latest changes in composer.json. You may be getti
`composer update` or `composer update <package name>`.
Package operations: 104 installs, 0 updates, 0 removals
    0 [>---------------------------]    0 [->--------------------------]
  - Installing symfony/flex (v2.5.0): Extracting archive
  - Installing symfony/runtime (v7.2.3): Extracting archive
```


- Installing phpdocumentor/reflection-docblock (5.6.1): Extracting archive
- Installing symfony/dotenv (v7.2.0): Extracting archive
- Installing symfony/expression-language (v7.2.0): Extracting archive
- Installing symfony/yaml (v7.2.3): Extracting archive
0/102 [>----------] 0%
44/102 [============>----------] 43%
72/102 [====================>--------] 70%
94/102 [==========================>--] 92%
102/102 [============================] 100%
Generating optimized autoload files
75 packages you are using are looking for funding.
Use the `composer fund` command to find out more!
Run composer recipes at any time to see the status of your Symfony recipes.
Executing script cache:clear [OK]
Executing script assets:install public [OK]
Executing script lexik:jwt:generate-keypair [OK]
-----> Detected Symfony App
-----> Setting up Symfony app
-----> Warming up cache
// Warming up the cache for the prod environment with debug false
[OK] Cache for the "prod" environment (debug=false) was successfully warmed.
-----> End
-----> Vendoring binaries into slug
Build complete, shipping your container...

[terminal distant + commande php bin/console do:mi:st -n]

```
-----> Hello clement-machtelinckx, nice to see you!

C:\Users\chong>scalingo -a ldveh1 run bash
-----> Starting container one-off-898  Done in 0.161 seconds
-----> Connecting to container [one-off-898]...
-----> Process 'bash' is starting...

You are in a one-off container. This is a copy of your production
environment. It is destroyed at the end of its execution. Thus, do not expect
any created file to show up in your production environment. Production is
immutable.

If you need to interact with a database, you need to download the CLI of this
database. You can download and install your database CLI with
dbclient-fetcher <DB type> [<version>]

If you need to use the Scalingo CLI, you can install it with
install-scalingo-cli

[09:00][osc-fr1] Scalingo ~ $
```

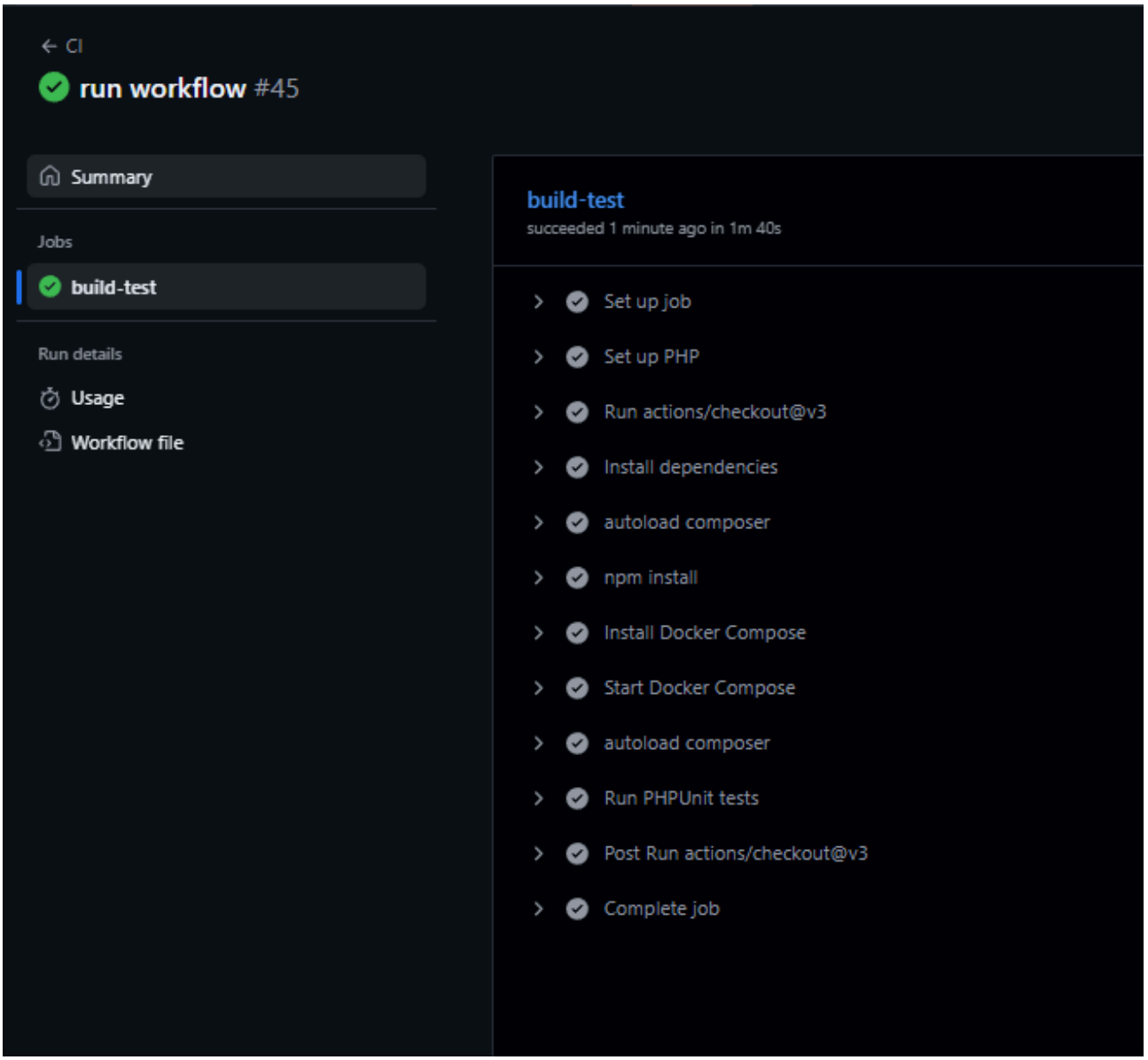
```
[09:00][osc-fr1] Scalingo ~ $ php bin/console do:mi:st
php bin/console do:mi:st
+----------------------+----------------------+-----------------------------------------------------------------------------+
| Configuration                                                                                                              |
+----------------------+----------------------+-----------------------------------------------------------------------------+
| Storage              | Type                 | Doctrine\Migrations\Metadata\Storage\TableMetadataStorageConfiguration |
|                      | Table Name           | doctrine_migration_versions                                                 |
|                      | Column Name          | version                                                                     |
|----------------------+----------------------+-----------------------------------------------------------------------------|
| Database             | Driver               | Symfony\Bridge\Doctrine\Middleware\IdleConnection\Driver                    |
|                      | Name                 | ldveh1_4234                                                                 |
|----------------------+----------------------+-----------------------------------------------------------------------------|
| Versions             | Previous             | DoctrineMigrations\Version20250716130841                                    |
|                      | Current              | DoctrineMigrations\Version20250721083643                                    |
|                      | Next                 | Already at latest version                                                   |
|                      | Latest               | DoctrineMigrations\Version20250721083643                                    |
|----------------------+----------------------+-----------------------------------------------------------------------------|
| Migrations           | Executed             | 10                                                                          |
|                      | Executed Unavailable | 0                                                                           |
|                      | Available            | 10                                                                          |
|                      | New                  | 0                                                                           |
|----------------------+----------------------+-----------------------------------------------------------------------------|
| Migration Namespaces | DoctrineMigrations   | /app/migrations                                                             |
+----------------------+----------------------+-----------------------------------------------------------------------------+
[09:00][osc-fr1] Scalingo ~ $
[09:00][osc-fr1] Scalingo ~ $
```

Activité type n°3 exemple 2

1. .github/workflows/ci.yml

```yaml
build-test:
  runs-on: ubuntu-latest

  steps:

    # 1. Set up PHP 8.2
    - name: Set up PHP
      uses: shivammathur/setup-php@v2
      with:
        php-version: '8.2'

    # 2. Check out repository
    - uses: actions/checkout@v3

    # 3. Install Composer dependencies in BDD_Creator directory
    - name: Install dependencies
      working-directory: ./BDD_Creator
      run: composer install

    # 4. Autoload Composer
    - name: autoload composer
      working-directory: ./BDD_Creator
      run: composer dump-autoload

    # 5. Run npm install in the frontend folder
    - name: npm install
      working-directory: ./front_bdd_creator
      run: npm install

    # 6. Install Docker Compose
    - name: Install Docker Compose
      run: |
        sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
        sudo chmod +x /usr/local/bin/docker-compose

    # 7. Start Docker Compose
    - name: Start Docker Compose
      working-directory: ./
      run: docker-compose up -d

    # AUtoload Composer
    - name: autoload composer
      working-directory: ./BDD_Creator
      run: docker exec -i bdd_depresion_php-api_1 composer dump-autoload

    # 8. Run PHPUnit tests
    - name: Run PHPUnit tests
      run: |
        sleep 10  # Attendre 10 secondes pour que MySQL soit prêt
        docker exec -i bdd_depresion_php-api_1 ./vendor/bin/phpunit tests/functional/DatabaseTest.php
```

2. Pipeline GitHub Actions en cours d'exécution



3. Test unitaire lancé automatiquement