

AUTOMATYZACJA PRACY PROGRAMISTY WWW – COMPOSER

SPIS TREŚCI

Spis treści	1
Cel zajęć.....	1
Rozpoczęcie	1
Jak wypełnić to sprawozdanie?	1
Pobranie i uruchomienie PHP	2
Inicjalizacja projektu z wykorzystaniem Composer	3
Utworzenie i wykonanie programu	5
Zarządzanie zależnościami	7
Wykorzystanie różnych poziomów logowania Monolog	9
Dependency Injection	11
Podsumowanie.....	13

CEL ZAJĘĆ

Celem głównym zajęć jest zdobycie umiejętności tworzenia projektów oraz łączenia zależności z wykorzystaniem narzędzia Composer.

ROZPOCZĘCIE

Rozpoczęcie zajęć. Przedstawienie prowadzącego. Przedstawienie uczestników. Przedstawienie zasad laboratorium.

JAK WYPEŁNIĆ TO SPRAWOZDANIE?

Zapisz ten plik na dysku twardym jako kopię. Zmień nazwę pliku:

- grN na odpowiedni numer grupy (np. gr3),
- nazwisko-imie na Twoje dane bez polskich znaków.

Otwórz kolejno Plik -> Informacje -> Właściwości -> Właściwości zaawansowane -> Niestandardowe.

Zaktualizuj właściwości:

Właściwości:	Nazwa	Wartość	Typ
	Imie	Imie	Tekst
	Nazwisko	Nazwisko	Tekst
	Numer al...	00000	Tekst
	Kod kursu	AI2	Tekst
	Kod labor...	LAB A	Tekst
	Grupa	1	Liczba
	Wersja	1	Liczba

1

Czytaj tę instrukcję, wypełniaj polecenia, uzupełniaj zrzuty ekranu zgodnie z poleceniami.

Gotowe sprawozdanie wyślij w nieprzekraczalnym terminie **w postaci pliku PDF**.

POBRANIE I URUCHOMIENIE PHP

Zaloguj się do systemu Windows / pulpitu zdalnego `rdp.wi.zut.edu.pl`:

- spoza sieci ZUT potrzebny VPN: <https://uci.zut.edu.pl/uslugi-uci/vpn.html>;
- nazwa użytkownika: `WIAD\ab12345`
- komputer: `rdp.wi.zut.edu.pl`

Utwórz katalog `I:\AI2-lab`. Jeśli musisz umieścić ten folder gdzie indziej – upewnij się, że nie ma spacji i ogonków.

Odwiedź stronę <https://windows.php.net/download/>. Pobierz PHP 8.2.10 x64 NTS.

Wypakuj pobrane repozytorium do `I:\AI2-lab\php-8.2.10-nts-Win32-vs16-x64`.

Otwórz panel sterowania. W polu wyszukiwania wpisz `path`. Wybierz edycję zmiennych środowiskowych użytkownika. Znajdź zmienną `Path` i kliknij edycję. Dodaj ścieżkę `I:\AI2-lab\php-8.2.10-nts-Win32-vs16-x64`.

Skopiuj plik `I:\AI2-lab\php-8.2.10-nts-Win32-vs16-x64\php.ini-development` jako `php.ini`, po czym edytuj jego zawartość – odkomentuj poniższe ustawienia:

```
extension_dir = "ext"
...
extension=curl
extension=gd
extension=intl
extension=mbstring
extension=openssl
extension=pdo_sqlite
```

Utwórz katalog `I:\AI2-lab\labA`.

Otwórz ulubiony terminal (CMD, wiersz polecenia, PowerShell, Git Bash) i wejdź do katalogu `I:\AI2-lab\labA`.

Wykonaj komendę

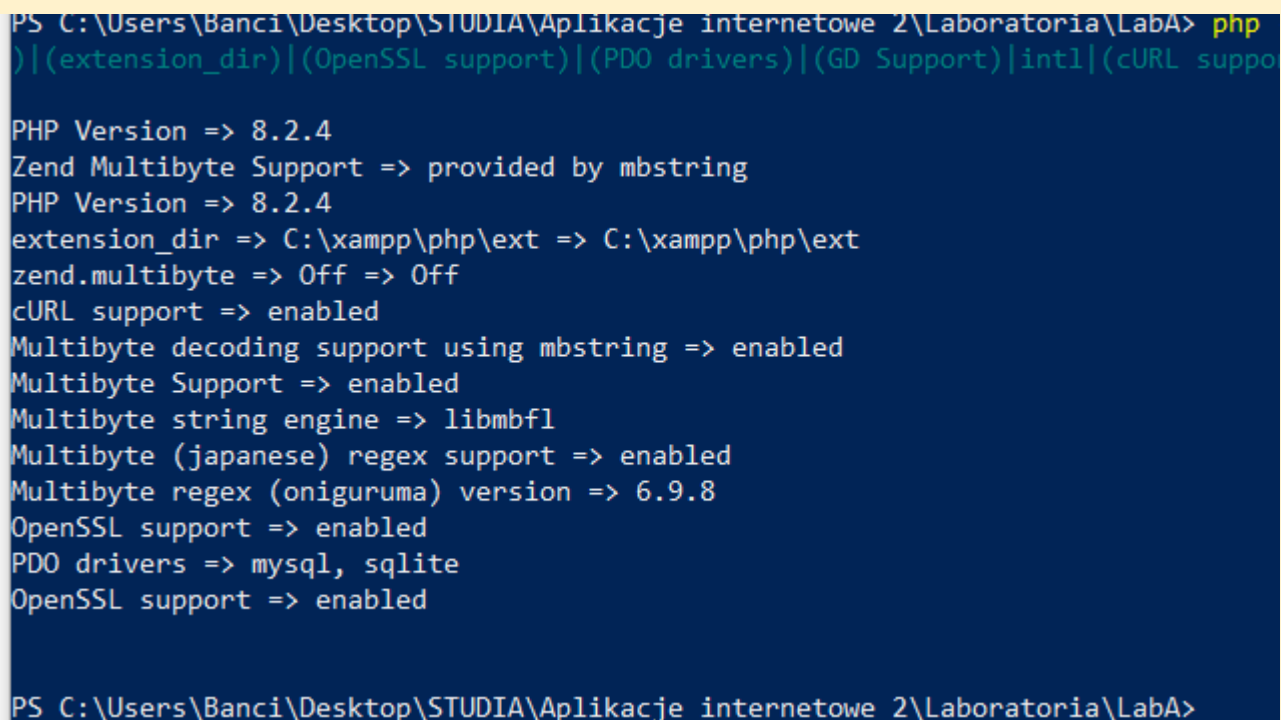
```
php -i | Select-String -Pattern '(PHP Version)|(extension_dir)|(OpenSSL support)|(PDO drivers)|(GD Support)|intl|(cURL support)|multibyte'
```

Oczekiwany wynik:

```
PS C:\Users\artur\workspace\AI2-lab\labA> php -i | Select-String -Pattern '(
PHP Version)|(extension_dir)|(OpenSSL support)|(PDO drivers)|(GD Support)|in
tl|(cURL support)|multibyte'

PHP Version => 8.2.10
Zend Multibyte Support => provided by mbstring
PHP Version => 8.2.10
extension_dir => ext => ext
zend.multibyte => Off => Off
cURL support => enabled
GD Support => enabled
intl
intl.default_locale => no value => no value
intl.error_level => 0 => 0
intl.use_exceptions => Off => Off
Multibyte Support => enabled
Multibyte string engine => libmbfl
Multibyte (japanese) regex support => enabled
Multibyte regex (oniguruma) version => 6.9.8
OpenSSL support => enabled
PDO drivers => sqlite
OpenSSL support => enabled
```

Zastąp poniższy obrazek swoim zrzutem ekranu:



```
PS C:\Users\Banci\Desktop\STUDIA\Aplikacje internetowe 2\Laboratoria\LabA> php
)|(extension_dir)|(OpenSSL support)|(PDO drivers)|(GD Support)|intl|(cURL suppo

PHP Version => 8.2.4
Zend Multibyte Support => provided by mbstring
PHP Version => 8.2.4
extension_dir => C:\xampp\php\ext => C:\xampp\php\ext
zend.multibyte => Off => Off
cURL support => enabled
Multibyte decoding support using mbstring => enabled
Multibyte Support => enabled
Multibyte string engine => libmbfl
Multibyte (japanese) regex support => enabled
Multibyte regex (oniguruma) version => 6.9.8
OpenSSL support => enabled
PDO drivers => mysql, sqlite
OpenSSL support => enabled

PS C:\Users\Banci\Desktop\STUDIA\Aplikacje internetowe 2\Laboratoria\LabA>
```

Punkty:

0

1

INICJALIZACJA PROJEKTU Z WYKORZYSTANIEM COMPOSER

Przejdź terminalem i eksploratorem plików do katalogu I:\AI2-lab\labA.

Pobierz archiwum PHAR composera w wersji 2.6.3 do katalogu I:\AI2-lab\labA:

- <https://getcomposer.org/download/2.6.3/composer.phar>

Sprawdź wersję composera:

```
| php composer.phar --version
```

Zainicjuj projekt:

```
| php composer.phar init
```

Ważne ustawienia:

- package name: inazwisko/lab-composer
- author: Imie Nazwisko
- package type: project
- license: MIT
- interaktywne wyszukiwanie pakietów: nie
- PSR-4 mapping: ENTER (zostawić domyślne)

Zweryfikuj ustawienia i zatwierdź utworzenie projektu:

```
Package name (<vendor>/<name>) [artur/lab-a]: akarczmarczyk/lab-composer
Description []:
Author [Artur Karczmarczyk <artur@ideaspot.pl>, n to skip]:
Minimum Stability []:
Package Type (e.g. library, project, metapackage, composer-plugin) []: project
License []: MIT

Define your dependencies.

Would you like to define your dependencies (require) interactively [yes]? noWould you li
ke to define your dev dependencies (require-dev) interactively [yes]? no
Add PSR-4 autoload mapping? Maps namespace "Akarczmarczyk\LabComposer" to the entered re
lative path. [src/, n to skip]:

{
    "name": "akarczmarczyk/lab-composer",
    "type": "project",
    "license": "MIT",
    "autoload": {
        "psr-4": {
            "Akarczmarczyk\\LabComposer\\": "src/"
        }
    },
    "authors": [
        {
            "name": "Artur Karczmarczyk",
            "email": "artur@ideaspot.pl"
        }
    ],
    "require": {}
}

Do you confirm generation [yes]?
Generating autoload files
Generated autoload files
PSR-4 autoloading configured. Use "namespace Akarczmarczyk\LabComposer;" in src/
Include the Composer autoloader with: require 'vendor/autoload.php':
```

Na koniec wykonaj polecenie:

```
| php composer.phar install
```

Umieść poniżej zrzut ekranu z procesu inicjalizacji projektu composerem:

```

Do you confirm generation [yes]?
Generating autoload files
Generated autoload files
PSR-4 autoloading configured. Use "namespace Abancewicz\LabComposer;" in src/
Include the Composer autoloader with: require 'vendor/autoload.php';
PS C:\Users\Banci\Desktop\STUDIA\Aplikacje internetowe 2\Laboratoria\LabA> php composer.phar install
No composer.lock file present. Updating dependencies to latest instead of installing from lock file. See https://getcomp
oser.org/install for more information.
Loading composer repositories with package information
Updating dependencies
Nothing to modify in lock file
Writing lock file
Installing dependencies from lock file (including require-dev)
Nothing to install, update or remove
Generating autoload files
PS C:\Users\Banci\Desktop\STUDIA\Aplikacje internetowe 2\Laboratoria\LabA>

```

Umieść poniżej zrzut ekranu przedstawiający otrzymaną strukturę katalogów, z wykorzystaniem polecenia:

Get-ChildItem -Path . -Recurse -Force -ErrorAction SilentlyContinue | Select-Object
FullName

```

PS C:\Users\Banci\Desktop\STUDIA\Aplikacje internetowe 2\Laboratoria\LabA> Get-ChildItem -Path . -Recurse -Force -
ction SilentlyContinue | Select-Object FullName

FullName
-----
C:\Users\Banci\Desktop\STUDIA\Aplikacje internetowe 2\Laboratoria\LabA\src
C:\Users\Banci\Desktop\STUDIA\Aplikacje internetowe 2\Laboratoria\LabA\vendor
C:\Users\Banci\Desktop\STUDIA\Aplikacje internetowe 2\Laboratoria\LabA\AI2-LA-gr1-BANCEWICZ-ALEKSANDRA.docx
C:\Users\Banci\Desktop\STUDIA\Aplikacje internetowe 2\Laboratoria\LabA\composer.json
C:\Users\Banci\Desktop\STUDIA\Aplikacje internetowe 2\Laboratoria\LabA\composer.lock
C:\Users\Banci\Desktop\STUDIA\Aplikacje internetowe 2\Laboratoria\LabA\composer.phar
C:\Users\Banci\Desktop\STUDIA\Aplikacje internetowe 2\Laboratoria\LabA\~$2-LA-gr1-BANCEWICZ-ALEKSANDRA.docx
C:\Users\Banci\Desktop\STUDIA\Aplikacje internetowe 2\Laboratoria\LabA\vendor\composer
C:\Users\Banci\Desktop\STUDIA\Aplikacje internetowe 2\Laboratoria\LabA\vendor\autoload.php
C:\Users\Banci\Desktop\STUDIA\Aplikacje internetowe 2\Laboratoria\LabA\vendor\composer\autoload_classmap.php
C:\Users\Banci\Desktop\STUDIA\Aplikacje internetowe 2\Laboratoria\LabA\vendor\composer\autoload_namespaces.php
C:\Users\Banci\Desktop\STUDIA\Aplikacje internetowe 2\Laboratoria\LabA\vendor\composer\autoload_psr4.php
C:\Users\Banci\Desktop\STUDIA\Aplikacje internetowe 2\Laboratoria\LabA\vendor\composer\autoload_real.php
C:\Users\Banci\Desktop\STUDIA\Aplikacje internetowe 2\Laboratoria\LabA\vendor\composer\autoload_static.php
C:\Users\Banci\Desktop\STUDIA\Aplikacje internetowe 2\Laboratoria\LabA\vendor\composer\ClassLoader.php
C:\Users\Banci\Desktop\STUDIA\Aplikacje internetowe 2\Laboratoria\LabA\vendor\composer\installed.json
C:\Users\Banci\Desktop\STUDIA\Aplikacje internetowe 2\Laboratoria\LabA\vendor\composer\installed.php
C:\Users\Banci\Desktop\STUDIA\Aplikacje internetowe 2\Laboratoria\LabA\vendor\composer\InstalledVersions.php
C:\Users\Banci\Desktop\STUDIA\Aplikacje internetowe 2\Laboratoria\LabA\vendor\composer\LICENSE

PS C:\Users\Banci\Desktop\STUDIA\Aplikacje internetowe 2\Laboratoria\LabA>

```

Punkty:	0	1
---------	---	---

UTWORZENIE I WYKONANIE PROGRAMU

Otwórz katalog I:\AI2-lab\labA za pomocą PhpStorm. Utwórz dwa pliki, zmieniając odpowiednio przestrzeń nazw:

```

<?php // src/Duck.php
namespace Abancewicz\LabComposer;

class Duck
{

```

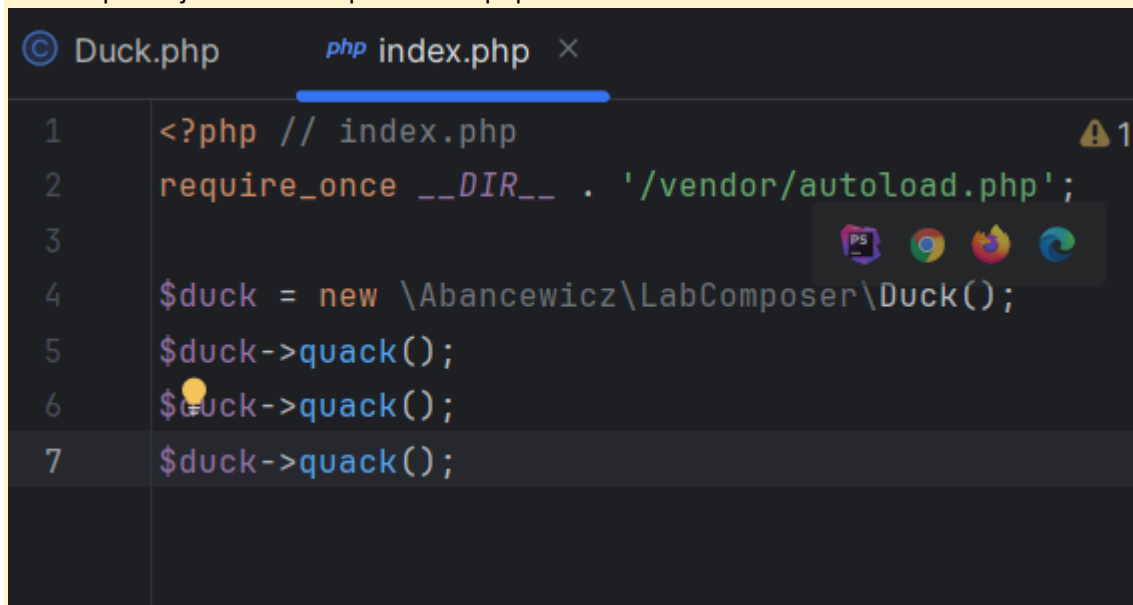
```
public function quack()  
{  
    echo "Quack\n";  
}  
}
```

```
<?php // index.php  
require_once __DIR__ . '/vendor/autoload.php';  
  
$duck = new \Abancewicz\LabComposer\Duck();  
$duck->quack();  
$duck->quack();  
$duck->quack();
```

Następnie z poziomu terminala wykonaj program:

```
labA>php .\index.php  
Quack  
Quack  
Quack
```

Umieść poniżej zrzut ekranu pliku index.php:



Umieść poniżej zrzut ekranu pliku src/Duck.php:

```

1  <?php // src/Duck.php
2  namespace Abancewicz\LabComposer;
3
4  1 usage
5  class Duck
6  {
7      3 usages
8      public function quack()
9      {
10         echo "Quack\n";
11     }
12 }

```

Umieść poniżej zrzut ekranu wywołania i działania programu (komenda `php index.php`):

```

PS C:\Users\Banci\Desktop\STUDIA\Aplikacje internetowe 2\Laboratoria\LabA> php .\index.php
Quack
Quack
Quack
PS C:\Users\Banci\Desktop\STUDIA\Aplikacje internetowe 2\Laboratoria\LabA>

```

Punkty:	0	1
---------	---	---

ZARZĄDZANIE ZALEŻNOŚCIAMI

Przyjrzyj się strukturze programu, w szczególności plikom `composer.json` i `composer.lock`. Oba pliki muszą być commitowane. Dlaczego?

Zapoznaj się z zawartością katalogu `vendor`. Katalog `vendor` nie powinien być commitowany. Dlaczego?

Zainstaluj i odinstaluj pakiet `monolog/monolog`. Zbadaj jak zmienia się zawartość `composer.json`, `composer.lock` i katalogu `vendor`.

```

php composer.phar require monolog/monolog
php composer.phar remove monolog/monolog

```

AI2 LAB A – Bancewicz Aleksandra – Wersja 1
Ponownie zainstaluj pakiet `monolog/monolog`:

```
| php composer.phar require monolog/monolog
```

Po czym skasuj katalog `vendor` i spróbuj uruchomić program. Czy działa?

```
| php index.php
```

Wykonaj `composer install` i zbadaj zawartość katalogu `vendor`. Spróbuj ponownie uruchomić program.

Omów jak zmienia się zawartość plików `composer.json`, `composer.lock` i katalogu `vendor`:

a) po zainstalowaniu pakietu `monolog/monolog`:

Composer dodaje nowy wpis w sekcji "require" pliku `composer.json`

b) po odinstalowaniu pakietu `monolog/monolog`:

wpis dotyczący `monolog/monolog` zostaje usunięty z sekcji "require"

Dlaczego po zainstalowaniu pakietu `monolog/monolog` i skasowaniu katalogu `vendor` aplikacja przestała się uruchamiać? Wyjaśnij. Umieść zrzut ekranu.

Katalog `vendor` zawiera wszystkie pliki zależności potrzebne do działania aplikacji, w tym biblioteki zewnętrzne, autoloader Composera (`autoload.php`), a także wszystkie zainstalowane pakiety, w tym `monolog/monolog`. Gdy katalog `vendor` zostanie usunięty, aplikacja traci dostęp do tych zależności, w tym do autoloadera.

W efekcie, aplikacja nie będzie mogła się uruchomić, ponieważ brakuje jej podstawowych komponentów, które są przechowywane w katalogu `vendor`.


```

Quack
Quack
PS C:\Users\Banci\Desktop\STUDIA\Aplikacje internetowe 2\Laboratoria\LabA> php .\index.php
PHP Warning:  require_once(C:\Users\Banci\Desktop\STUDIA\Aplikacje internetowe 2\Laboratoria\LabA\vendor/autoload.php): Failed to open stream: No such file or directory in C:\Users\Banci\Desktop\STUDIA\Aplikacje internetowe 2\Laboratoria\LabA\index.php on line 2

Warning: require_once(C:\Users\Banci\Desktop\STUDIA\Aplikacje internetowe 2\Laboratoria\LabA\vendor/autoload.php): Failed to open stream: No such file or directory in C:\Users\Banci\Desktop\STUDIA\Aplikacje internetowe 2\Laboratoria\LabA\index.php on line 2
PHP Fatal error:  Uncaught Error: Failed opening required 'C:\Users\Banci\Desktop\STUDIA\Aplikacje internetowe 2\Laboratoria\LabA\vendor/autoload.php' (include_path='C:\xampp\php\PEAR') in C:\Users\Banci\Desktop\STUDIA\Aplikacje internetowe 2\Laboratoria\LabA\index.php:2
Stack trace:
#0 {main}
   thrown in C:\Users\Banci\Desktop\STUDIA\Aplikacje internetowe 2\Laboratoria\LabA\index.php on line 2

Fatal error: Uncaught Error: Failed opening required 'C:\Users\Banci\Desktop\STUDIA\Aplikacje internetowe 2\Laboratoria\LabA\vendor/autoload.php' (include_path='C:\xampp\php\PEAR') in C:\Users\Banci\Desktop\STUDIA\Aplikacje internetowe 2\Laboratoria\LabA\index.php:2
Stack trace:
#0 {main}
   thrown in C:\Users\Banci\Desktop\STUDIA\Aplikacje internetowe 2\Laboratoria\LabA\index.php on line 2
PS C:\Users\Banci\Desktop\STUDIA\Aplikacje internetowe 2\Laboratoria\LabA>

```

Punkty:	0	1
---------	---	---

WYKORZYSTANIE RÓŻNYCH POZIOMÓW LOGOWANIA MONOLOG

W tej części wykorzystamy bibliotekę monolog do logowania komunikatów i błędów. Zmodyfikuj kod `index.php`, żeby dodać logowanie do pliku `monolog.log`:

```

<?php // index.php
require_once __DIR__ . '/vendor/autoload.php';

$ds = DIRECTORY_SEPARATOR;
$log = new \Monolog\Logger("my_log");
$log->pushHandler(new \Monolog\Handler\StreamHandler(__DIR__ . $ds . 'monolog.log',
\Monolog\Logger::ERROR));
$log->error("This is some error.");
$log->warning("This is some warning.");
$log->notice("This is some notice.");

```

```
$log->info("This is some info.");
$log->debug("This is some debug.");

$duck = new \Akarczmarczyk\LabComposer\Duck();
$duck->quack();
$duck->quack();
$duck->quack();
```

Uruchom program:

```
php index.php
```

Sprawdź zawartość pliku `monolog.log`.

Kolejno zmieniaj `ERROR` w kodzie na `WARNING`, `NOTICE`, `INFO` i `DEBUG` i uruchamiaj program. Omów wpływ zmiany na liczbę zapisywanych logów. Omów korzyści praktyczne płynące z umieszczania funkcji logujących w całym programie i przełączania poziomu logów w jednym miejscu.

Monolog, który jest używany do logowania w aplikacjach PHP, obsługuje różne poziomy logów. Zmienianie poziomu logowania wpływa na to, jakie informacje są zapisywane w logach.

ERROR: Zapisuje błędy, które mogłyby spowodować awarię aplikacji lub jej nieprawidłowe działanie. To poważne sytuacje, które wymagają natychmiastowej uwagi.

WARNING: Zapisuje ostrzeżenia o problemach, które mogą prowadzić do błędów, ale na razie nie powodują krytycznych skutków. Warto je monitorować.

NOTICE: Mniej istotne niż ostrzeżenia, ale nadal ważne, np. informacje o mniej poważnych błędach lub nieefektywnym kodzie.

INFO: Zapisuje ogólne informacje o działaniu aplikacji, które mogą być przydatne do analizy, ale nie są krytyczne.

DEBUG: Zapisuje szczegółowe informacje, które są pomocne podczas debugowania aplikacji, np. dane wejściowe, zmienne, kontrola przepływu kodu.

Wstaw reprezentatywny fragment pliku `monolog.log`:

```

1 [2024-10-11T09:43:49.226220+02:00] my_log.ERROR: This is some error. [] []
2 [2024-10-11T09:46:14.482682+02:00] my_log.ERROR: This is some error. [] []
3 [2024-10-11T09:46:44.485653+02:00] my_log.WARNING: This is some warning. [] []
4 [2024-10-11T09:46:36.368796+02:00] my_log.ERROR: This is some error. [] []
5 [2024-10-11T09:46:36.371477+02:00] my_log.WARNING: This is some warning. [] []
6 [2024-10-11T09:46:36.371564+02:00] my_log.NOTICE: This is some notice. [] []
7 [2024-10-11T09:46:51.482478+02:00] my_log.ERROR: This is some error. [] []
8 [2024-10-11T09:46:51.485127+02:00] my_log.WARNING: This is some warning. [] []
9 [2024-10-11T09:46:51.485212+02:00] my_log.NOTICE: This is some notice. [] []
10 [2024-10-11T09:46:51.485277+02:00] my_log.INFO: This is some info. [] []
11 [2024-10-11T09:47:03.175611+02:00] my_log.ERROR: This is some error. [] []
12 [2024-10-11T09:47:03.178499+02:00] my_log.WARNING: This is some warning. [] []
13 [2024-10-11T09:47:03.178587+02:00] my_log.NOTICE: This is some notice. [] []
14 [2024-10-11T09:47:03.178632+02:00] my_log.INFO: This is some info. [] []
15 [2024-10-11T09:47:03.178671+02:00] my_log.DEBUG: This is some debug. [] []
16

```

Punkty:	0	1
---------	---	---

DEPENDENCY INJECTION

Dependency injection to technika programistyczna wchodząca w skład architektury heksagonalnej, który umożliwia uniezależnienie klasy od jej zależności. W tej sekcji wstrzykniemy do klasy `Duck` loggera, aby można było logować zdarzenia w instancjach tej klasy.

Zmodyfikuj klasę `Duck.php`, żeby:

- wykorzystywała technikę `Dependency Injection` do przekazywania obiektu klasy `LoggerInterface`;
- logowała (poziom `INFO`) stworzenie klasy `Duck`;
- logowała (poziom `DEBUG`) wykonanie metody `Duck::quack()`.

Następnie przetestuj uruchamianie kodu źródłowego z poziomami logowania `ERROR`, `WARNING`, `INFO`, `DEBUG`.

```
<?php // src/Duck.php
namespace Akarczmarczyk\LabComposer;

use Psr\Log\LoggerInterface;

class Duck
{
    /** @var LoggerInterface */
    private $logger;

    public function __construct(LoggerInterface $logger = null)
    {
        $this->logger = $logger;
        if ($this->logger) {
            $this->logger->info("Duck created.");
        }
    }

    public function quack()
    {
        if ($this->logger) {
            $this->logger->debug("Quack() executed.");
        }
        echo "Quack\n";
    }
}
```

```
<?php // index.php
require_once __DIR__ . '/vendor/autoload.php';

$ds = DIRECTORY_SEPARATOR;
$log = new \Monolog\Logger("my_log");
$log->pushHandler(new \Monolog\Handler\StreamHandler(__DIR__ . $ds . 'log.log',
\Monolog\Logger::ERROR));
```


```

$log->error("error");
$log->warning("warning");

$duck = new \Akarczmarczyk\LabComposer\Duck($log);
$duck->quack();
$duck->quack();
$duck->quack();

```

Kolejno zmieniaj ERROR w kodzie na WARNING, NOTICE, INFO i DEBUG i uruchamiaj program. Omów wpływ zmiany na liczbę zapisywanych logów. Zamieść odpowiedni wycinek pliku monolog.log:



```

13 [2024-10-11T09:47:03.178587+02:00] my_log.NOTICE: This is some notice. [] []
14 [2024-10-11T09:47:03.178632+02:00] my_log.INFO: This is some info. [] []
15 [2024-10-11T09:47:03.178671+02:00] my_log.DEBUG: This is some debug. [] []
16 [2024-10-11T10:11:27.011660+02:00] my_log.ERROR: error [] []
17 [2024-10-11T10:11:27.014275+02:00] my_log.WARNING: warning [] []
18 [2024-10-11T10:11:27.014347+02:00] my_log.INFO: info [] []
19 [2024-10-11T10:11:27.014412+02:00] my_log.DEBUG: debug [] []
20 [2024-10-11T10:11:41.086014+02:00] my_log.ERROR: error [] []
21 [2024-10-11T10:11:41.088674+02:00] my_log.WARNING: warning [] []
22 [2024-10-11T10:11:41.088768+02:00] my_log.INFO: info [] []
23 [2024-10-11T10:11:41.088811+02:00] my_log.DEBUG: debug [] []
24 [2024-10-11T10:11:41.089217+02:00] my_log.INFO: Duck created. [] []
25 [2024-10-11T10:11:41.089311+02:00] my_log.DEBUG: Quack() executed. [] []
26 [2024-10-11T10:11:41.089574+02:00] my_log.DEBUG: Quack() executed. [] []
27 [2024-10-11T10:11:41.089835+02:00] my_log.DEBUG: Quack() executed. [] []
28

```

Zmieniając poziom logowania z ERROR na WARNING, NOTICE, INFO i DEBUG wpływam na to, które logi są zapisywane w pliku. Monolog rejestruje logi tylko dla poziomu, który jest równy lub bardziej szczegółowy niż ustawiony w handlerze.

Monolog ma 5 poziomów logowania (od najbardziej szczegółowego do najbardziej ogólnego):

DEBUG — najniższy poziom, loguje szczegóły, które są przydatne w czasie debugowania.

INFO — używane do informacyjnych komunikatów, które mogą być pomocne w analizie aplikacji, ale nie są błędami.

NOTICE — mniej istotne komunikaty niż INFO, ale mogą wskazywać na potencjalny problem.

WARNING — ostrzeżenia o potencjalnych problemach, które nie są błędami.

ERROR — błędy, które mogą wpływać na działanie aplikacji.

CRITICAL, ALERT, EMERGENCY — poziomy wyższe niż ERROR (nie są używane w tym przykładzie).

Punkty:	0	1
---------	---	---

PODSUMOWANIE

W kilku zdaniach podsumuj zdobyte podczas tego laboratorium umiejętności.

Podczas tego laboratorium zdobyłam umiejętności związane z zarządzaniem zależnościami w PHP przy użyciu narzędzia Composer, w tym instalowaniem i odinstalowywaniem pakietów, a także rozumieniem roli plików composer.json i composer.lock. Nauczyłam się również, jak korzystać z technik logowania, w tym integracji Monolog z aplikacją, a także jak zastosować technikę Dependency Injection do wstrzykiwania zależności, takich jak logger, do klas. Poznałam różnice między poziomami logowania (np. ERROR, WARNING, INFO, DEBUG) i ich wpływ na ilość i rodzaj zapisywanych logów. Dzięki tym doświadczeniom zyskałam większą biegłość w organizowaniu i kontrolowaniu zależności oraz logowania w aplikacjach PHP.

Zweryfikuj kompletność sprawozdania. Utwórz PDF i wyślij w terminie.