



## **SYLABUS PRZEDMIOTU**

<b>Nazwa przedmiotu:</b>	<b>Warsztat Programisty</b>
<b>Kod przedmiotu:</b>	<b>WPR</b>
<b>Kierunek / Profil:</b>	Informatyka / praktyczny
<b>Tryb studiów:</b>	stacjonarny
<b>Rok / Semestr:</b>	1 / 1
<b>Charakter:</b>	obowiązkowy
<b>Odpowiedzialny:</b>	mgr inż. Adam Urbanowicz
<b>Wersja z dnia:</b>	19.02.2026

### **1. Godziny zajęć i punkty ECTS**

<b>Wykłady</b>	<b>Ćwiczenia</b>	<b>Laboratoria</b>	<b>Z prowadzącym</b>	<b>Praca własna</b>	<b>Łącznie</b>	<b>ECTS</b>
16 h	—	16 h	32 h	68 h	100 h	4

### **2. Cel dydaktyczny**

Celem przedmiotu jest wykształcenie u studentów praktycznych umiejętności pracy warsztatowej programisty, w szczególności w zakresie korzystania z narzędzi programistycznych, debugowania kodu, pracy zespołowej z wykorzystaniem systemu kontroli wersji Git oraz platformy GitHub. Zajęcia mają na celu zapoznanie studentów z podstawami automatyzacji procesów (CI/CD), debugowania aplikacji internetowych z użyciem narzędzi deweloperskich przeglądarek, podstawami technologii webowych (HTML, CSS, JavaScript), a także wprowadzenie do świadomego i odpowiedzialnego wykorzystania narzędzi opartych na sztucznej inteligencji w pracy programisty. Dodatkowym celem jest przygotowanie studentów do budowania własnego portfolio programistycznego.

### **3. Treści programowe**

1. Organizacja pracy programisty: repozytorium kursowe, wymagane konta (GitHub), struktura materiałów, zasady oddawania prac i checklisty jakości.
2. Przegląd popularnych IDE i narzędzi: Visual Studio / VS Code / IntelliJ / Rider (porównanie), konfiguracja środowiska, rozszerzenia, formatowanie i linting.
3. Debugowanie kodu: podstawy debugera (breakpointy, step in/out/over, watch, call stack), praca na przykładach w aplikacji konsolowej.
4. Debugowanie błędów wykonania: wyjątki, logowanie, asercje, minimalny repro-case, czytanie stack trace, kultura zgłaszania błędów (issue).
5. Git: model pracy i podstawowe komendy (init/clone/status/add/commit/log/diff), ignorowanie plików (.gitignore), dobre komunikaty commitów.
6. Git: praca z repozytorium zdalnym (remote, fetch, pull, push), rozwiązywanie typowych problemów, praca w parach i code review (wstęp).
7. Branche w Git: tworzenie i przełączanie gałęzi, merge vs rebase (intuicja), konflikty i ich rozwiązywanie na prostych przykładach.
8. Pull Requesty na GitHub: workflow PR, szablon opisu, review, komentarze, poprawki, podstawy pracy z Issues i Projects (tablica zadań).
9. Wprowadzenie do prostych pipeline'ów: czym jest CI/CD, podstawowy workflow (np. build/test/lint), plik konfiguracyjny, uruchomienia automatyczne po push/PR.
10. Publikacja statycznej strony na GitHub Pages: struktura repo, branch/folder publikacji, podstawy konfiguracji, dodanie prostej strony kursowej.
11. DevTools w przeglądarce: inspekcja DOM/CSS, podgląd i modyfikacja stylów na żywo, box model, debugowanie responsywności.
12. DevTools zaawansowane: zakładka Network (żądania/odpowiedzi), Console (logowanie, błędy), Sources (breakpointy w JS), podstawy wydajności (Performance/Lighthouse).
13. Wstęp do HTML: struktura dokumentu, semantyka, formularze (podstawy), dostępność (a11y) i dobre praktyki.
14. Wstęp do CSS: selektory, kaskada i specyficzność, flexbox/grid (wprowadzenie), responsywność, organizacja stylów w projekcie.
15. Wstęp do JavaScript: typy i zmienne, funkcje, zdarzenia w DOM, proste operacje na elementach strony; wstęp do pracy z agentem AI w roli asystenta programisty (promptowanie, weryfikacja odpowiedzi, zasady bezpieczeństwa i etyki).

## 4. Efekty kształcenia

---

### Wiedza

- Student potrafi czytać ze zrozumieniem kod języków skryptowych, tworzyć skrypty oraz zweryfikować poprawność utworzonego kodu.
- Student potrafi projektować aplikacje zgodnie z paradigmatem strukturalnym i obiektowym. Student posiada podstawową wiedzę na temat protokołu HTTP oraz tworzenia bezpiecznych aplikacji internetowych i bazodanowych.
- Student zna odpowiednie narzędzia wspomagające tworzenie skryptów oraz wspomagające projektowanie aplikacji internetowych, bazodanowych

## Umiejętności

- Student potrafi wybrać odpowiednie środowisko programistyczne i inne narzędzia, które wspomagają w projektowaniu aplikacji oraz potrafi dobrać model procesu wytwarzania aplikacji do specyfiki przedsięwzięcia. Student potrafi posługiwać się wybranym środowiskiem oraz analizować poprawność działania stworzonej aplikacji.
- Student potrafi wykorzystać odpowiednie biblioteki do tworzenia aplikacji oraz wykorzystać narzędzia, które umożliwiają tworzenie, debbugowanie i uruchomienie projektów programistycznych. Student potrafi przedstawić stworzoną aplikację oraz omówić jej sposób działania.
- Student potrafi skonfigurować środowisko potrzebne do stworzenia skryptów i aplikacji internetowej i bazodanowej, tzn. posiada wiedzę na temat konfiguracji bazy danych oraz oprogramowania, które jest potrzebne do tworzenie projektu programistycznego.

## 5. Kryteria oceny

---

- Ćwiczenia / Laboratorium/Lektorat:
- rozwiązywanie zadań
- warsztaty
- Ćwiczenia/Laboratorium/Projekt/Lektorat/Seminarium
- Kryteria oceny
- Ćwiczenia/Laboratorium/Projekt/Lektorat
- Procentowy udział o ocenie ostatecznej:
  - laboratorium- 20%
  - projektu- 50%
  - kolokwium- 30%
- Uzyskanie pozytywnej oceny od 50% zdobytych punktów, które są pomnożone przez odpowiednie wagi procentowe, przy czym z kolokwium, należy uzyskać co najmniej 40% punktów.

## 6. Metody dydaktyczne

---

Wykład, laboratoria, praca własna studenta.

## 7. Literatura

---

### Podstawowa:

- Dokumentacja języka: [www.php.net](http://www.php.net)
- [w3schools.com](http://w3schools.com)
- Larry Ullman ,PHP i MySQL. Dynamiczne strony WWW. Szybki start, Helion 2018

### Uzupełniająca:

- Brak danych.

