Politechnika Gdańska
Wydział Elektroniki, Telekomunikacji
i Informatyki

**Studia podyplomowe**

# Aplikacje i usługi internetowe

Plan studiów w roku akademickim 2017/2018

Maksymalna liczba słuchaczy w grupie laboratoryjnej: 17
Maksymalna liczba słuchaczy w grupie projektowej: 24

*W przypadku, gdy liczba słuchaczy przekroczy powyższe ograniczenia przewiduje się utworzenie drugiej grupy laboratoryjnej lub projektowej.*

Zestawienie godzin zajęć dla poszczególnych przedmiotów podano w tabeli na następnej stronie. Prowadzenie godzin zajęć oznaczonych drukiem wytłuszczonym stanowią ofertę będącą przedmiotem niniejszego dokumentu. Godziny zajęć oznaczone drukiem pochyłym (kursywą) prowadzone będą przez pracowników Politechniki Gdańskiej.

Na następnych stronach podano programy dla poszczególnych przedmiotów.

Profil A: Junior Front-end Developer

| L.P. | Nazwa przedmiotu | Łączna liczba godzin przedmiotu | Wykłady | | | Laboratoria | | | Projekty | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Wykłady razem | PG | firmy zewnętrzne | Labor. razem | PG | firmy zewnętrzne | Projekty razem | PG | firmy zewnętrzne |
| 1 | ARCHITEKTURY SYSTEMÓW INTERNETOWYCH | *12* | *12* | *12* | | | | | | | |
| 2 | NARZĘDZIA PROGRAMISTYCZNE | *16* | | | | *16* | | **16** | | | |
| 3 | PODSTAWY INŻYNIERII OPROGRAMOWANIA | *18* | *8* | *8* | | | | | *10* | *10* | |
| 4 | DOKUMENTY CYFROWE | *30* | *10* | *10* | | *20* | | **20** | | | |
| 5 | TESTOWANIE OPROGRAMOWANIA | *10* | *2* | | **2** | *8* | | **8** | | | |
| 6 | PODSTAWY PROGRAMOWANIA | *42* | *12* | *12* | | *30* | | **30** | | | |
| 7 | ZAAWANSOWANE PROGRAMOWANIE W JĘZYKU SKRYPTOWYM | *42* | *12* | *12* | | *30* | | **30** | | | |
| 8 | PROJEKT GRUPOWY | *40* | | | | | | | *40* | | **40** |
| Razem | | *210* | *56* | *54* | 2 | *104* | | 104 | *50* | *10* | 40 |

**ARCHITEKTURY SYSTEMÓW INTERNETOWYCH**

Ilość godzin pracy na uczelni: 12 godz. wykładu
Ilość godzin pracy zdalnej (między zjazdami): 4

*Zakres materiału:*

1. Modele przetwarzania rozproszonego
2. Protokół TCP adresacja gniazda i porty
3. DNS
4. Komunikacja Klient-Serwer
   a. Systemy typu mainframe i „cienki" klient
   b. Aplikacje front-endowe, mobilne i „gruby" klient
5. Usługi webowe
   a. Architektura SOA - przykłady
   b. Usługi REST
   c. Interoperacyjność
6. Podstawy protokołu http w kontekście RESTful API
   a. Cechy
   b. Metody POST, GET
   c. nagłówki
   d. kodowanie base64, url encode
   e. MIME
7. Sesja internetowa
   a. Omówienie
   b. Typy
   c. Bezpieczeństwo charakterystyczne w kontekście aplikacji frontendowych: SSL/TLS, cookie, AJAX.
8. Serwer www
   a. Budowa serwera i jego konfiguracja
   b. Pamięć podręczna serwer/klient – współpraca
   c. Współpraca z rozszerzeniami php/nodejs
9. Modele aplikacji
   a. skalowalność i architektura sprzętowa
   b. Modele software'owe mvc, mvvc, mvp
10. Systemy chmurowe
11. Budowa przeglądarek
12. Techniczne aspekty funkcjonowania sieci Internet
13. Klasyfikacja sieci i usług
14. Realizacja połączeń sieciowych
15. Warstwowe architektury sieci
16. Protokoły IPv4 i IPv6

*Zagadnienia do realizacji w ramach pracy zdalnej między zjazdami (grupowa / indywidualna)*

Zaprojektowanie RESTowego API na zadany temat z uwzględnieniem:

- Błędów komunikacji

- Zarządzaniem pamięcią podręczną
- Odzwierciedlenia operacji na obiektach w protokole GET, POST, UPDATE
- projekt w http://raml.org

*Metodyka prowadzenia zajęć:*

Indywidualna praca słuchacza. Praca pomiędzy zjazdami wymaga częściowej pracy słuchaczy w grupach dwuosobowych – przegląd jakościowy wytworzonego kodu.

**NARZĘDZIA PROGRAMISTYCZNE**

Ilość godzin pracy na uczelni: 16 (zajęcia laboratoryjne)
Ilość godzin pracy zdalnej (między zjazdami): 6

*Zagadnienia – praca podczas zjazdów na uczelni:*

1. Praca z systemem kontroli wersji
    a. Tworzenie nowego repozytorium.
    b. Umieszczenie kodu w repozytorium.
    c. Wysłanie kodu na zdalny serwer.
    d. Wycofanie wprowadzonych zmian.
    e. Tworzenie gałęzi.
    f. Zmiana gałęzi.
    g. Zintegrowanie zmian z różnych gałęzi.
    h. Przegląd historii i stanu repozytorium.
    i. Praca z graficznym interfejsem do obsługi repozytorium.
2. Praca ze zintegrowanym środowiskiem programistycznym
    a. Tworzenie projektu.
    b. Wsparcie w wytwarzaniu kodu – asysta edytora.
    c. Praca z repozytorium.
    d. Nawigacja w narzędziu.
    e. Refaktoryzacja wytwarzanego kodu.
    f. Debugowanie wytwarzanego kodu.
    g. Tworzenie testów jednostkowych.
    h. Wykorzystywanie testów jednostkowych w wytwarzaniu kodu.
    i. Wykorzystanie narzędzi podnoszących jakość: inspekcje, JSHint.
3. Narzędzia budowania aplikacji: grunt/gulp/npm.
4. Narzędzie ciągłej integracji.

*Zagadnienia – praca zdalna między zjazdami (grupowa / indywidualna) Przygotowanie projektu z testami.*

1. Umieszczenie projektu na serwerze ciągłej integracji.
2. Zrefaktoryzowanie przygotowanego projektu.
3. Dołączenie poprawionego kodu do głównej gałęzi projektu.
4. Przegląd jakościowy wytworzonego kodu przez innego słuchacza.
5. Potwierdzenie na serwerze ciągłej integracji poprawności wytworzonego rozwiązania.

*Wymaganie sprzętowe / narzędziowe*

1. Serwer repozytorium kodu (GIT), udostępniony przez uczelnię lub darmowe rozwiązanie dla projektów Open Source (https://bitbucket.org, github.org).
2. Graficzne narzędzie do pracy z repozytorium SourceTree

3. Zintegrowane narzędzie programistyczne – WebStorm/PhpStorm
4. Travis Ci lub narzędzie udostępnione przez uczelnię

*Metodyka prowadzenia zajęć*
Zajęcia w grupach projektowych. Indywidualna praca słuchacza. Praca pomiędzy zjazdami wymaga częściowej pracy słuchaczy w grupach dwuosobowych – przegląd jakościowy wytworzonego kodu.

**PODSTAWY INŻYNIERII OPROGRAMOWANIA**

Ilość godzin pracy na uczelni 8 godz. wykładu, 10 godz. zajęć projektowych
Ilość godzin pracy zdalnej (między zjazdami) 32

*Zagadnienia – praca podczas zjazdów na uczelni:*

1. Wprowadzenie: Podstawowe pojęcia. Proces wytwarzania oprogramowania. Modelowanie, projektowanie, programowanie.
2. Cykl życia oprogramowania: Model kaskadowy. Podejście iteracyjne, model spiralny, metodyki zwinne.
3. Inżynieria wymagań: Specyfikacja wymagań systemowych.
4. Analiza problemu: Podstawowe modele UML
5. Projektowanie systemu: architektura system, interfejs użytkownika, logika biznesowa, warstwa danych
6. Wybrane problemy implementacji
7. Testowanie i pielęgnacja
8. Model SCRUM jako najpopularniejsza metodyka zwinna.

*Zagadnienia – praca zdalna między zjazdami (grupowa / indywidualna)*

1. Specyfikacja wymagań dla przykładowego projektu
2. Analiza przypadków użycia,
3. Modelowanie obiektowe klas
4. Projektowanie architektury systemu
5. Projektowanie logiki biznesowej
6. Projektowanie interfejsu użytkownika
7. Projektowanie struktury danych
8. Wstępna implementacja

*Wymaganie sprzętowe / narzędziowe*
Repozytorium GIT/SVN
Visual Paradigm

*Metodyka prowadzenia zajęć*
Wykład z prezentacją multimedialną. Projekt zespołowy w grupach 3-4 osobowych

**DOKUMENTY CYFROWE**

Ilość godzin pracy na uczelni: 10 godz. wykładu, 20 godz. zajęć laboratoryjnych
Ilość godzin pracy zdalnej (między zjazdami) 26

*Zagadnienia – praca podczas zjazdów na uczelni:*

Wykład:

1. Prezentacja stosu języków i technologii webowych z krótką charakterystyką elementów kluczowych dla tworzenia dokumentów cyfrowych i ich publikacji
2. Język HTML:
    a. Hipertekst i hipermedia,
    b. Historia i wersje języka HTML
    c. Struktura dokumentu, poprawność składniowa i semantyczna
    d. Przegląd najważniejszych znaczników HTML5 i ich atrybutów
3. Arkusze stylów CSS:
    a. Rola arkuszy CSS
    b. Metody dołączania stylów do dokumentów hipertekstowych
    c. Selektory elementów
    d. Przegląd najważniejszych atrybutów stylów z uwzględnieniem CSS 3
    e. Czcionki ikoniczne na przykładzie Font Awesome
    f. Ograniczenia stylów CSS i wprowadzenie do tematu preprocesorów CSS
    g. Typowe możliwości preprocesorów CSS na przykładzie rozwiązania Less
4. Interfejsy responsywne (RWD):
    a. Klasy urządzeń klienckich i ich charakterystyka
    b. Specyficzne wymagania dla witryn mobilnych
    c. Dyrektywy @media, progi responsywne
    d. Przykładowe rozwiązania: Bootstrap, Susy, grid natywny CSS3
    e. Podejście *mobile-first* w projektowaniu witryn internetowych
5. Systemy szablonów
    a. Wprowadzenie do narzędzia Mustache
    b. Budowanie strony WWW za pomocą systemu szablonów, danych z plików JSON, preprocesora CSS i menadżera zadań Grunt
6. Grafika w Internecie
    a. SVG:
        - Wprowadzenie do grafiki wektorowej
        - Rola SVG we współczesnym Internecie, kompatybilność w przeglądarkach
        - Dołączanie i osadzanie grafiki SVG w dokumentach hipertekstowych
        - Podstawowe znaczniki SVG
    b. Optymalizacja plików graficznych przy pomocy zewnętrznych programów/stron oraz zadań menadżera Grunt

Laboratoria:

1. HTML+CSS
    a. Budowanie dokumentów hipertekstowych, podstawowe elementy HTML
    b. Dołączanie stylów CSS
    c. Walidacja opracowanych plików
    d. Testowanie spójności wyglądu w różnych przeglądarkach
    e. CSS reset/Normalize.css
    f. Zastosowanie czcionek ikonicznych
    g. Zastosowanie grafiki SVG na stronie internetowej
2. Preprocesory stylów CSS
    a. Zastosowanie preprocesora CSS w aplikacji przygotowanej na wcześniejszym laboratorium
    b. Wprowadzenie elementów takich jak: zmienne i arytmetyka na nich, domieszki, domieszki z parametrami, zagnieżdżanie reguł CSS,
    c. Praca z preprocesorem na etapie wytwarzania aplikacji i generowanie statycznego arkusza CSS dla produkcyjnej instancji serwisu
3. Responsywne interfejsy
    a. Budowanie interfejsów o układzie gridowym na prostych przykładach
    b. Testy dla różnych klas urządzeń z użyciem narzędzi deweloperskich w przeglądarkach
4. Menadżery zadań – automatyzacja zadań dotycząca dokumentów cyfrowych (parsowanie, łączenie, minifikacja, uglifikacja, generowanie kodu CSS za pomocą preprocesora)
5. Systemy szablonów
    a. Wprowadzenie do narzędzia Mustache
    b. Budowanie strony WWW za pomocą systemu szablonów, danych z plików JSON, preprocesora CSS i menadżera zadań Grunt
6. Grafika w Internecie
    a. Grafika wektorowa SVG
    b. Optymalizacja plików graficznych przy pomocy zewnętrznych programów/stron oraz zadań menadżera Grunt

*Zagadnienia – praca zdalna między zjazdami*

Opracowanie projektu, który przy pomocy zadań menadżera Grunt automatycznie generuje wynikowy (zminifikowany i zwersjonowany) kod serwisu WWW - na podstawie szablonów HTML, danych z plików JSON oraz przy pomocy preprocesora SASS/LESS

*Metodyka prowadzenia zajęć*

Zajęcia laboratoryjne odbywać się będą w sali komputerowej (praca indywidualna przy komputerach) wyposażonej w ekran i rzutnik oraz w tablicę; stacje robocze wyposażone w oprogramowanie:

- przeglądarki internetowe: Chrome, Firefox, Internet Explorer
- edytory tekstowe: Brackets, Atom
- środowiska programistyczne: Netbeans, Eclipse
- dodatkowe narzędzia: git

**TESTOWANIE OPROGRAMOWANIA**

Ilość godzin pracy na uczelni: 2 godz. wykładu, 8 godz. laboratorium
Ilość godzin pracy zdalnej (między zjazdami) 16

*Zagadnienia – praca podczas zjazdów na uczelni:*

1. Debugowanie a testowanie – metody i środki
2. Techniki debugowania i śledzenia
3. Testowanie jednostkowe
4. Testowanie interfejsu użytkownika - automatyzacja
5. Testy obciążeniowe
6. Testy penetracyjne
7. Testowanie użyteczności
8. Testy akceptacyjne

*Zagadnienia – praca zdalna między zjazdami*

1. Testowanie jednostkowe
2. Testowanie interfejsu użytkownika
3. Testy obciążeniowe
4. Testy penetracyjne

*Wymaganie sprzętowe / narzędziowe* jasmine/selenium

*Metodyka prowadzenia zajęć*
Praca w laboratorium – prezentacja i praca własna; dalsza praca własna w domu; wynik – raport z testów.

**PODSTAWY PROGRAMOWANIA**

Ilość godzin pracy na uczelni 12 godz. wykładów, 30 godz. laboratorium
Ilość godzin pracy zdalnej (między zjazdami) 16

*Zagadnienia omawiane tylko na wykładach podczas zjazdów na uczelni*

1. Komputer jako urządzenie cyfrowe. rodzaje komputerów, model programowy komputera (procesor, pamięć, urządzenia zewn.). Procesory wielordzeniowe i wielowątkowe.
2. Przechowywanie i przetwarzanie informacji w postaci ciągów złożonych z zer i jedynek. Bity, bajty, słowa; koncepcje przechowywania liczb, znaków, dźwięków w postaci ciągów zerojedynkowych.
3. Hierarchia różnych typów pamięci, pamięć fizyczna i wirtualna, porządek bajtów w pamięci.
4. Ogólne koncepcje programowania - programowanie jako proces projektowania, tworzenia, testowania i utrzymywania kodu źródłowego programów komputerowych lub urządzeń mikroprocesorowych. Programowanie a inżynieria oprogramowania.
5. Wiedza potrzebna do programowania: umiejętność projektowania aplikacji, umiejętność analizy algorytmów, znajomość języków programowania i narzędzi programistycznych, znajomość zasad pracy kompilatorów.
6. Translacja i interpretacja programów
7. Kodowanie znaków, kody ASCII, Unicode, UTF-8, UTF-16, kodowanie quoted-printable, kodowanie base64.

*Zagadnienia omawiane na wykładach i na zajęciach laboratoryjnych – praca podczas zjazdów na uczelni*

1. Wstęp. Krótka historia. JavaScript jako język interpretowany. „Hello world!”. Składnia języka. Zagnieżdżanie skryptów w HTML, umieszczanie kodu w plikach JS.
2. Typy zmiennych, operatory, komentarze, funkcje, obiekty, zasięg zmiennych.
3. Instrukcje warunkowe: if, else, switch. Pętle: for, while, do, break, continue.
4. Zdarzenia: onLoad, onUnload, onFocus, onChange, onClick, onMouseOver, onMouseDown, onKeyPress, i inne.
5. Formularze. Walidacja danych. Okna dialogowe. Obsługa błędów.
6. Podstawy biblioteki jQuery. Zdarzenia, animacje.
7. Obiekt canvas. Rysowanie kształtów, edycja obrazków, animacje.

*Zagadnienia – praca zdalna między zjazdami:*

Zadania utrwalające i poszerzające treści przekazywane na zajęciach laboratoryjnych.

**ZAAWANSOWANE PROGRAMOWANIE W JĘZYKU SKRYPTOWYM**

Ilość godzin pracy na uczelni 12 godz. wykładów, 30 godz. laboratorium
Ilość godzin pracy zdalnej (między zjazdami) 16

*Zagadnienia – praca podczas zjazdów na uczelni:*

Wykład:

1. Wprowadzenie do ECMAScript 6
2. Wprowadzenie do języka TypeScript
   a. zmienne
   b. pętle
   c. funkcje
   d. interpolacja
   e. interfejsy
   f. transpilacja do języka JavaScript
3. Rozszerzenia reaktywne na przykładzie RxJS
   a. wzorzec projektowy Observator
   b. strumienie danych
   c. operatory
4. Wzorzec MV*:
   a. Składowe wzorca MVC
   b. Składowe wzorca MVVM
5. Angular jako przykład frameworka MVC po stronie klienta
   a. Podstawowe składowe
      - Widok jako projekcja szablonu
      - Kontrolery
      - Model danych
      - Zasięgi
   b. Dwukierunkowe wiązanie danych (*data binding*)
   c. Serwisy, wstrzykiwanie zależności
   d. Obsługa formularzy
   e. Routing, budowanie witryny z wielu widoków
   f. Wstęp do komunikacji z serwerem
6. Usługi typu REST
   a. HTTP jako protokół usług internetowych (przypomnienie z przedmiotu *Architektury systemów internetowych*)
      - Operacje (*HTTP verbs*) i ich semantyka
      - Kody odpowiedzi
      - Formaty danych, XML, JSON
   b. Wsparcie narzędziowe dla wytwarzania i konsumowania usług REST
      - Formaty Swagger, RAML, API Blueprints

- Generowanie opisu usługi
- Generowanie klienta na podstawie opisu

7. Bezpieczeństwo aplikacji internetowych
   a. Przegląd najczęściej spotykanych ataków na aplikacje internetowe
   b. Przyczyny podatności
   c. Strategie obrony
   d. Analiza luk bezpieczeństwa w przykładowych aplikacjach

Laboratoria:

1. ECMAScript 6
   - klasy, funkcje strzałkowe, typy danych, stałe, zmienne, mapy itp.
   - transpiler kodu Babel
2. TypeScript
   - zmienne
   - pętle
   - funkcje
   - interpolacja
   - interfejsy
   - kompilacja do JavaScript
3. RxJS
   - wzorzec projektowy Observator
   - strumienie danych
   - operatory
4. MV* Patterns
   - MVC
   - MVVM
5. Angular 2
   - Angular CLI i generowanie projektu
   - komponenty
   - serwisy
   - formularze, walidacja
6. REST API
   - komunikacja REST z zewnętrznym API, np. Firebase
7. Usługi w chmurze
   - uruchomienie własnego serwisu w chmurze Back& (https://www.backand.com)

Zagadnienia – praca zdalna między zjazdami (~~grupowa~~ / indywidualna): aplikacja MVC w przeglądarce wykorzystująca mikroserwisy po stronie serwera

*Metodyka prowadzenia zajęć*
– zajęcia praktyczne: w formie laboratoriów, praca indywidualna przy komputerach
– praca zdalna (między zjazdami): indywidualnie

**PROJEKT GRUPOWY**

Ilość godzin pracy na uczelni 40
Ilość godzin pracy zdalnej (między zjazdami) 18

*Zagadnienia – praca podczas zjazdów na uczelni:*

W ramach przedmiotu słuchacze mają za zadanie wytworzyć aplikacje w tematyce ustalonej z prowadzącym zajęcia. Zajęcia zawierają elementy pracy w metodyce Agile. Podczas wytwarzania aplikacji słuchacze mają za zadanie wykorzystać jak najwięcej elementów wiedzy zdobytej podczas zajęć.
Praca podczas projektu powinna zostać podzielona na co najmniej 3 etapy. Zespół określa zakres prac oraz jej podział na zadania. Zespół w porozumieniu z prowadzącym przydziela zadania do poszczególnych etapów.
Etap kończy się dostarczeniem zaplanowanego wcześniej fragmentu aplikacji. Zadania powinny być umieszczone w systemie zarządzania zadaniami. Zespół powinien na końcu etapu podsumować wykonane prace i zaproponować ulepszenia procesu wytwarzania aplikacji. Następnie wprowadzać ulepszenia w kolejnych etapach.

Rozwiązanie wytworzone podczas prac powinno spełniać następujące założenia:

1. Nie naruszać praw autorskich.
2. Korzystać wyłącznie z bibliotek i narzędzi, których licencja zezwala na to.
3. Być wytworzone z wykorzystaniem technologii: Angular, HTML5, CSS.
4. Powinno wykorzystywać zewnętrzne API – np. bazę danych (https://firebase.google.com/) albo inne publiczne API (https://www.publicapis.com)
5. Powinno być wersjonowane i składowane w repozytorium kodu.
6. Powinno zawierać testy jednostkowe do kluczowych funkcjonalności.
7. Powinno zostać umieszczone na serwerze ciągłej integracji, na której uruchamiane będą co najmniej testy jednostkowe.
8. Powinno mieć wytworzoną specyfikacje techniczną posiadającą:
    a. Opis funkcjonalności.
    b. Instrukcję instalacji.
    c. Instrukcję obsługi.
    d. Opis wymagań dla uruchomienia i działania aplikacji.

Zagadnienia – praca zdalna między zjazdami

1. Opracowanie tematu projektu.
2. Zaplanowanie i podzielenie pracy.
3. Wytworzenie rozwiązania.
4. Opracowanie dokumentacji.

*Wymaganie sprzętowe / narzędziowe*

1. System zarządzania zadaniami w metodyce Agile np. udostępniony przez uczelnię.
2. Serwer repozytorium kodu (GIT) udostępniony przez uczelnię lub darmowe rozwiązanie dla projektów Open Source (np. https://kask.eti.pg.gda.pl/gitlab_aui).
3. Zintegrowane narzędzie programistyczne – WebStorm.

*Metodyka prowadzenia zajęć*

Zajęcia odbywają się w grupach projektowych. Grupy pracują nad wytworzeniem aplikacji o tematyce zaproponowanej przez słuchaczy.
Prowadzący pełni rolę *Product Ownera* z metodyki Scrum.
W ramach zajęć na uczelni odbywać się będą spotkania przewidziane w metodyce Scrum.