Projekt i Implementacja Systemów Webowych/ Laboratorium

Mgr inż. Maciej Małecki (maciej.malecki@pwr.edu.pl)
Mgr inż. Adam Puchalski (adam.puchalski@pwr.edu.pl)

Konsultacje
Do ustalenia

Organizacja zajęć

Zajęcia laboratoryjne są prowadzone bez przerwy przez 1,5 godziny.

Ocena

- Skala ocen na zaliczenie: 5 (bdb), 4.5 (+db), 4 (db), 3.5 (+dst), 3 (dst), 2 (ndst).
- Zaliczenie na ocenę 5.5 (cel) może uzyskać student, który wykaże się wiedzą lub umiejętnościami znacznie wykraczającymi poza zakres przewidziany w programie nauczania.

Wymagania

- Na zajęcia należy przychodzić punktualnie.
- Materiał z zajęć, na których student nie był obecny, musi być opanowany, zadania wykonane.
- Na zajęcia należy przygotować się poprzez zapoznanie się z materiałem z wykładów i z wcześniej zapowiedzianymi tematami zajęć.
- Na zajęciach, zadanie realizowane będzie w 6 etapach.
- Kolejne etapy projektu oddaje się osobiście na kolejnych zajęciach.
- Etapów nie można oddawać na konsultacjach.
- Etap oddany na drugich zajęciach po terminie jest oceniana najwyżej na ocenę dostateczną (3,0), etap oddany po tym czasie oceniana jest na ocenę **0**.
- Szczegółowe zasady dotyczące pracy z kodem:
    - Wszyscy członkowie zespołu zobowiązani są do wprowadzania zmian bezpośrednio do prywatnego repozytorium zespołu.
    - Każdy członek zespołu zobowiązany jest utrzymywać repozytorium w stanie “zielonym” - CI po zmianach powinno być zielone, w przypadku błędów należy je niezwłocznie usuwać.
    - Podczas oddawania danego etapu zadania prowadzący sprawdza stan CI - niestabilne CI może być przyczyną niezaliczenia etapu.
    - Zmiany w repozytorium (kodzie aplikacji) powinny być odpowiednio i czytelnie komentowane.

- Zaimplementowany system powinien dać się uruchomić na dowolnym komputerze (laptopie) wyposażonym w system Microsoft Windows.

Istnieje możliwość propozycji własnych tematów projektów związanych z *Projekt i Implementacja Systemów Webowych*. Informacja należy podać na 2 zajęciach i uzyskać akceptację prowadzącego.

Plan zajęć

1. Zajęcia organizacyjne, omówienie zagadnienia, stos technologiczny, podział na zespoły robocze, przygotowanie środowiska pracy w ramach GitHub.com.
2. Akceptacja własnych tematów. Projekt aplikacji, podział na moduły funkcjonalne, planowanie pracy.
3. Frontend - widoki administracyjne.
4. Frontend - widoki użytkowe.
5. Backend - interfejs dla systemu zewnętrznego.
6. Backend - warstwa bazodanowa.
7. Integracja z aplikacją kliencką.
8. Prezentacja rozwiązania, ocena projektu.
9. Zajęcia końcowe, wpisy do indeksu

Kopiowanie prac innych studentów skutkuje automatycznie niezaliczeniem projektu!

Wpływ na ocenę będzie miało:

- Spełnienie wymagań funkcjonalnych i niefunkcjonalnych.
- Jakość rozwiązania – zarówno wewnętrzna jak i zewnętrzne.
- Strategia testowania i zastosowane narzędzia.
- Zastosowane narzędzia deweloperskie.

Konfiguracja środowiska developerskiego

1. Następujące narzędzia powinny być zainstalowane na lokalnej stacji roboczej:
    a. JDK 8 (zmienna JAVA_HOME musi być prawidłowo ustawiona w systemie)
    b. GIT (ewentualnie klient graficzny, np GitExtensions)
    c. IDE (sugerowane)
        i. Idea IntelliJ (frontend+backend)
        ii. Visual Studio Code (frontend+backend)
        iii. Eclipse (tylko backend)
2. Do realizacji projektu niezbędny jest dostęp do portalu github.com. Każdy student powinien utworzyć swoje prywatne konto na tym portalu. Ponieważ github.com jest portalem komercyjnym, należy utworzyć konto studenckie (https://education.github.com/). GitHub wymaga, aby adres e-mail studenta był z domeny edukacyjnej (np @pwr.wroc.pl).

3. Należy odpowiednio skonfigurować Git'a tak, aby dla commitera używany był adres e-mail użyty do założenia konta na GitHub.com.
4. Każdy ze studentów, korzystając ze swojego konta studenckiego, powinien dołączyć do organizacji pwr-piisw: https://github.com/pwr-piisw.
5. Każdy student powinien utworzyć sobie konto na portalu travis-ci.com korzystając ze studenckiego konta github.com.
6. Projekt powinien być realizowany w grupach 2-3 osobowych - na pierwszych zajęciach należy dokonać podziału na zespoły.
7. Każdy zespół powinien założyć dla siebie "team" w ramach organizacji pwr-piisw.
8. Każdy zespół powinien założyć jedno **prywatne** repozytorium kodu w ramach organizacji.
9. Do prywatnego repozytorium należy nadać uprawnienia Admin lub Write dla wszystkich członków zespołu (ale nie dla pozostałych studentów!).
10. Do prywatnego repozytorium należy nadać uprawnienia Read dla prowadzącego zajęcia (adampuchalskipwr, maciejmaleckipwr).
11. Do utworzonego właśnie repozytorium należy zaimportować zawartość repozytorium oasp-seed (z organizacji): https://github.com/pwr-piisw/oasp-seed.git
12. W ramach travis-ci.org należy uruchomić CI dla prywatnego repozytorium zespołu (wystarczy, gdy zrobi to jedna osoba).

Technologia

Projekt bazuje na platformie OASP (https://oasp.github.io/oasp). Punktem wyjściowym jest zalążek aplikacji OASP zawierający część serwerową opartą o SpringBoot oraz JPA oraz o część Frontendową opartą o Angular 4. Szczegółowe informacje na temat projektu zalążkowego można znaleźć w następnym punkcie.
Więcej informacji na temat wymaganych technologii można znaleźć w materiałach na wykłady: https://pwr-piisw.github.io/wyklady/#/
Aplikacja nie wymaga żadnego serwera aplikacji do działania - SpringBoot posiada wbudowany serwer HTTP i jest to wystarczające do pracy w ramach zajęć.

Dodatkowe informacje

- Więcej informacji o SpringBoot: https://projects.spring.io/spring-boot/
- Angular: https://angular.io/ (nie mylić z AngularJS!)
- Biblioteka styli CSS: http://getbootstrap.com/
- Biblioteka komponentów (opcjonalnie): https://www.primefaces.org/primeng/#/
- Testy jednostkowe Front-End: https://jasmine.github.io/

Projekt - Inteligentny dom

W tym punkcie przedstawiono przykładowy projekt do realizacji w ramach zajęć laboratoryjnych. Uczestnicy kursu mogą zaproponować inny temat projektu, o ile jego opis (w formie podobnej do zaprezentowanej w tym dokumencie) zostanie przedstawiony prowadzącemu najpóźniej na drugich zajęciach laboratoryjnych oraz zostanie przez niego zaakceptowany. Zaproponowany temat projektu powinien mieć zbliżoną złożoność do opisanego w tym dokumencie.

Opis zagadnienia

Inteligentny dom (lub w ogólności inteligentny budynek) to system integrujący czujniki oraz elektroniczne urządzenia będące wyposażeniem budynku w celu umożliwienia automatycznej pracy podsystemów oświetlenia, ogrzewania, wentylacji i podobnych. Systemy inteligentnego domu wykorzystuje się do poprawy komfortu i bezpieczeństwa korzystania z budynku a także do podniesienia efektywności energetycznej.

Prosty system domu inteligentnego pozwala na zbieranie informacji z czujników a także na wydawanie prostych dyspozycji (np. włącz światło, uruchom system zraszania trawników, zasuń rolety). Bardziej rozbudowane systemy umożliwiają zbudowanie mechanizmów, w których komendy wydawane są automatycznie, w zależności od warunków (zarówno wewnętrznych jak i zewnętrznych), pory dnia oraz roku.

Istotnym uzupełnieniem wyżej opisanego systemu jest możliwość zdalnego dostępu do funkcjonalności. Najprostszym a zarazem najbardziej funkcjonalnym rozwiązaniem jest udostępnienie odpowiedniego interfejsu (aplikacji webowej) w internecie, do której dostęp możliwy jest zarówno przy użycia zwykłej przeglądarki internetowej w komputerze, ale również z poziomu smartfonu. Właściciel systemu może to osiągnąć instalując odpowiedni serwer HTTP w ramach własnych zasobów bądź też zintegrować swój system z platformą oferującą taką funkcjonalność. Celem projektu jest zaimplementowanie takiej platformy.

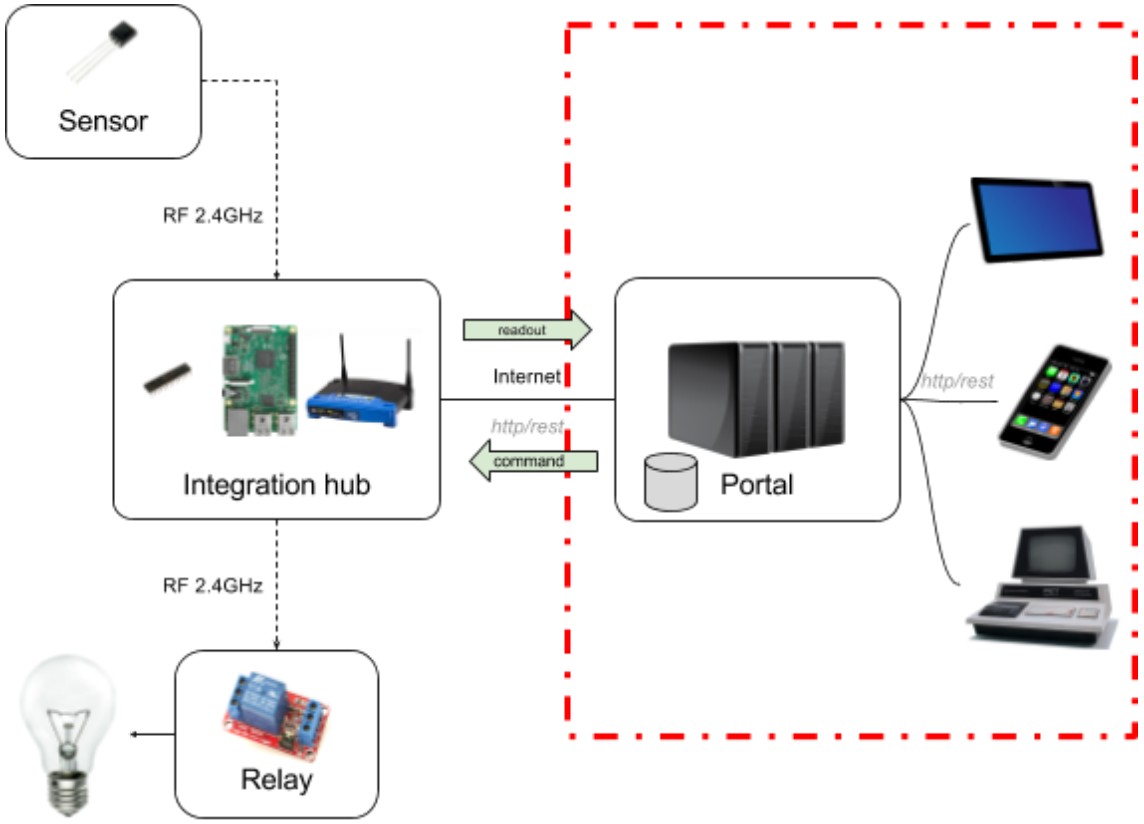
Cel projektu

Przy założeniu, że użytkownik dysponuje odpowiednią infrastrukturą (czujniki, sterowniki, moduł komunikacji radiowej oraz sterownik posiadający dostęp do internetu) celem projektu jest zaprojektowanie oraz implementacja systemu, który pozwoli użytkownikom odpowiedniej infrastruktury publikację danych systemu inteligentnego domu w internecie przy użyciu responsywnego interfejsu użytkownika (działającego na dowolnych urządzeniach, w tym smartfonach).

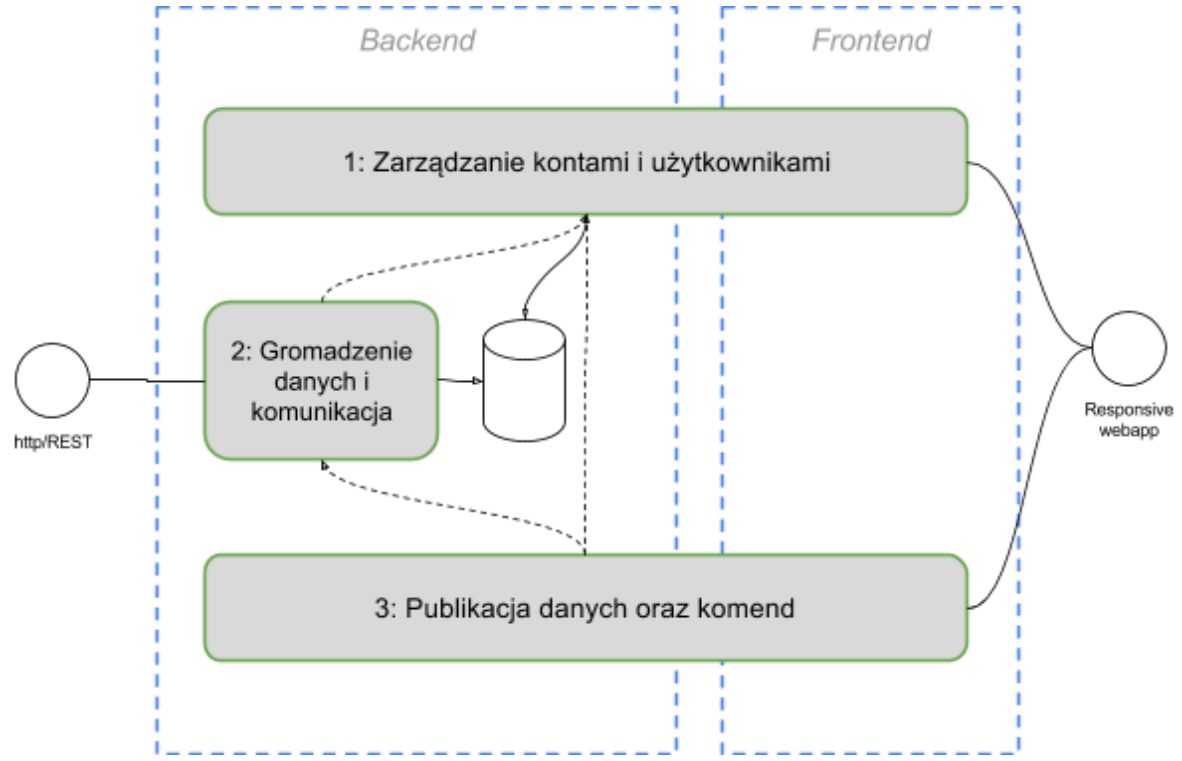
Realizując projekt należy abstrahować od konkretnej infrastruktury technicznej (jak czujniki i systemy sterowania domu inteligentnego). System taki można łatwo symulować stosując bezpośrednie wywołania interfejsu REST (np. używając dedykowanych narzędzi, takich jak SOAP UI i podobnych). Możliwe jest także użycie odpowiedniego symulatora (w postaci

aplikacji napisanej w Javie bądź JavaScript/Typescript) - nie jest to jednak wymagane do zaliczenia projektu (ale może mieć pozytywny wpływ na ocenę).

Zarys architektury



Zakres projektu zaznaczono czerwonym prostokątem. Portal zawiera mechanizm utrwalania danych (baza danych) oraz publikuje interfejs HTTP/REST dla systemów domu inteligentnego. System udostępnia także klienta webowego (Rich Web Client działający w przeglądarce internetowej), który komunikuje się z portalem także przy użyciu interfejsu HTTP/REST.



W projektowanym systemie zidentyfikowano trzy moduły funkcjonalne:

1. *Zarządzanie kontami i użytkownikami* - moduł administracyjny, pozwalający na tworzenie wielu kont (z jednym kontem można zintegrować dokładnie jeden system domu inteligentnego) oraz na nadawanie uprawnień dla innych użytkowników.
2. *Gromadzenie danych i komunikacja* - moduł komunikujący się z systemem domu inteligentnego odpowiedzialny za gromadzenie nadesłanych danych oraz pośredniczący w wysyłaniu komend.
3. *Publikacja danych oraz komend* - moduł odpowiedzialny za odpowiednią prezentację danych zgromadzonych przez moduł “2”.

Moduły “1” oraz “3” zawierają zarówno część serwerową jak i frontendową (webowa aplikacja responsywna). Moduł “2” zawiera jedynie część serwerową, ale dodatkowo publikuje interfejs REST służący do integracji z systemami domu inteligentnego.

Wszystkie moduły implementujemy w ramach pojedynczej aplikacji, część backend oraz frontend jest rozdzielona w naturalny sposób technologicznie. Część backendowa komunikuje się z aplikacją webową za pośrednictwem wewnętrznego interfejsu RESTowego.

Moduły rozdzielamy fizycznie przy użyciu pakietowania (osobne foldery dla plików źródłowych). Rozdzielenie dotyczy także aplikacji webowej.

Aktorzy

W systemie zidentyfikowano trzech aktorów:

1. *Inteligentny dom* - zdalny system komunikujący się z użytkownikiem za pośrednictwem portalu.
2. *Lokalny administrator* - właściciel konta w portalu, zarządzający punktami kontrolnymi oraz nadający uprawnienia użytkownikom.
3. *Użytkownik* - osoba monitorująca stan odczytów punktów kontrolnych oraz wydająca rozkazy dla punktów kontrolnych (tam gdzie jest to wspierane).

User stories

1. Jako <**inteligentny dom>** chcę <**przesłać do przypisanego konta w portalu informację o odczycie parametru punktu kontrolnego>** w celu <**udostępnienia odczytu użytkownikowi>.**
2. Jako <**inteligentny dom>** chcę <**przyjąć rozkaz wydany za pośrednictwem portalu>** w celu **<wykonania go>**.
3. Jako <**lokalny administrator>** chcę <**nadać nazwę punktowi kontrolnemu, dla którego przesłano odczyt>** aby <**ułatwić jego identyfikację przez użytkownika>**.
4. Jako <**lokalny administrator>** chcę <**nadać użytkownikowi uprawnienie do odczytu stanu punktu kontrolnego>**.
5. Jako <**lokalny administrator>** chcę <**nadać użytkownikowi uprawnienie do wykonania rozkazu dla punktu kontrolnego>**.
6. Jako <**lokalny administrator>** chcę <**utworzyć konto użytkownika>** w celu <**udostępnienia mu funkcjonalności konta w portalu>**.
7. Jako <**lokalny administrator>** chcę <**utworzyć konto w portalu>**.
8. Jako <**lokalny administrator>** chcę **<zalogować się do portalu>** w celu **<zarządzania kontem>**.
9. Jako <**lokalny administrator>** chcę **<zobaczyć listę wszystkich punktów kontrolnych, dla których przesłano odczyty>.**
10. Jako <**użytkownik>** chcę **<zobaczyć udostępnione odczyty punktów kontrolnych>**.
11. Jako <**użytkownik>** chcę **<wydać rozkaz dla punktu kontrolnego** w celu **wykonania konkretnej akcji>**.

Protokół komunikacyjny

Każda realizacja portalu musi implementować interfejs REST wyspecyfikowany w tym dokumencie.

Prześlij odczyt

HTTP Method	`POST`
URI template	`/domotics/{portalId}/api/v1.0/readouts`
Request content type	`application/json`

Request body	<pre>{
	  "Hub": {
	  "Id": "3790f3c4-d79b-48a4-9299-c0860b395cea",
	  "Name": "Universal meter",
	  "Version": "1.0"
	  },
	  "Sensors": [
	  {
	  "SensorType": {
	  "Id": "1",
	  "Type": "Analog Temperature Sensor",
	  "Model": "LM35DZ",
	  "Unit": "Celsius"
	  },
	  "Readout": {
	  "Time": "2017-08-17 09:46:52",
	  "Value": "25.0"
	  }
	  },
	  {
	  "SensorType": {
	  "Id": "2",
	  "Type": "Humidity Sensor",
	  "Model": "HR202L",
	  "Unit": "Scalar"
	  },
	  "Readout": {
	  "Time": "2017-08-17 09:43:12",
	  "Value": "0.85"
	  }
	  }
	  ],
	  "ControlPoints": [
	  {
	  "ControlPointType": {
	  "Id": "3",
	  "Type": "One Channel Relay",
	  "Model": "SRD-05VDC-SL-C"
	  },
	  "Commands": [
	  {
	  "Id": "1",
	  "Name": "Channel 0 On"
	  },
	  {
	  "Id": "2",
	  "Name": "Channel 0 Off"
	  }
	  ],
	  "State": "1",
	  "URI": "http://127.0.0.1:8081/domotics/control/↲
	  3790f3c4-d79b-48a4-9299-c0860b395cea/3"
	  }
	  ]
	}</pre>

Response content type	`n/a`

Opis

1. Dla uproszczenia przyjmujemy, że żądanie “prześlij odczyt” przekazuje do portalu pełne metadane dotyczące pojedynczego komponentu systemu domu inteligentnego, łącznie z URI pozwalającym na zdalne wykonanie komendy za pośrednictwem portalu.
2. Sekcja “Sensors” zawiera odczyty czujników wraz z typem czujnika, rodzajem stosowanych jednostek oraz czasem dokonania odczytu i odczytaną wartością.
3. Sekcja “ControlPoints” zawiera listę elementów sterowalnych wraz z listą dostępnych komend dla każdego takiego elementu. Sekcja ta zawiera także aktualny stan każdego z elementów.

Wykonaj komendę

HTTP Method	`POST`
URI template	`Przesłane w polu URI dla elementu sterowalnego.`
Request content type	`application/json`

Request body	`{`
	`"CommandId": "1"`
	`}`

Response content type	`application/json`

Response body	`{`
	`"State": "1"`
	`}`

Opis

1. System domu inteligentnego publikuje interfejs akceptujący metodę POST wraz z przesłanymi danymi.
2. Zakładamy, że identyfikator elementu oraz identyfikator punktu kontrolnego są częścią URI.

3. Przesyłamy komendę do wykonania.