



Politechnika Krakowska im.
Tadeusza Kościuszki
Wydział Fizyki, Matematyki
i Informatyki



Technologie aplikacji internetowych

Projekt: 20: *Format transmisji danych. Formaty JSON, XML.
Tworzenie żądań. Przetwarzanie odpowiedzi z serwera,
wyświetlanie danych z odpowiedzi na stronie internetowej.*

Data oddania:
11-01-2017

Studia stacjonarne
Kierunek: *Informatyka*
Stosowana
Rok akademicki: 2016/2017

Wykonał: *Piotr*
Tomaszewski
Nr albumu: 104896

1. Cel projektu. Założenia i dziedzina projektowania

Celem tego projektu jest stworzenie strony internetowej w oparciu o HTML5, JavaScript oraz CSS3. Jeśli chodzi o stronę backend'ową to silnik strony napisany będzie w języku Java z wykorzystaniem serwletów. Strona ta ma zawierać w sobie zaawansowane wzorce projektowe. W zakres projektu wchodziły tematy związane z formatowaniem danych zapisanych jako JSON lub XML. Należy stworzyć żądania które odczytają lub zapiszą do pliku porcję danych. Jako że będzie to strona internetowa, to dane będą odpowiednio wyświetlane na tej stronie internetowej.

2. Charakterystyka środowiska implementacyjnego

Strona zostanie zbudowana przy użyciu języka JAVA z wykorzystaniem serwletów, oraz hipertekstowego języka znaczników, obecnie szeroko wykorzystywanego do tworzenia stron internetowych: HTML5 oraz wykorzystany został język służący do opisu formy prezentacji stron WWW: CSS3. Dodatkowo, To wizualizacji danych za pomocą animacji wykorzystano skryptowy język programowania JavaScript i jego framework: JQuery.

Projekt zostanie wykonany w NetBeans IDE 8.1 - w zintegrowanym środowisku programistycznym dla języka Java, którego głównym celem jest przyspieszenie budowy aplikacji Java, w tym również usług Web Services i aplikacji przeznaczonych na urządzenia mobilne. Poza standardową funkcjonalnością narzędzia możliwe jest również rozszerzenie go między innymi o wsparcie dla języków programowania C i C++, wsparcie użycia XML i schematów XML, BPEL i Java Web Services czy modelowania UML. NetBeans został napisany w Javie dzięki czemu jest bardzo elastyczny i można go uruchomić na różnych platformach systemowych (Windows, Linux).

3. Opis stosowanych technologii oraz rozwiązań

3.1. JAVA EE



Technologia Java to podstawa projektowania niemal każdego rodzaju aplikacji sieciowej — stanowi ogólnosięwiatowy standard wykorzystywany przy tworzeniu aplikacji dla urządzeń mobilnych, aplikacji wbudowywanych, gier, zawartości i treści internetowych oraz oprogramowania dla przedsiębiorstw. Społeczność ponad 9 milionów oddanych programistów przyczynia się do sprawnego tworzenia, implementacji oraz użytkowania aplikacji i usług.

Technologia Java była testowana, udoskonalana, rozszerzana i ulepszana przez oddaną społeczność programistów, architektów i entuzjastów. Została zaprojektowana tak, aby umożliwić programowanie przenośnych, wysoce wydajnych aplikacji dla możliwie najszerszego spektrum platform przetwarzania cyfrowego. Dzięki aplikacjom dostępnym w heterogenicznych środowiskach, firmy mogą dostarczać więcej usług oraz zwiększać produktywność, komunikatywność i współpracę użytkowników końcowych, radykalnie zmniejszając koszty związane z prowadzeniem działalności i utrzymywaniem aplikacji konsumenckich. Technologia Java stała się nieocenionym narzędziem dla programistów, umożliwiając im:

- uruchamianie programów, pisanych na jednej platformie, na praktycznie dowolnej innej platformie;
- tworzenie programów, które mogą być uruchamiane w przeglądarkach internetowych i mogą uzyskiwać dostęp do usług internetowych;
- opracowywanie aplikacji serwerowych obsługujących fora internetowe, sklepy internetowe, ankiety, formularze HTML i wiele innych;
- łączenie aplikacji lub usług wykorzystujących język Java w celu stworzenia aplikacji i usług ściśle dostosowanych do konkretnych potrzeb;
- pisanie skutecznych i wydajnych aplikacji dla telefonów komórkowych, odległych urządzeń przetwarzających, mikrokontrolerów, modułów bezprzewodowych, sensorów, bramek, produktów konsumenckich oraz praktycznie wszelkich innych urządzeń elektronicznych.

3.2. JSON / XML



JSON, to nie tyle technologia czy nowy język co inne spojrzenie na to co już istnieje. Być może nawet samo odkodowanie tego skrótu da już nam dużo więcej informacji: "Java Script Object Notation", czyli format wymiany danych,

podobnie jak XML. Jednak XML (eXtensible Markup Language) ma w porównaniu z JSONem [wymawiane jak angielskie imię "dzejson"] jeden minus – do jego wykorzystania w projektach webmasterkich najczęściej potrzebujemy dodatkowych klas i obiektów.

JSON jest jednym z nieformalnych sposobów przekazywania danych do aplikacji opartych o AJAX. W typowych przypadkach dane w formacie JSON są pobierane z serwera jako tekst przy wykorzystaniu obiektu *XMLHttpRequest* języka JavaScript, a następnie przekształcane w obiekt. Tekst powinien być kodowany za pomocą UTF-8, który jest w JSON domyślny.

Dostęp do danych w formacie JSON jest bardziej naturalny z poziomu języka JavaScript niż dostęp do tych samych danych w formacie XML, ponieważ JSON stanowi składniowy podzbiór języka JavaScript. Wbrew podobieństwu w nazwie nie jest już jednak tak naturalny dla Javy i wymaga stosowania specjalnych bibliotek. Ponadto w starych przeglądarkach może występować problem z analizą JSON-a, podczas gdy analiza XML przy przesyłaniu z zastosowaniem AJAX jest wbudowane w prawie wszystkie współczesne przeglądarki (choć sama implementacja AJAX nie jest spójna).

XML był początkowo jedynym językiem w usługach sieciowych, a więc dominującym przy sformalizowanej wymianie danych. Jednak w nowszych wersjach standardów dopuszczana jest także wymiana danych za pomocą JSON-a (np. w REST).

XML może być łatwiejszy przy czytaniu dowolnego fragmentu dokumentu - ze względu na nazwę znacznika widoczną także w końcowym znaczniku. Jednak z tego samego względu XML w praktyce zajmuje znacząco więcej miejsca niż analogiczny obiekt przesyłany za pomocą formatu JSON. Ilość przesyłanych danych jest z kolei szczególnie istotna w związku z rozwojem urządzeń mobilnych (w szczególności smartfonów).

3.3. HTML5



HTML5 jest najnowszą wersją standardu opisującego język HTML. Termin ten możemy zdefiniować na dwa sposoby:

- jest to nowa wersja języka HTML, zawierająca nowe elementy, atrybuty i zachowania,
- większy zestaw technologii, które pozwala na bardziej różnorodne i potężne tworzenie stron i aplikacji internetowych. Zestaw ten czasem nazywamy HTML5 & friends, jednak często skracamy do nazwy po prostu HTML5.

Zawarta poniżej treść przeznaczona jest do zastosowania przez wszystkich programistów, strona zawiera dużo informacji na temat technologii HTML5, która została opisana w kilku grupach podzielonych według ich funkcji.

- Semantyka: pozwala na bardziej precyzyjne opisanie zawartości.
- Komunikacja: pozwala w sposób nowoczesny na komunikację z serwerem.
- Offline & Storage: pozwala stronom internetowym na bardziej efektywne przechowywanie danych lokalnie i w trybie offline.
- Multimedia: odtwarzanie plików audio i wideo bezpośrednio z przeglądarki.

- Efekty i Grafika 2D/3D: pozwala w znacznie bardziej zróżnicowany sposób prezentować możliwości stron lub aplikacji internetowych.
- Wydajność & Integracja: zapewnia większą prędkość i lepszą optymalizację wykorzystania sprzętu komputerowego.
- Dostęp do urządzenia: zastosowanie w wielu urządzeniach wejścia i wyjścia.
- Style: pozwala autorom na tworzenie ładniejszych motywów.

3.4. CSS3



CSS3 to najnowsza wersja kaskadowych stylów stosowanych przy kodowaniu projektów stron internetowych. Upowszechnienie tej wersji oczekiwało wielu web developerów na całym świecie, a jest to związane z bardzo ciekawymi rozwiązaniami, które małym wysiłkiem mogą znacząco wpłynąć na jakość i szybkość kodowania stron internetowych. Pozwala na zaokrąglanie rogów, cienie, gradienty, przezroczystość, a także wielokolumnowe teksty na stronie www.

Przed wszystkim, wykorzystywanie CSS3 nie jest zależne od żadnych specjalnych nagłówek, czy oznaczeń jak to miało miejsce np. w HTML5. Po prostu wystarczy zacząć pisać nowe style w tym samym pliku css, który wykorzystywaliśmy do tej pory.

3.5. JavaScript



JavaScript jest skryptowym językiem programowania, stworzonym w latach 90-tych przez Brendana Eichę i firmę Netscape. Jest to język wykorzystywany na stronach WWW. Same skrypty odpowiadają za interaktywność poprzez reagowanie na zdarzenia, sprawdzanie formularzy czy za tworzenie elementów służących do nawigacji. Ważnym jest, aby podczas tworzenia i ulepszania strony www żaden z elementów nie przestał działać po wyłączeniu w przeglądarce internetowej obsługi skryptu. Sam skrypt nie ma pełnego dostępu do komputera obsługiwane przez użytkownika.

Za pomocą JavaScript można również tworzyć aplikacje. Aby umożliwić to użytkownikom fundacja Mozilla, organizacja non-profit założona w 2003 roku, umożliwia korzystanie ze środowiska zbudowanego z technologii JSLib, XBL (język znaczników), XUL (język zgodny z XML służący do opisywania interfejsu GUI i WEB) i XPCOM (międzyplatformowy model komponentów). Dzięki ww. technologiom możliwe jest stworzenie aplikacji, które wykorzystują zasoby systemowe, posiadają graficzny interfejs oraz dostosowują się do konkretnej platformy.

3.6. jQuery



jQuery to framework. Jest to zbiór funkcji, za pomocą których możemy osiągnąć zakładane efekty bez konieczności powtarzania zbędnego kodu. Jest to biblioteka napisana w języku JavaScript.

Frameworków do JavaScript jest sporo. Należą do nich między innymi Prototype, Moo Tools oraz Dojo. Każde z nich zawiera zarówno podstawowe narzędzia, jak i unikalne rozwiązania. Usprawniają pracę z DOM i umożliwiają tworzenie różnych efektów animacyjnych. jQuery

oferuje bardzo przejrzystą konstrukcję. Większość nazw funkcji jest logicznie powiązana w stosunku do zadań, jakie mają zostać przez nią wykonane. Także sposób uruchomienia i zapisu funkcji jest stosunkowo przyjazny użytkownikowi. Taki jest właśnie cel frameworków. Mają za zadanie przyspieszyć i ułatwić użytkownikowi pracę, dostarczając przy tym trafnych rozwiązań.

Dzięki jQuery możemy na przykład zablokować poszczególne elementy formularza. Wystarczy użycie specjalnej funkcji, która pobiera wszystkie elementy formularza i ustawi w nich atrybut disabled. Oczywiście potrzebna będzie nazwa formularza.

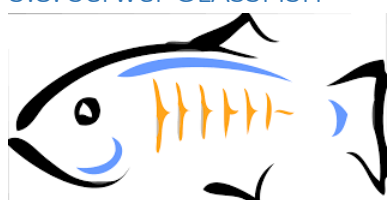
3.7. GIT



GIT to system plików zaprojektowany jako system kontroli wersji. Stworzył go Linus Torvalds jako narzędzie wspomagające rozwój jądra systemu operacyjnego Linux. GIT stanowi wolne oprogramowanie i został opublikowany pod ochroną licencji GNU GPL.

Git jest dobry do obsługi dużych projektów jak Linux, które mają po kilkanaście tysięcy plików, setki łat dziennie. Nazywany jest "stupid content tracker", ponieważ ma bardzo proste algorytmy scalania, prostsze niż w innych systemach kontroli wersji. Dzięki temu jest kilkukrotnie szybszy. Jeśli chcemy się dowiedzieć, czym się różnią dane wersje jądra, odpowie w mig. Jeżeli jednak będziemy chcieli znać historię zmian danego pliku, potrwa to trochę dłużej. Wynika to z tego, że git nie pracuje na poziomie plików, ale zestawów plików.

3.8. Serwer GLASSFISH



Serwer aplikacyjny dostępny od Sun Microsystems pierwotnie miał swe początki jako alians iPlanet (kooperacja inżynierów Sun Microsystems z NetScape). Nazwa ta została wprowadzona w 2002 roku i była przewidziana na grupę produktów, które powstały

w wyniku aliansu. Serwer GlassFish jest dostępny w dwóch wersjach – OpenSource oraz Enterprise. Z uwagi na JSON i XML będziemy pracować z wersją Enterprise. GlassFish jest referencyjną implementacją Serwera Aplikacji dla platformy Java EE. Zawiera referencyjną implementację specyfikacji JPA. Ponad to jest otwarty, dystrybuowany za licencji CDDL, a część elementów na licencji GNU GPL v2.

4. Rozwiązania (technologie, architektura) zapewniające skalowalność, wydajność i bezpieczeństwo.

Skalowalność pionowa

- Odpowiednia pamięć serwera
- Wystarczająco szybkie procesory serwera pozwalające przetworzyć żądania
- Optymalizacja oprogramowania w celu zmniejszenia czasu odpowiedzi serwera
- Cache'owanie treści

Skalowalność pozioma

- wykorzystanie serwera posiadającego mechanizm load-balancingu (rozproszenie obciążenia na inne procesory)
- wykorzystanie bazy danych
- wykorzystanie technologii przetwarzania w chmurze

Wydajność

- zmniejszenie wielkości strony
- Wykorzystanie technologii opartych na buforowaniu
- Kompresja danych
- Niezbyt duże rozmiary plików graficznych
- Unikanie przekierowań

Bezpieczeństwo

- Bezpieczne, autoryzowane połączenie (HTTPS)
- Weryfikacja osób mających dostęp do strony
- Odpowiednio zakodowane hasła
- Ograniczony dostęp do katalogów i plików
- Unikanie przechowywania poufnych danych
- Zabezpieczenie przed wpisywaniem hasła w nieskończoność.

5. Analiza wymagań, warstwa modelu architektury aplikacji.

Wymagania funkcjonalne

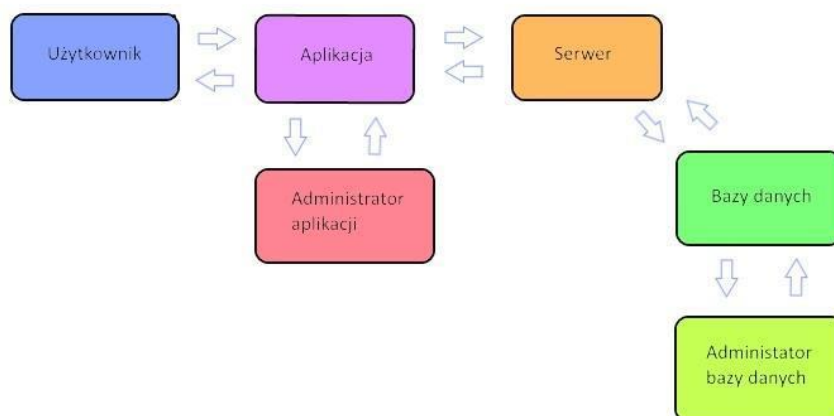
Analiza zebranych wymagań funkcjonalnych umożliwiła zidentyfikowanie i przedstawienie pożądanego zachowania systemu. Wymagania funkcjonalne ilustrują usługi, które ma oferować system oraz jednoznacznie identyfikują reakcje systemu na określone sytuacje. Te wymagania opisują możliwości systemu w zakresie zachowania oraz dostępnych operacji.

- Panel powitalny połączony wraz z głównym menu aplikacji, w którym możemy wybrać różne opcje spośród interesujących nas produktów.
- Na stronie głównej stworzenie listy najbardziej popularnych produktów odczytanych z pliku JSON
- Możliwość przeglądania produktów na podstronach pobranych z plików XML.
- Możliwość dodawania produktu do ulubionych.
- Logowanie się administratora strony w celu dokonywania zmian na produktach
- Zapisywanie zaktualizowanych produktów do plików JSON lub XML.

Wymagania niefunkcjonalne

- **Użyteczność** - Aplikacja posiada przyjazny, prosty w obsłudze interfejs. Dzięki temu nie jest potrzebne szkolenie użytkownika.
- **Niezawodność** - Aplikacja kliencka może być włączona przez cały czas. Aplikacja serwera może być wyłączona przez jedną godzinę w miesiącu w celu aktualizacji bazy danych lub poprawienia ewentualnych błędów lub plików.
- **Wydajność** - Serwer wymienia z klientem niewielkie paczki danych, dzięki temu liczba jednoczesnych użytkowników jest bardzo duża.
- **Bezpieczeństwo** - Aplikacja kliencka nie ma bezpośredniego dostępu do bazy danych. Jedynie serwer, który jest obsługiwany przez zaufane osoby, taki dostęp posiada. Aplikacja wymienia z serwerem komunikaty o określonej strukturze. W przypadku, gdyby ktoś zmienił strukturę komunikatu, po prostu nie zostanie on obsłużony.
- **Środowisko programistyczne** - Aplikacja jest zbudowana w oparciu o języki HTML5, CSS3, JavaScript, Java oraz strukturalny język zapytań (SQL) używany do tworzenia, modyfikowania baz danych oraz umieszczania i pobierania danych z baz danych. Dodatkowo dane przechowywane są w formacie JSON oraz XML.

6. Diagramy dotyczące funkcjonowania oraz komunikacji pomiędzy komponentami aplikacji.



Rysunek 1 Ogólny schemat budowy aplikacji

Powyższy diagram przedstawia sposób funkcjonowania aplikacji wraz z przepływem komunikacji pomiędzy komponentami aplikacji. Dostęp do aplikacji ma użytkownik, który przegląda stronę oraz administrator aplikacji, którego zadaniem jest dodawanie, aktualizacja, usuwanie produktów dostępnych w sprzedaży firmy. Ponadto serwer obsługuje komunikaty wysyłane ze strony internetowej aplikacji i przetwarza je w odpowiedni sposób. Po przetworzeniu serwer wysyła odpowiednie komunikaty do strony. Serwer posiada również dostęp do bazy danych, w której zawarte będą dane do logowania dla administratora strony, dzięki czemu w sposób bezpieczny będzie mógł się logować do panelu zarządzającego.

7. Interfejs graficzny aplikacji.

Strona główna będzie zawierała przyjazny interfejs graficzny na którym będzie znajdowała się lista produktów pobranych z pliku JSON, które są najbardziej popularne.



Rysunek 2 Przykład strony głównej aplikacji internetowej

Na podstronach znajdują się listy produktów w zależności od kategorii wraz ze szczegółami – wartościami odżywczymi, kaloriami, ceną, itd. Wartości te będą pobierane z plików XML.



Rysunek 3 Przykład podstrony na której znajdują się artykuły pobrane z XML

Dodatkowo zostanie zaimplementowany interfejs dla administratora strony, który będzie dodawał, edytował, usuwał produkty. Dane te zostaną zaktualizowane w odpowiednich plikach JSON lub XML.

8. Powiązania z bazą danych.

Baza danych będzie posiadała jedną tabelę „members” w której przechowywane będą loginu, hasła i adresy email użytkownika mającego dostęp do edycji danych na stronie internetowej.

```
1 CREATE TABLE `mydatabase`.`members` (  
2   `id` int( 11 ) NOT NULL AUTO_INCREMENT ,  
3   `username` varchar( 30 ) NOT NULL ,  
4   `email` varchar( 50 ) NOT NULL ,  
5   `password` varchar( 128 ) NOT NULL ,  
6   PRIMARY KEY ( `id` ) ,  
7   UNIQUE KEY `username` ( `username` )  
8 ) ENGINE = MYISAM DEFAULT CHARSET = utf8;
```

Rysunek 4 Kod stworzenia tabeli 'members' w bazie danych

Teraz mamy tabelę, która przechowuje nasze informacje o członku. Należy pamiętać, że do zapisywania haseł używane jest szyfrowanie MD5. Zalecane jest, aby zaszyfrować hasło, w celach bezpieczeństwa, na wypadek kradzieży danych.

```
INSERT INTO `members`(`username`, `email`, `password`) VALUES  
('myuser','myemail@domain.com','mypassword')
```

Rysunek 5 Dodawanie użytkownika mającego dostęp do edycji aplikacji w bazie danych

Aby uzyskać dostęp do bazy danych będziemy używać pośredni skrypt w języku JAVA, tak aby poprawnie połączyć się z naszą bazą danych i odpowiednio dokonywać działań na niej.

9. Walidacja wprowadzanych danych.

Walidacja odbywa się na podstawie wzorca danych, który definiujemy w wyrażeniu regularnym. Wyrażenia regularne pozwalają w szybki sposób walidować nasze dane, jednak każdy wzorzec składa się ze znaków alfanumerycznych oraz meta znaków wyglądających na pierwszy rzut oka dość groźnie, ale nie jest to aż tak straszne.

Meta znaki są używane do tworzenia wyrażeń regularnych, pozwalają na tworzenie zaawansowanych konstrukcji służących do walidacji danych. W przypadku bardziej złożonych zagadnień, ciężko jest znaleźć idealne wyrażenie, które wyłapie wszystkie możliwe warianty, np. problem taki stanowi walidacja adresu e-mail. Poniższe wzorce stanowią przykłady dla kilku popularnych sytuacji.

Tą technologię wykorzystywać będziemy w JavaScript. Do tworzenia wyrażeń regularnych w JavaScript wykorzystamy klasę RegExp.

```
^((?:19|20)\d\d)[- /.](0[1-9]|1[012])[- /.](0[1-9]|12)[0-9]|3[01])$
```

Rysunek 6 Wzorzec walidacji daty

```
^(20|21|22|23|[01]d)d(([:][0-5]d){1,2})$
```

Rysunek 7 Wzorzec walidacji czasu

```
^[A-Z0-9._%+-]+@[A-Z0-9.-]+\.(?:[A-Z]{2}|com|org|net|gov|mil|biz|info|mobi|name|aero|jobs|museum)$
```

Rysunek 8 Wzorzec walidacji adresu e-mail

10. Mechanizmy autentykacji / autoryzacji. Bezpieczeństwo aplikacji.

Uwierzytelnienie będące podstawowym mechanizmem bezpieczeństwa informacji należy rozumieć jako proces potwierdzenia tożsamości użytkownika. Jeśli ma miejsce poprawne uwierzytelnienie, wówczas zachodzi pewność, iż użytkownik jest tym za kogo się podaje. To z kolei daje podstawę do dalszego udostępnienia mu zasobów, które są jego własnością. Jak widać mechanizm uwierzytelnienia stanowi rodzaj bariery, po przejściu której każdy użytkownik jest jednoznacznie rozpoznany przez system.

Potrzeba uwierzytelnienia wzrosła w dobie Internetu z uwagi na rozpowszechnienie pracy zdalnej przy jednoczesnym ograniczeniu możliwości identyfikowania użytkowników fizycznie/osobiście. Wymagane zatem jest aby omawiany mechanizm był starannie zaimplementowany w każdym systemie, a przede wszystkim w aplikacji WWW.

W naszej aplikacji wykorzystamy **pojedynczy model uwierzytelnienia**, po którym użytkownikowi przyznawany jest identyfikator sesji. Do momentu wygaśnięcia sesji, co pociąga za sobą zniszczenie

identyfikatora po stronie serwera, użytkownik jest uwierzytelniony i ma dostęp do swoich zasobów. Czynnikiem uwierzytelnienia jest hasło, które jest wcześniej skonfigurowane po stronie aplikacji manualnie przez administratora lub w procesie automatycznej rejestracji. Hasła zwyczajowo są typu wielokrotnego i dopiero po podaniu ich w parze z prawidłową nazwą użytkownika stanowią dane wejściowe dla mechanizmu uwierzytelnienia.

11. Wymagania i warunki do uruchomienia aplikacji.

Do stworzenia aplikacji administrator potrzebuje serwera internetowego wraz z domeną. Do serwera należy dołączyć najnowsze środowisko JAVA oraz phpmyadmin obsługujące bazę danych. Bardzo często kupując serwer od dostawcy mamy już zagwarantowany dostęp do tych wszystkich narzędzi, również narzędzi analitycznych. Kolejnym krokiem jest przesłanie wszystkich plików na serwer za pomocą połączenia ftp umieszczonych w folderze do przesłania.

12. Opis etapów uruchamiania programu (aplikacji).

Uruchomienie aplikacji odbywa się od momentu wejścia użytkownika na stronę internetową. Wystarczy wpisać adres. Strona ta dostępna jest dla większości przeglądarek internetowych zgodnych ze współczesnymi trendami w projektowaniu stron internetowych np. Google Chrome, Mozilla Firefox czy nowsze wersje Internet Explorer.

Użytkownikowi ukazuje się strona główna na której może przeglądać najpopularniejsze produkty (pobrane z pliku JSON), natomiast na podstronach znajdują się wszystkie dostępne produkty pobrane z plików XML, uporządkowane w zależności od kategorii.

13. Kody źródłowe (reprezentatywne fragmenty).

```
if (request.getParameterMap().containsKey("zapisz")) {  
    System.out.println(request.getParameter("zapisz"));  
    System.out.println(request.getParameter("name"));  
    System.out.println(request.getParameter("age"));  
    System.out.println(Arrays.asList(request.getParameterValues("messages")));  
  
    App newApp = new App(request.getParameter("name"), Integer.parseInt(request.getParameter("age")), Arrays.asList(request.getParameterValues("messages")));  
    resp.getDescriptor().replace(request.getParameter("zapisz"), newApp);  
  
    try (PrintWriter out = response.getWriter()) {  
        try (Writer writer = new FileWriter("C:\\\\Users\\Procislav\\Desktop\\Sem2\\ITAI\\Projekt2\\Projekt2-war\\build\\web\\output.json")) {  
            gson.toJson(resp, writer);  
        }  
        out.println("Zapisano");  
    }  
} else {
```

Rysunek 9 Kod zapisujący dane do pliku JSON

```

fileName = request.getParameter("plik");
System.out.println(fileName);
reader = new JsonReader(new FileReader(fileName));
gson = new Gson();
resp = gson.fromJson(reader, Response.class);

int ile = resp.getDescriptor().size();
Collection<App> kolekcja = resp.getDescriptor().values();
Set<String> klucze = resp.getDescriptor().keySet();
System.out.println("Ile elementow (" + ile + ") kolekcja: " + klucze);
System.out.println("mapa: " + resp.getDescriptor());


showData(resp.getDescriptor(), response.getWriter());

for (Map.Entry<String, App> entry : resp.getDescriptor().entrySet()) {
    App app = entry.getValue();
    System.out.println(entry.getKey() + "/" + app.getName() + ", " + app.getAge() + ", " + app.getMessages());
}

```

Rysunek 10 Odczyt danych z pliku

Arabeska_Wisienka_4_szt



Opis: Sernikowy nus na bazie bitej śmietany z wiśniową konfiturą na waniliowym biszkopiku. Pychał. Wymiary: 6 cm wys 4 cm

Cena: 17.2

Ilość: 4

Czas: 1 dzień

Baza smaków:
sernik - wiśnia - wanilia

Tort_Minecraft



Opis: Granył Tort w wersji podstawowej na 10 os. Rozmiar tortu 16 na 16 cm. Wybierz smak, wielkość oraz bezpłatny napis na tortcie.

Cena: 110.0

Ilość: 10

Czas: 3 dni

Baza smaków:
truskawka - wanilia
sernik - pomarańcza
jagoda - wanilia
zurałina - wanilia
czekolada - wanilia
czekolada - banan
czekolada - cappuccino
tiromisu

Rysunek 11 Wyświetlanie listy dostępnych produktów z pliku JSON


```

if (event=="pull"){
    //metody.showData(lis.getList());
    showTempl(request,response,metody.showData("Bankietówki",lis.getList(),request.isRequestedSessionIdValid()));
}

else if (event.equals("edit") && (request.isRequestedSessionIdValid())){
    showTempl(request,response, metody.formEdit(lis, param,request.isRequestedSessionIdValid()));
} else if (event.equals("del") && (request.isRequestedSessionIdValid())){
    content+="
```

Rysunek 15 Zdarzenia wykonywane w zależności od zdarzenia odebranego przez serwer

14. Odniesienie do implementacji: definicje klas, powiązania pomiędzy klasami.

```

public class Response {
    // @SerializedName("items")
    Map<String, App> descriptor;

    public Map<String, App> getDescriptor() {
        return descriptor;
    }

    public void setDescriptor(Map<String, App> descriptor) {
        this.descriptor = descriptor;
    }
}

```

Rysunek 16 Klasa Response zawierająca w sobie listę zmapowaną produktów, której kluczem są nazwy produktów

```

public class App {
    // @SerializedName("item")
    String name;
    int age;
    List<String> messages;

    public App(String name, int age, List<String> message){
        this.name=name;
        this.age=age;
        this.messages=message;
    }

    public String getName() { ...3 lines }

    public void setName(String name) { ...3 lines }

    public int getAge() { ...3 lines }

    public void setAge(int age) { ...3 lines }

    public List<String> getMessages() { ...3 lines }

    public void setMessages(List<String> messages) { ...3 lines }
}

```

Rysunek 17 Klasa App zawierająca w sobie definicję parametrów określających cechy produktu

15. Wyniki przeprowadzonych testów. Interpretacja wyników.

Zdarzenie	Rezultat
Użytkownik poda zły login lub hasło.	Pojawił się komunikat mówiący o nieprawidłowym logowaniu z możliwością przypomnienia hasła.
Użytkownik nie uzupełnił wszystkich wymaganych pól przy logowaniu	Pojawił się komunikat o konieczności uzupełnienia pól, przy których znajduje się symbol gwiazdki (*).
Użytkownik nie prawidłowo wpisał dane przy dodawaniu produktu	System powiadomił użytkownika o nieprawidłowych danych niezbędnych do dodania produktu
Użytkownik pozostawił wymagane puste pole przy edycji produktu	Pojawił się komunikat o konieczności uzupełnieniu pól, przy których znajduje się symbol gwiazdki (*).

16. Informacje o zastosowaniu w projekcie specyficznego podejścia.

W projekcie, oprócz standardowego przetworzenia danych w formacie JSON przez JAVA, wykorzystanie zostanie zarządzanie tym formatem w jQuery.

Dzięki temu formatowi można z łatwością przysyłać dane między poszczególnymi elementami aplikacji. Przykładowo JSON wykorzystuje się w AJAX'ie jako format do przysyłania danych, o czym napiszę nieco później. JQuery jako niezwykle popularny framework JavaScript, ma wbudowane dwie funkcje do obsługi tego typu struktur danych.

Pierwszą funkcją, jaką możemy operować w JQuery jest `jQuery.parseJSON`. Jako argument przyjmuje ciąg znaków w formacie JSON, który ma zostać przeanalizowany przez funkcję.

Inną funkcją jakiej możemy używać jest `jQuery.getJSON()`. Sposób użycia funkcji `$.getJSON()` jest bardzo prosty. Jako pierwszy argument podajemy nazwę pliku z rozszerzeniem json, natomiast drugi to handler dla funkcji, która zostanie wywołana w przypadku gdy skrypt zakończy się sukcesem.

Dzięki tym funkcjom możemy operować na plikach JSON.

17. Ograniczenia aplikacji (programu). Możliwe rozszerzenia projektu.

System jest na tyle uniwersalny, że jego kierunki rozwoju nie muszą iść wcale w branżę informatyczną. Ważne jest, żeby zachować pierwotny stan aplikacji, z wszystkimi podstawowymi stworzonymi funkcjami. Na pewno najważniejszą możliwością rozwoju byłoby poprawienie komunikacji pomiędzy przetwarzaniem danych z JSON na bardziej uniwersalne, a nie pod konkretne dane jak w tym momencie ma to miejsce.

Aktualnie system nie obsługuje systemu newsletter'a dla klientów aplikacji. Można wprowadzić dodatkowo sklep internetowy, w którym chętne osoby będą mogły przez Internet kupować produkty.

18. Podsumowanie i wnioski

Jako podstawowe cele projektowe, które wcześniej założono zostały wykonane. Produkty zapisane w formacie JSON są odczytywane na stronie głównej, a po zalogowaniu administrator ma możliwość ich edycji i kasowania. Analogicznie wygląda sytuacja z produktami zapisanymi w formacie XML, które znajdują się na podstronach strony internetowej – cukierni. Jeśli administrator będzie wprowadzał nie poprawne dane do formularza podczas wprowadzania lub edycji produktu, pojawi mu się komunikat, że dane są niepoprawne z prośbą o ich poprawienie. Podczas usuwania produktu najpierw pojawia się komunikat w JavaScript, upewniający, czy na pewno chcemy usunąć produkt. Jeśli wciśniemy OK – zostanie on usunięty. W przeciwnym wypadku powrócimy do listy produktów.

Strona ma bardzo prosty i przyjazny interfejs, o uspokajającej zielonej barwie. Przeprowadzone badania wpływu kolorów na ludzką psychikę wykazują, że zieleń ma kojący wpływ na układ nerwowy, działa relaksacyjnie i odświeża. Na zielono maluje się wnętrza pomieszczeń, aby zmniejszyć stres związany z chorobą czy pracą.

W Javie istnieje kilka możliwości przetwarzania tego formatu. JEE (ang. Java Enterprise Edition) udostępnia standardowe API, jednakże odpowiedzialność za konwersję spada na nas. Podobne możliwości oferuje biblioteka json-simple. Jednak pomoc w serializacji i deserializacji obiektów pomaga biblioteka Google Gson. Jest to elastyczna biblioteka, pozwalająca na dostosowanie działania. Domyślnie obsługuje typy proste, generyczne, kolekcje oraz klasy zagnieżdżone. Google Gson używa mechanizmu refleksji, tak więc nie potrzebuje dodatkowych informacji do działania. Klasy muszą mieć jedynie zdefiniowany bezargumentowy konstruktor. Jednym z wartych uwagi mechanizmów jest to, iż biblioteka pozwala na pisanie własnych klas serializujących i deserializujących. Za pomocą adnotacji @SerializedName można ustawić jaką nazwa atrybutu zostanie przypisana danemu polu. Istnieją także inne możliwości: obsługa pól z wartością null, wykluczanie pól z serializacji, manipulacja drzewem wyrażenia.

Natomiast do mapowania informacji zawartych w plikach XML służy biblioteka JAXB. Pozwala ona na wykonanie mapowania klas Java do ich reprezentacji w formacie XML. Biblioteka dostarcza takie funkcje jak marshalling (konwersja obiektu Java do formatu XML) i unmarshalling (konwersja zawartości XML do obiektu Java). Wykorzystanie JAXB nie wymaga dołączania do projektu żadnych dodatkowych bibliotek jeżeli jest wykorzystywane JDK od wersji 6, ponieważ JAXB jest już w nim wbudowane. Klasy które chcemy wykorzystać do mapowania z XML muszą zawierać odpowiednie adnotacje JAXB.

19. Literatura.

- Jon Duckett „HTML i CSS. Zaprojektuj i zbuduj witrynę WWW”
- Jon Duckett „JavaScript i jQuery. Interaktywne strony WWW dla każdego”
- Dawid Sawyer McFarland „CSS nieoficjalny podręcznik”
- Matthew MacDonald „HTML5. Nieoficjalny podręcznik ”
- Danowski Bartosz „Wstęp do HTML5 i CSS3”
- Robert C. Martin „Czysty kod Podręcznik Dobrego Programisty”
- Cay S. Horstmann, Gary Cornell „Java: Techniki zaawansowane”
- Artykuł redakcji ZeroCool, „Google Gson”, <http://www.zerocool.pl/google-gson/>
- David Geary, Cay S. Horstmann „JavaServer Faces. Wydanie III”