Uniwersytet w Białymstoku

Wydział Matematyczno-Fizyczny

Instytut Matematyki

Dorota Borowska

Interfejs WWW do bazy danych informacji o poprawkach systemu Solaris

Praca dyplomowa napisana
pod kierunkiem
dr. Mariusza Żynela

Białystok 2005

Składam serdeczne podziękowania
dr. Mariuszowi Żynelowi
za pomoc w przygotowaniu i realizacji projektu,
za cenne rady i wskazówki podczas pisania pracy

Dorota Borowska

Spis treści

Wstęp

W dobie dzisiejszej komputeryzacji jest bardzo ostra konkurencja na rynku oprogramowania. Producenci wypuszczają do użytku bardzo dużo nowych programów, ale nie koniecznie są one sprawdzone i nie zawsze działają bez zarzutów. W przypadku małej aplikacji szybko można zaaktualizować ją do nowszej wersji. Gorzej jest ze złożonymi aplikacjami i systemami operacyjnymi, których przeinstalowanie zabiera dużo więcej czasu, a jeszcze więcej może zająć ich dostosowanie do naszych potrzeb. Na potrzeby takiego oprogramowania tworzone są mniej lub bardziej złożone systemy łatek (ang. patch).

Razem z panem Andrzejem Drobnikowskim zajeliśmy się problemem nanoszenia poprawek w systemie Solaris. W ramach swojej pracy pan Andrzej projektuje bazę danych i tworzy zestaw skryptów, umożliwiających zarządzanie poprawkami. Celem mojej pracy jest opracowanie interfejsu WWW w PHP do tej bazy danych. Wybór PHP jako środowiska programistycznego jest podyktowany jego dużą popularnością. Z tego samego powodu wybraliśmy MySQL jako bazę danych oraz Apache jako serwer HTTP. Wszystkie te narzędzia dostępne są za darmo wraz z kodem źródłowym.

Nasza aplikacja ma umożliwiać zdalne importowanie i aktualizację bazy na podstawie danych z poszczególnych komputerów i od producenta sytemu operacyjnego, filtrowanie informacji z bazy i tworzenie raportów oraz przygotowywanie pakietów poprawek gotowych do instalacji. Bardzo ważnym założeniem co do naszej aplikacji jest możliwość korzystania z bazy przez wielu niezależnych użytkowników. W związku z tym moim zadaniem było opracowanie zasad autentykacji i autoryzacji przy dostępie do naszej aplikacji. Jednym z podstawowych założeń jest także przenoszalność naszej aplikacji i możliwość jej instalacji na wielu różnych komputerach.

Witryna jest wykonana w całości w języku angielskim tak aby była dostępna dla użytkowników z całego świata. Z drugiej strony zależy nam na współpracy przy projekcie z innymi osobami, dlatego też wszystkie komentarze, nazwy zmiennych, funkcji i plików są angielskie.

Pomysł projektu nie jest nowy. Wykorzystujemy efekty pracy Bruce'a Riddle - administratora w dużej korporacji amerykańskiej zajmującej się projektowaniem układów scalonych, który odpowiedzialny jest za około 500 komputerów pracujących pod kontrolą Solaris. Wyniki naszej pracy można obejrzeć na `http://patchdb.solaris-x86.net`. Serwis uruchomiony jest na serwerze w Instytucie Matemtayki UwB.

Rozdział 1

Narzędzia i technologie wykorzystane w aplikacji

1.1 Interfejs MySQL w PHP

PHP jest niezwykle przydatnym narzędziem w przyborniku webmastera. Należy do technik określanych jako server-side (skrypty wykonywane na serwerze) i pozwala na szybkie tworzenie aplikacji internetowych wykorzystujących bazy danych i generujących dynamiczne strony WWW. Jedną z najważniejszych zalet PHP jest możliwość współpracy z wieloma bazami danych m.in. Sybase, PostgreSQL, Oracle, MySQL oraz innych.

1.1.1 PHP

Krótka historia

PHP utworzony został w 1994 przez Rasmusa Lerdorfa. Użył on tego języka na swojej stronie domowej. Zadaniem tej pierwszej, niepublikowanej wersji PHP było zbieranie informacji o osobach odwiedzających witrynę, a spośród ciekawych funkcji należy wymienić możliwość umieszczania pytań SQL w stronach WWW. Już w następnym roku PHP (Personal Home Page Tools) zyskało sobie dość dużą popularność i zostało udostępnione użytkownikom Sieci. Nie miało ono jeszcze wtedy zbyt dużych możliwości. Obsługiwało proste instrukcje, pozwalało realizować popularne usługi (licznik, księga gości, itp) wykorzystywane na stronach WWW. Wkrótce pojawiły się jednak liczne sugestie dotyczące rozszerzenia możliwości parsera PHP.

W 1995 roku narodziło się PHP/FI (PHP2), uzupełnione o pakiet interpretujący dane z formularzy HTML (FI = Form Interpreter) oraz możliwość obsługi baz danych mSQL. Kiedy ludzie zaczęli dodawać swój kod, nastąpił niewiarygodny rozwój nowej wersji PHP. Technologii tej używano na coraz większej liczbie witryn WWW.

Kolejna zmiana nastąpiła dwa lata później. PHP przestało być jedynie

osobistym projektem Rasmusa i garstki zapaleńców, stając się dziełem zespołowym. Zeev Suraski i Andi Gutmans, pisząc część kodu zupełnie od nowa, stworzyli nowy parser, będący podstawą PHP3.

Najnowsza wersja (PHP4) korzysta z potężnej platformy skryptowej Zend (`http://www.zend.com`), co zwiększa jej wydajność oraz może działać jako moduł serwera innego niż Apache. Obecnie PHP dostarczane jest z wieloma komercyjnymi produktami, a z technologii tej korzysta już ponad 5 milionów stron na całym świecie.

Kod źródłowy PHP jest powszechnie dostępny na zasadzie Open Source (`http://www.opensource.com`).

Czym jest PHP?

PHP jest językiem skryptowym działającym po stronie serwera. Skrypt PHP można umieścić w obrębie strony HTML, zostanie on wykonany ilekroć strona jest odwiedzana. Fragment dokumentu, który ma zostać zinterpretowany jako skrypt PHP zaznacza się w ten oto sposób:

```
<?php  (treść skryptu) ?>
```

Słówko "php" można opuścić.

W PHP są tylko dwa typy danych: skalary i listy. Skalar to dowolny napis. Na skalarach reprezentujących liczby całkowite bądź rzeczywiste można wykonywać operacje arytmetyczne. Skalary takie zachowują się jak liczby bądź napisy w zależności od kontekstu. Listy możemy indeksować liczbami całokowitymi, tak jak w wielu innych językach programowania, możemy też indeksować napisami, wówczas mówimy o listach asocjacyjnych.

Nazwy zmiennych w PHP poprzedzone są znakiem `$`. Przypisanie wartości zmiennej może wyglądać jak poniżej:

```
$zmienna = "Jakiś tekst";
$tablica = array("R" => "Red", "G" => "Green", "B" => "Blue");
```

Jak widać każda instrukcja w PHP kończy się znakiem `;`. W drugim wierszu mamy przykład inicjalizacji listy asocjacyjnej. Na tej liście znajdują się trzy elementy, do których odwołujemy się pisząc: `$tablica['R']`, `$tablica['G']` lub `$tablica['B']`.

PHP posiada bardzo rozbudowaną bibliotekę funkcji. Ilość dostępnych funkcji uzależniona jest od wersji PHP i od tego, jakie moduły zostały skompilowane. Można też tworzyć własne funkcje, tak jak poniżej:

```
function f($a, $b, $c) {
    return $b*$b-4*$a*$c;
}
```

Powyższa funkcja wylicza deltę dla trójmianu kwadratowego.

Język PHP byłby bezużyteczny bez instrukcji warunkowych i pętli. Instrukcją warunkową w PHP jest instrukcja `if/elseif/else`. Jeśli chodzi o pętle mamy tutaj dostępne cztery typy: `while`, `do..while`, `for`, `foreach`. Trzy pierwsze z nich to klasyczne pętle, występujące w niemal wszystkich językach programowania, natomiast `foreach` jest zapożyczona z Perla i umożliwia łatwą iterację wewnątrz tablic. O PHP można byłoby napisać dużo więcej, ale nie ma to zbyt wielkiego sensu dlatego, że niniejsza praca nie jest podręcznikiem PHP, z drugiej strony w książkach (por. [1]) i w Internecie (por. [3]) dostępnych jest wiele mateiałów i publikacji na temat PHP.

1.1.2 MySQL

MySQL jest relacyjną bazą danych SQL (Structured Query Language), dostępną publicznie od 1996 roku, ale historia jej tworzenia sięga roku 1979. Serwer bazodanowy MySQL wyprodukowany został przez szwedzką firmę T.c.X DattaKonsultAB. Pochodzenie nazwy MySQL nie jest do końca jasne. Autorzy, którzy pracowali nad MySQL, używali nazwy, bądź przedrostka "my" do wielu bibliotek, narzędzi i katalogu głównego. My to także imię córki jednego z twórców MySQL.

MySQL jest chyba jedną z najpopularniejszych baz danych języka SQL. Charakteryzuje się ogromną elastycznością i jeszcze większą prędkością. Jest to bardzo solidny system zarządzania relacyjnymi bazami danych. Serwer MySQL kontroluje dostęp do danych w celu zapewnienia równoczesnego dostępu wielu użytkownikom oraz dostępu jedynie dla uwierzytelnionych użytkowników. Oznacza to, że MySQL jest serwerem wielodostępnym i wielowątkowym.

Prawdopodobnie MySQL zawdzięcza swoją popularność dzięki interfejsom do wielu różnych języków programowania. Podstawowy interfejs w języku C posłużył do stworzenia interfejsu do Perla, PHP, Meta-HTML oraz wielu innych wyspecjalizowanych języków skryptowych. Niebagatelny wpływ na popularnośc MySQL ma również prostota i niezawodność jego interfejsu.

Z uwagi na to, że jest wiele publikacji poświeconych MySQL (por. [2]), nie będziemy w naszej pracy opisywać ani API[1] ani też języka SQL. Ograniczymy się tutaj do przedstwienia interfejsu do PHP.

1.1.3 Współpraca z bazą danych

Jedną z najważniejszych cech nowoczesnych języków programowania lub narzędzi programistycznych jest zdolność współpracy z bazą danych. Jest to spowodowane tym, że systemy zarządzania relacyjnymi bazami danych posiadają wiele bardzo wydajnych i niezwykle użytecznych mechanizmów zarządza-

[1] Aplication Programming Interface – interfejs programowy aplikacji.

nia danymi, jak na przykład indeksowanie, relacje pomiędzy danymi, obsługa transakcji, kaskadowe operacje wykonywane na danych i wiele innych. PHP pozwala na dostęp do danych przy użyciu bogatego zestawu funkcji związanych z wieloma różnymi bazami danych. Funkcje dotyczące bazy MySQL mają prefiks `mysql_`.

Pracę z bazą danych rozpoczynamy od nawiązania z nią połączenia. Odpowiedzialne są za to dwie funkcje:

```
mysql_connect(serwer, użytkownik, hasło)
mysql_pconnect(serwer, użytkownik, hasło)
```

Obie funkcje jako wynik zwracają identyfikator połączenia. Różnica polega na tym, że `mysql_pconnect()` zanim otworzy nowe połączenie do bazy, spróbuje użyć wcześniej otwartego połączenia. Przyśpiesza to działanie skryptu gdyż otwarcie nowego połączenia jest dość kosztowne. To rozwiązanie ma jednak wadę, ponieważ utrudnia korzystanie z tabel tymczasowych (ang. temporary table). Tabele takie są przechowywane w pamięci i powinny być usuwane w momencie rozłączenia z serwerem bazy danych. W sytuacji gdy dwóch różnych użytkowników uruchamia ten sam skrytp PHP i wykorzystywane jest jedno połączenie do bazy danych, nastąpi konflikt przy tabelach tymczasowych.

Obie funkcje pobierają takie same argumenty: nazwę komputera, na którym uruchomiony jest serwer MySQL, identyfikator użytkownika oraz jego hasło. Poniżej przedstawiamy przykład użycia `mysql_connect()`

```
mysql_connect("localhost", "test", "abc")
  or die ("Nie można się połączyć");
print ("Połączenie nawiązane");
```

Następnym krokiem jest wybór właściwej bazy danych. Funkcją PHP realizującą to zadanie jest `mysql_select_db()`, jej prototyp przedstawia się następująco:

```
mysql_select_db(nazwa_bazy [, identyfikator_połączenia])
```

Identyfikator połączenia jest opcjonalny. Jeśli się go nie poda, zostanie użyty identyfikator ostatnio otwieranego połączenia.

W odróżnieniu od Perla i Meta-HTML w PHP wszystkie zapytania do bazy danych kieruje się za pomocą jednej funkcji:

```
mysql_query(zapytanie [, identyfikator_połączenia])
```

niezależnie od tego czy dane zapytanie zwraca jakieś wartości czy też nie. Argumentem tej funkcji jest tekst zapytania, które ma zostać wykonane. Podobnie jak przy `mysql_select_db()` identyfikator połączenia jest opcjonalny. Jeżeli zapytanie jest typu "select", to funkcja `mysql_query()` zwraca identyfikator wyniku. Dla pozostałych zapytań SQL `mysql_query()` zwraca TRUE lub FALSE w zależności od tego czy dane zapytanie zakończyło się sukcesem czy nie. Poniżej przedstawiamy przykład użycia opisywanej funkcji:

```
mysql_query( "SELECT * FROM tabela")
  or die ("Zapytanie zakończone niepowodzeniem");
```

Do odczytywania odpowiedzi na nasze zapytania w PHP mamy trzy funkcje:

```
mysql_fetch_row(wynik)
mysql_fetch_array(wynik)
mysql_fetch_assoc(wynik)
```

Są one podobne do siebie i zwracają jeden wiersz wyniku w postaci tablicy. Różnica polega na sposobie indeksowania tych tablic. Tablica zwrócona przez `mysql_fetch_row()` jest indeksowana liczbami naturalnymi (od 0, tak jak wszystkie tablice w PHP). Tablica uzyskana przez `mysql_fetch_array()` indeksowana może być liczbami naturalnymi, jak poprzednia, może być również indeksowana nazwami kolumn z bazy danych. Funkcja `mysql_fetch_assoc()` zwraca tablicę asocjacyjną, którą indeksujemy nazwami kolumn z bazy danych.

Wywołanie jednej z tych funkcji zwraca kolejny wiersz danych uzyskanych w wyniku zapytania i przesuwa wewnętrzny wskaźnik do następnego wiersza. Jeżeli nie ma więcej danych, funkcja zwraca FALSE. Wywołanie funkcji `mysql_fetch_array()` nie jest zauważalnie wolniejsze niż wywołanie `mysql_fetch_row()`, a daje możliwość indeksowania na dwa sposoby. Zazwyczaj funkcje "fetch" woła się w pętli while.

Po zakończeniu odczytywania wyniku zapytania należy zwolnić zajmowaną pamięć. Do tego celu służy funkcja:

```
mysql_free_result(wynik)
```

Zwalnia ona całą pamięć przydzieloną podanemu wskaźnikowi wyniku. Wołanie tej funkcji nie jest obowiązkowe, ale w skryptach, w których mamy wiele zapytań może ona przyśpieszyć wykonanie skrytpu.

Jeśli kończymy pracę w bazie danych, dobrym zwyczajem jest rozłączenie się. Robi się to wołając funkcję:

```
mysql_close([identyfikator_połączenia])
```

Połączenia niestałe, otwarte za pomocą `mysql_connect()`, zamykane są automatycznie po skończeniu przetwarzania skryptu PHP.

Poniżej przedstawiamy prosty, ale kompletny przykład komunikacji z bazą MySQL z poziomu PHP.

```
<?php

/* Nawiązanie połączenia z bazą */
$polaczenie = mysql_connect('host', 'uzytkownik', 'haslo');
```

```
/* Wybranie odpowiedniej bazy danych */
mysql_select_db('bazadanych', $polaczenie);

$sql = "SELECT imie, nazwisko FROM dane_pracownika";

/* Zapytanie sql do bazy i zapamiętanie wyniku */
$wynik = mysql_query($sql);

/* Pętla dopóki istnieją dane */
while ($wiersz = mysql_fetch_array($wynik)) {

    echo '$wiersz['imie'] $wiersz['nazwisko']';
}

/* Rozłączenie z bazą danych */
mysql_close($polaczenie);

?>
```

Powyższy skrypt odczytuje dwie kolumny: `imie` i `nazwisko` z tabeli `dane_pracownika` i wypisuje je.

Interfejs MySQL w PHP posiada dodatkowo kilka funkcji, które nie są dostępne w pozostałych interfejsach SQL. Dostarczone są specjalizowane funkcje do tworzenia i usuwania baz danych oraz funkcje umożliwiające odczytanie struktury bazy danych. Na przykład za pomocą funkcji `mysql_list_tables()` można uzyskać listę wszystkich tabel w bazie danych. Dokładny opis wszystkich funkcji z interfejsu MySQL znaleźć można w [3]. Poniżej zaprezentujemy jeszcze kilka bardziej przydatnych funkcji:

`mysql_num_rows(wynik)` — zwraca ilość wierszy w odpowiedzi na zapytanie,

`mysql_num_fields(wynik)` — zwraca ilość pól w odpowiedzi na zapytanie,

`mysql_list_dbs([identyfikator_połączenia])` — zwraca wszystkie nazwy baz znajdujących się na serwerze,

`mysql_list_tables(nazwa_bazy, [identyfikator_połączenia])` — pobiera listę tabel z bazy MySQL,

`mysql_result(wynik, numer_wiersza, [pole])` — zwraca zawartość pojedyńczej komórki z tabeli odpowiedzi,

`mysql_error([identyfikator_połączenia])` — zwraca tekst komunikatu błędu z ostatnio użytej funkcji MySQL lub pusty ciąg znaków, jeśli błąd nie wystąpił.

1.2 Kontrola dostępu do aplikacji

Zakładamy, że z tworzonej przez nas aplikacji korzystać będzie mogło wielu użytkowników jednocześnie. Każdy z tych użytkowników będzie przechowywał w naszej bazie danych informacje o tym, jakie patche są zainstalowane na jego komputerach. Nie są to być może tajne dane, ale powinniśmy zadbać o to, by użytkownicy mieli dostęp wyłącznie do swoich danych. Chodzi o to, aby uniknąć przypadkowego bądź celowego zniszczenia informacji o komputerze jednego z użytkowników przez kogoś innego. Z drugiej strony haker wiedząc, jakie patche są zainstalowane na danym komputerze, wie również jakich patchy brakuje i może wykorzystać to do włamania się. Z tego powodu zdecydowaliśmy, że dostęp do naszej aplikacji będzie ograniczony. Aby z niej korzystać, użytkownik zobowiązany jest zarejestrować się.

Przeanalizujmy jakie narzędzia i techniki są dostępne do tego aby kontrolować dostęp w naszej aplikacji. Poszukiwać ich będziemy w PHP oraz w serwerze HTTP (w naszym wypadku tym serwerem jest Apache 1.3).

Najczęściej stosowane metody autentykacji, czyli weryfikacji użytkownika czy jest tym za kogo się podaje, w aplikacjach WWW są oparte o jeden z dwóch mechanizmów: cookies lub autoryzacja HTTP.

Cookies

Jest to mechanizm służący do przechowywania danych w przeglądarce. W ten sposób można śledzić lub identyfikować powracających użytkowników. Zazwyczaj "ciasteczka" są wykorzystywane do automatycznego rozpoznawania danego użytkownika przez serwer, dzięki czemu serwer może wygenerować stronę ściśle dedykowaną danemu użytkownikowi.

Zasada działania tego mechanizmu jest bardzo prosta. Z poziomu PHP lub innego języka skryptowego, działającego po stronie serwera, wysyłane jest żądanie SET_COOKIE (jako nagłówek HTTP). Odbierając to polecenie przeglądarka zobowiązana jest zapamiętać dane wchodzące w skład nagłówka SET_COOKIE. W nagłówku tym podaje się zmienną (identyfikator) z przypisaną wartością i określa się jak długo dana zmienna jest ważna, ścieżkę do dokumetnów na serwerze, z którą jest związana oraz domenę serwera. Gdy przeglądarka przyjmie "ciasteczko" wówczas przy każdym odwołaniu do dokumentu jeśli ścieżka do tego dokumentui i domena serwera pasują do tych określonych w SET_COOKIE, przeglądarka wysyła nagłówek HTTP o nazwie COOKIE zawierający zmienną i jej wartość. W ten sposób możemy po stronie serwera wygenerować unikalny identyfikator i zapamiętać go w przeglądarce odwiedzającej nas osoby. Przy następnych odwiedzinach naszej strony przez tę osobę otrzymamy jej identyfikator i będziemy mogli podjąć akcje związane z konkretnym użytkownikiem. Taka akcja może polegać na wypisaniu na stronie powitania z imieniem i nazwiskiem, ale może także służyć do tego by wykonać przelew bankowy.

Na ogół autentykacja z wykorzystaniem "ciasteczek" odbywa się w ten sposób, że za pomocą formularza w HTML prosimy użytkownika o podanie nazwy i hasła, sprawdzamy je w bazie danych na serwerze i jeśli logowanie było pomyślne to w przeglądarce użytkownika ustawiamy zmienną z numerem sesji tego użytkownika. Przy kolejnych odwołaniach do serwisu nie ma potrzeby sprawdzania hasła, wystarczy sprawdzić numer sesji. Operacja sprawdzania numeru sesji jest przezroczysta z punku widzenia użytkownika i odbywa się w pełni automatycznie.

W PHP SET_COOKIE wysyłamy używając funkcji `setcookie()`:

```
setcookie(nazwa_zmiennej, wartość,
  data_ważności, ścieżka, domena, bezpieczne)
```

"Ciasteczkami" możemy też zarządzać odwołując się bezpośrednio do przeglądarki za pomocą skryptów JavaScript osadzanych w dokumentach HTML.

Autoryzacja HTTP

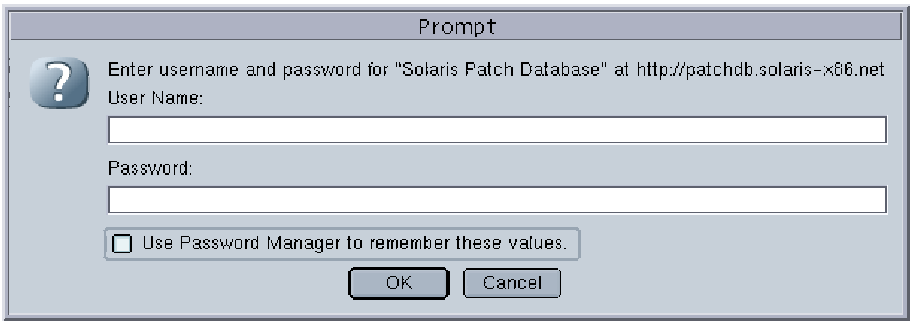
Autoryzacja HTTP odbywa się całkowicie na poziomie serwera. Aby ograniczyć dostęp do plików i podkatalogów w określonym katalogu, należy odpowiednio skonfigurować serwer HTTP. Brzmi to dość poważnie, ale w przypadku serwera Apache jest dosyć proste. Wystarczy w danym katalogu umieścić plik o nazwie `.htaccess` o mniej więcej takiej zawartości:

```
AuthMySQLDB patchdb
AuthMySQLUserTable users
AuthMySQLNameField email
AuthMySQLPasswordField password
AuthMySQLPwEncryption crypt
AuthMySQLUserCondition "active = 'Y'"
AuthName "Solaris Patch Database"
AuthType Basic
require valid-user
satisfy all
```

W pliku `.htaccess` jak widać podajemy skąd serwer HTTP ma pobrać dane o hasłach użytkowników. W powyższym przykładzie hasła pobierane są z bazy MySQL o nazwie `patchdb` z tabeli `users`. Identyfikator do logowania pobierany jest z kolumny `email`, a hasło z kolumny `password`. Metoda szyfrowania hasła to `crypt`, dodatkowo przy weryfikacji hasła spradzany jest dodatkowy warunek czy konto jest aktywne. Dane niezbędne do autentykacji nie muszą być pobierane z bazy MySQL. Hasła można umieścić w zwykłym pliku tekstowym.

Przy próbie odczytu dokumentu z chronionego katalogu serwer poprosi przeglądarkę o podanie nazwy użytkownika i hasła. Przeglądarka powinna

wyświetlić okno dialogowe (por. rys. 1.1), w którym użytkownik wpisuje swój login i hasło.



Rysunek 1.1: Okno logowania.

Jeśli zalogowanie powiedzie się, przy każdym odwołaniu do serwisu przeglądarka wysyła do serwera nagłówek HTTP o nazwie `Authorization`, np.:

```
Authorization: Basic username:passwd
```

gdzie `username` i `passwd` są zakodowane.

Autoryzacja HTTP jest wygodna dla autora aplikacji, gdyż mechanizm ten jest przezroczysty z puntku widzenia aplikacji. Utrudnieniem może być odpowiednie skonfigurowanie serwera HTTP.

O ile w przypadku cookies autentykacja musi być wykonana przez aplikację, to autentykacja HTTP odbywa się automatycznie na poziomie serwera HTTP. W ten sposób autor aplikacji nie musi pisać skryptu, który sprawdzałby hasła użytkowników. Na poziomie serwera sprawdzane jest również czy dany użytkownik ma prawo dostępu do żądanego zasobu. Oznacza to, że serwer wykonuje coś więcej niż tylko autentykację, polegającą na sprawdzeniu hasła, ale również autoryzuje dostęp do zasobów. Inaczej jest gdy opieramy autoryzację o "ciasteczka". Wówczas w każdym z dokumentów, do których dostęp jest ograniczony, sami musimy wykonać odpowiednie sprawdzenia.

Wydaje się, że autoryzacja HTTP jest lepsza pod każdym względem od autoryzacji opartej o "ciasteczka". Z pewnością jest ona dużo bardziej niezawodna, bo działa na niższym poziomie, ale przez to nie wiemy np. kiedy użytkownik się loguje i wylogowuje z serwisu.

Przy wyborze metody autoryzacji pozostaje jeszcze kwestia szyfrowania haseł. Dobrym zwyczajem jest przechowywanie haseł w postaci zaszyfrowanej. Uniemożliwia to wprawdzie przekazanie użytkownikowi hasła gdy go zapomni. Możemy przecież jednak wygenerować losowo nowe hasło, przekazać je użytkownikowi i zapisać je w systemie w postaci zaszyfrowanej. Szyfrujac hasła administrator unika też podejrzeń o podglądanie haseł.

W Unixie standardową funkcją szyfrującą jest `crypt()`. Funkcja ta jest nieodwracalna, nie ma więc możliwości odszyfrowania hasła. W naszej aplikacji zdecydowaliśmy się na tę funkcję, gdyż dostępna w MySQL funkcja `password()` jest niekompatybilna między starszymi wersjami MySQL, co uniemożliwiałoby przenoszenie danych między różnymi wersjami bazy.

Rozdział 2

Podręcznik użytkownika

Z naszej aplikacji może korzystać wiele osób w tym samym czasie. Powoduje to, że dostęp do aplikacji musi być kontrolowany. Mamy zatem trzy poziomy dostępu. Do pierwszego mają dostęp wszyscy bez ograniczeń, aby dostać się do drugiego należy się zarejestrować, natomiast trzeci poziom przewidziany jest dla administratora aplikacji, który ją zarządza.

Aplikacja, którą napisaliśmy jest aplikacją WWW i korzystamy z niej za pomocą dowolnej przeglądarki internetowej.

2.1 Ogólnodostępna część serwisu

2.1.1 Strona główna

Pracę z aplikacją zaczynamy od strony głównej. Umieściliśmy na niej najważniejsze z punktu widzenia użytkownika informacje. Na pierwszy plan wysuwa się tabela zawierająca ostatnio wydane patche do sytemu Solaris (rys. 2.1).

No.	Patch ID	Revision	Release date	Rec.	Sec.	Synopsis
1.	112807	16	2005-06-15	•	•	CDE 1.5: dtlogin patch
2.	114210	15	2005-06-15	•	•	CDE 1.5_x86: dtlogin patch
3.	117462	01	2005-06-15			SunOS 5.10_x86: patch boot/solaris/boot.bin
4.	118387	05	2005-06-15		•	Sun Management Centre 3.5.1: Patch for Solaris 7
5.	119075	06	2005-06-15			SunOS 5.10: ip Patch
6.	119076	05	2005-06-15			SunOS 5.10_x86: ip Patch
7.	119263	03	2005-06-15			SunMC 3.5pu1 libcfgpicIWG.so.1.0 patch

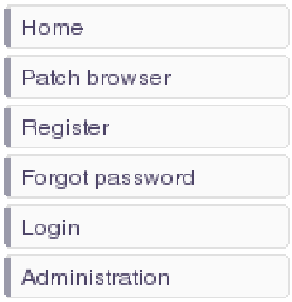
Rysunek 2.1: Tabela ostatnio wydanych patchy do Solaris.

W tej tabeli mamy listę patchy wydanych w ciągu ostatnich trzech dni od wydania najnowszego patcha. Wypisane są wszystkie patche bez względu

na wersję systemu i przeznaczenie. Tabela ta pozwala szybko zorientować się, czy wyszły interesujące nas poprawki. Mamy możliwość kliknięcia na każdy z patchy i obejrzenia szczegółowych informacji na jego temat. Strona opisu patcha zostanie opisana później w 2.1.3.

Na stronie głównej znajduje się również odnośnik umożliwiający ściągnięcie najnowszej wersji pilku `patchdiag.xref` ze strony SunSolve. Ważnym elementem strony głównej jest też wyszukiwarka patchy, połączona ze stroną SunSolve. Za jej pomocą możemy szybko przeszukiwać bazę danych poprawek u samego producenta. Mamy tutaj też odnośniki do strony zawierającej klastry rekomendowanych patchy, jak również do stron zawierających patche związane z bezpieczeństwem systemu.

W lewym górnym rogu okna umieszczone jest menu główne, z którego szybko możemy przejść do ważnych części aplikacji (rys. 2.2).



Rysunek 2.2: Menu główne.

Poszczególne pozycje w menu mają następujące znaczenie:

Home – przejście do strony domowej (głównej),

Patch browser – przeglądarka poprawek,

Register – po kliknięciu, pojawi się formularz do rejestracji,

Forgot password – ten odnośnik jest dla użytkowników, którzy zapomnieli swojego hasła,

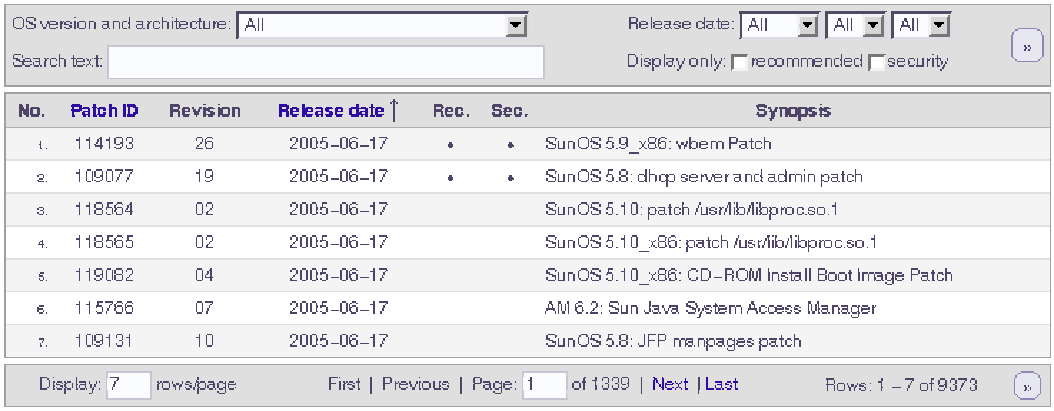
Login – w tym miejscu logują się zarejestrowani użytkownicy,

Administration – jaka sama nazwa mówi, jest to przejście do zarządzania aplikacją.

Na górze każdej strony serwisu mamy wyszukiwarkę, przy pomocy której możemy szybko odszukać patche w bazie danych aplikacji. Przeszukiwane są identyfiaktory, jak również tekst opisu. Możemy zatem znaleść poprawkę znając jej numer lub wszystkie takie, które zawierają podany ciąg znaków w swoim opisie.

2.1.2 Przeglądarka patchy

W ogólnodostępnej części można znleźć wiele cennych informacji o patchach. Każdy może zobaczyć listę ostatnio dodanych patchy, czy wyszukać informacje na temat jakiegoś patcha. W celu usprawnienia przeszukiwania bazy poprawek została przygotowana specjalna przeglądarka (patch browser), której zadaniem jest umożliwienie jak najszybszego dotarcia do interesujących patchy (rys. 2.3).



Rysunek 2.3: Przeglądarka patchy.

Przeglądarka ta ma wiele opcji wyszukiwania i sortowania. Wybierając z listy wersję systemu Solaris oraz platformę (OS version and architecture) możemy szybko odfiltrować odpowiednie patche. Możemy też określić rok, miesiąc i/lub dzień wydania poprawki. W przeglądarce przewidziana jest możliwość wpisania dowolnego tekstu, który będzie wyszukiwany w opisach poprawek. Dostępne są również dwa filtry, za pomocą których można szybko wybrać poprawki rekomendowane oraz związane z bezpieczeństwem systemu.

Sortowanie poprawek wykonywane jest według identyfikatora (patch ID) lub według daty wydania (release date). Aby przesortować tabelę, wystarczy kliknąć odpowiedni nagłówek. Strzałka umieszczona obok nazwy nagłówka pokazuje kierunek sortowania. Kierunek ten możemy zmienić, klikając nagłówek jeszcze raz.

Sami możemy zdecydować, ile wierszy ma być wyświetlonych na stronie. Numer wyświetlanej strony podany jest w środkowej części dolnej listwy. Obok umieszczono przyciski, które pozwalają przełączać się między stronami. Z prawej strony dolnej listwy podana jest informacja, które wiersze są aktualnie wyświetlane na stronie.

Kliknięcie na wiersz tabeli powoduje przejście do szczegółowego opisu patcha.

2.1.3 Szczegółowy opis patcha

Na tej stronie możemy obejrzeć wszystkie informacje na temat danej poprawki, zgromadzone w bazie naszej aplikacji:

- identyfikator poprawki,
- numer wersji,
- data wydania,
- skrócony opis,
- wersja systemu i platforma,
- czy poprawka jest rekomendowana,
- czy poprawka dotyczy bezpieczeństwa systemu.

Poniżej znajduje się odnośnik do odpowiedniej strony na SunSolve zawierającej dokładny opis poprawki. W naszej aplikacji dostępne są tylko najważniejsze dane, nie ma np. informacji, dokładnie jakie błędy dana poprawka usuwa. Te właśnie informacje i inne są dostępne na SunSolve.

Na stronie opisu poprawki możemy znaleźć również listę patchy wymaganych przez tę poprawkę (tzw. prerekwizytów), listę patchy zależnych od danej poprawki (tzn. tych, dla których jest prerekwizytem) oraz listę patchy konfliktowych (tzn. tych, które nie mogą być instalowane razem z naszą poprawką). Ponadto jest tutaj także lista pakietów, których nasza poprawka dotyczy. Możemy zatem zorientować się, czy dana poprawka jest dla nas potrzebna. Na końcu strony umieszczona jest tabela zawierająca historię danej poprawki. Umieszczone są tutaj informacje na temat poszczególnych wersji poprawki. Klikając na wiersze tej tabeli, przejdziemy do szczegółowego opisu konkretnej wersji.

2.1.4 Rejestracja użytkownika

Rejestracja nowego użytkownika polega na tym, że jego login, hasło i inne dodatkowe dane dodawane są do bazy naszej aplikacji, a dokładnie do tabeli użytkowników, gdzie te dane są przechowywane.

Rejestracja zaczyna się od wypełnienia formularza, w którym użytkownik musi uzupełnić takie pola jak: imię (first name), nazwisko (last name), adres e-mail (e-mail address) i dwukrotnie hasło (password) w celu wyeliminowania pomyłki. Skrypt znajdujący sie w pliku `register.php`, który generuje rejestrację, sprzężony jest z JavaScript, który sprawdza podawane wartości. Jeśli opuścimy jakąś wartość, pojawia się komunikat, a nieuzupełnione okienko podświetla się na czerwono. Adres e-mail musi mieć odpowiednią budowę zgodnie

z przyjętymi standardami. Hasło powinno składać się od 6 do 16 znaków i oczywiście musi być takie samo w polu `Password` i w polu `Verify password`.

Dane, które wstawi użytkownik przekazywane są do skryptu, który znajduje sie w pliku `register-save.php`. Na początku skryptu sprawdzane są dane z formularza (gdyż może zaistnieć sytuacja, że u użytkownika może być wyłączony JavaScript). Gdy wszystko jest w porządku, zostaje wysłana wiadomość do użytkownika, w celu aktywacji konta. Do momentu aktywacji konto jest nieaktywne i nie można się zalogować. Wysyłanie wiadomośći z instrukcją aktywacji konta ma na celu zapobiec rejestracjom dla żartu, a jednocześnie zweryfikowany zostanie adres e-mail użytkownika. Wynika stąd, że zarejestrować się mogą jedynie Ci, którzy mają skrzynkę pocztową. Wygląda to na poważne ograniczenie, ale nie wyobrażamy sobie administratora systemu, który nie posiada takiej skrzynki.

Adres e-mail użytkownika służy dalej jako jego identyfikator (popularnie nazywany login) w aplikacji. Oznacza to, że adres e-mail nie może się w bazie powtórzyć. Nie jest to jednak żadne ograniczenie bo różne osoby używają różnych adresów, z pewnością administratorzy systemów nie korzystają ze wspólnych skrzynek pocztowych.

W wiadomości otrzymanej od systemu, podany jest specjalny odnośnik, który wystarczy kliknąć aby uaktywnić założone konto. Po kliknięciu wejdziemy na stronę naszej aplikacji i będzie tam informacja, czy aktywacja się powiodła czy też nie. W zasadzie aktywacja może się nie udać jedynie w wyniku błędu aplikacji.

Po udanej aktywacji konta możemy logować się do aplikacji i swobodnie z niej korzystać. Nie ma limitu czasowego na używanie konta i konto nigdy nie wygasa.

2.1.5 Gdy zapomnimy hasło...

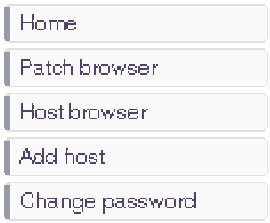
Gdy użytkownik zapomni hasło, ma możliwoć wygenerowania nowego hasła, które zostanie mu przekazane poprzez e-mail. Wykonuje się to po wybraniu z menu pozycji Forgot password. Na przywołanej w ten sposób stronie mamy prosty formularz zawierający tylko jedno pole — adres e-mail. Na ten adres zostanie wysłane nowe hasło.

Tak jak wcześniej pisaliśmy, hasła użytkowników są przechowywane w postaci zaszyfrowanej bez możliwości rozkodowania. Dlatego też gdy użytkownik zapomni hasło, generujemy nowe bo nie znamy jego aktualnego hasła.

2.2 Część użytkownika

Zarejestrowany użytkownik, z aktywnym kontem może zalogować się do naszej aplikacji i uzyskać w ten sposób dostęp do dodatowych funkcjonalności. Po zalogowaniu dostępne są wszystkie funkcje aplikacji z części ogólnodostępnej.

Dochodzi możliwość dodawania do bazy komputerów i zarządzania poprawkami na tych komputerach. Zmienia się nieco menu i wygląda ono teraz jak na rys. 2.4.



Rysunek 2.4: Menu użytkownika.

Nowo dodane pozycje mają następujące znaczenie:

Host browser – lista komputerów użytkownika,

Add host – formularz służący do dodania nowego komputera,

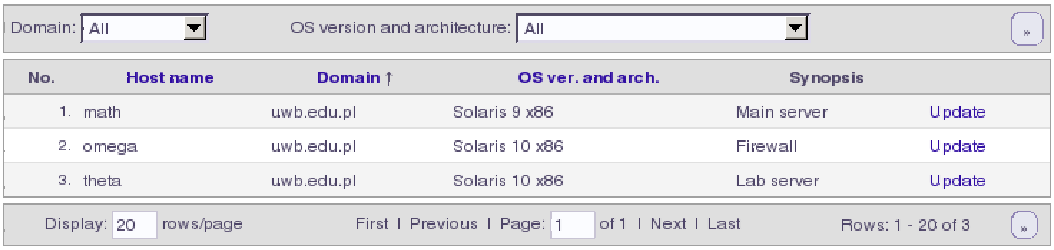
Change password – zmiana hasła.

Pod menu znajduje się ramka z imieniem i nazwiskiem zalogowanego użytkownika.

Domyślną stroną po zalogowaniu jest lista komputerów.

2.2.1 Lista komputerów

Na tej stronie w tabeli umieszczono spis wszystkich komputerów dodanych do tej pory przez użytkownika do bazy (rys. 2.5).



Rysunek 2.5: Lista komputerów.

Tabela bardzo przypomina przeglądarkę patchy. Podobieństwo nie jest przypadkowe, bo idea jest bardzo podobna — chodzi tutaj o szybkie wyszukanie interesujących nas komputerów. Na górze tabeli mamy dwa filtry: pierwszy służy do wybrania domeny, drugi, wersji systemu i platformy. Komputery użytkownika pogrupowane są w bazie poprzez przypisanie ich do domeny. Domena

nie musi się pokrywać z rzeczywistą domeną DNS, gdyż opisywany komputer nie musi być podłączony do Internetu. Może się zdarzyć więc, że dwóch różnych użytkowników ma tak samo nazywające się komputery w naszej bazie. Nie jest to problemem, bo klucz podstawowy w odpowiedniej tabeli utworzony jest z trzech pól: nazwy komputera, domeny oraz identyfikatora użytkownika.

Listę komputerów możemy sortować według nazwy komputera (host name), nazwy domeny (domain) oraz wersji systemu i platformy (OS ver. and arch.). Z sortowania korzystamy tak jak w przeglądarce patchy.

Dolna listwa tabeli pełni te same funkcje jak w przeglądarce patchy, możemy tu określić ilość wierszy na stronie oraz numer strony.

Kiedy klikniemy wiersz z opisem komputera, przejdziemy na stronę zawierającą szczegółowe informacje na temat zainstalowanych na tym komputerze poprawek. Dokładniej zawartość tej strony zostanie opisana w kolejnym podrodziale. Przy każdym komputerza na liście dodatkowo znajduje się przycisk Update, którego wciśnięcie powoduje załadowanie formularza, dzięki któremu będziemy mogli zaktualizować informacje o danym komputerze. Formularz ten jest taki sam jak formularz, służący do dodania nowego komputera. Różnica polega na tym, że przy aktualizacji (ang. update) cześć pól jest od razu wypełniana. Można je zostawić bez zmian lub wpisać inne dane. Można w ten sposób zmienić nazwę komputera lub dodać opis do komputera.

2.2.2 Dodanie komputera

Nowy komputer dodajemy wypełniając stosowny formularz. Musimy w nim okreslić nazwę komputera i przypisać mu domenę. Dla ułatwienia obok pola domeny umieszczono listę wyboru domen wcześniej użytych. Podajemy tutaj także numer wersji sytemu i jej platformę. Wyboru dokonujemy z listy, co zmniejsza prawdopodobieństwo błędu.

Bardzo ważne dane, jakie musimy wstawić do bazy, dodając nowy komputer to lista zainstalowanych na tym komputerze patchy. Trudno byłoby taką listę przygotować ręcznie, byłoby to strasznie pracochłonne i ciężko byłoby uniknąć błędu. Dlatego też wstawiamy tutaj gotowy plik sporządzony za pomocą programu `showrev` z opcją `-p`.

Kolejnym ważnym zestawem informacji jest lista zainstalowanych na komputerze pakietów oprogramowania. Listę taką przygotowujemy programem `pkginfo` z opcją `-l`. Wynik programu zachowujemy w pliku, a pełną do niego ścieżkę wstawiamy do odpowiedniego pola formularza. Oba pliki zostaną przesłane do serwera i tam poddane obróbce.

Każdy z komputerów możemy opisać, dodając krótki tekst. Przy większej ilości komputerów taki opis może być bardzo wygodny. Możemy tu notować różne rzeczy związane z instalacją patchy.

W formularzu mamy przycisk Delete za pomocą którego możemy usunąć dany komputer z bazy. Usunięcie jest nieodwracalne. Powinniśmy zatem zachować ostrożność.

2.2.3 Informacje o poprawkach na danym komputerze

Do strony informacji o poprawkach na danym komputerze przechodzimy, klikając nazwę tego komputera. Dane które są na tej stronie, uporządkowane są według następujących kategorii:

- Opis komputera (Host details)

  Są to wiadomości o wybranym komputerze: jego nazwa, domena, wersja systemu Solaris i jego platforma, data ostatniej aktualizacji i data utworzenia.

  Pod tymi informacjami znajduje się odnośnik do aktualizaji danych o naszym komputerze (Update host).

- Aktualne poprawki (Up to date patches)

  Pod tym nagłówkiem zamieszczona jest lista aktualnych patchy, których ostatnio wydana wersja jest taka sama jak na tym komputerze.

- Poprawki nieaktualne (Outdated patches)

  Kolejna lista dotyczy patchy, które były zainstalowane wcześniej a teraz wymagane jest zainstalowanie ich nowszych wersji.

- Poprawki nowsze na komputerze (Patches newer on the system)

  Rzadko ale czasem się zdarza, że na komputerze mamy zainstalowaną nowszą wersję patcha niż ta dostępna oficjalnie na SunSolve. Może się tak zdarzyć przy instalacji nowszej wersji systemu, z którą instalowane są różne poprawki.

- Poprawki zastąpione (Obsoleted patches)

  Tutaj znajdziemy listę patchy, które zostały zamienione przez inne.

- Poprawki wycofane (Withdrawn patches)

  Te patche, które się tu znajdują zostały wycofane. Krótko mówiąć, robiły więcej złego niż dobrego. Administrator powinnien je usunąć i zainstalować na ich miejsce inne proponowane poprawki.

- Nieznane poprawki (Unknown patches)

  Nasza baza danych informacji o poprawkach sięga początku bieżącego roku, a więc prawdopodobne jest to, iż możemy nie mieć wszystkich informacji o patchach. Lista takich patchy, o których nic nie wiemy, znajduje się właśnie tutaj.

- Nieinstalowane poprawki (Uninstalled patches)

Tutaj mamy listę patchy, które nigdy nie były instalowane na naszym komputerze, natomiast dotyczą pakietów oprogramowania u nas instalowanego. Administrator prawdopodobnie będzie zainteresowany instalacją tych poprawek.

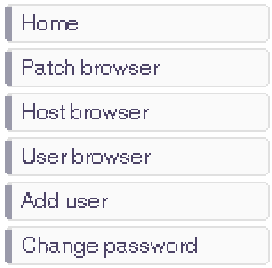
2.2.4 Zmiana hasła

Każdy użytkownik, któremu znudzi się jego hasło, ma możliwość jego zmiany. Po kliknięciu przycisku `Change password` pojawi się strona, gdzie użytkwonik może wpisać swoje nowe hasło. Nowe hasło powinno składać się od 6 do 16 znaków i oczywiście musi być takie samo w polu `Password` i w polu `Verify password`.

2.3 Część administracyjna serwisu

Do tej części dostęp ma jedynie administrator aplikacji. Administratorem jest użytkownik z identyfikatorem *admin*. Zalogowany administrator ma dostęp do dodatkowych funkcji takich jak: dodanie nowego użykownika, czy wgląd do spisu wszystkich użytkowników. Dostępne są także wszystkie funkcje, które były w części użytkownika. Nie ma tutaj jednak żadnych ograniczeń. Administrator widzi wszystkie komputery. Ma możliwość ich usuwania. Poszczególnych użytkowników może zobaczyć także tylko administrator, sami użytkownicy nie mają takiej możliwości.

Menu wygląda znów nieco inaczej niż w poprzednich częściach naszej aplikacji. Jego wygląd przedstawiony jest na rysunku 2.6.



Rysunek 2.6: Menu administratora

Nowo dodane pozycje mają następujące znaczenie:

User browser – lista użytkowników,

Add user – formularz służący do dodania nowego użytkownika.

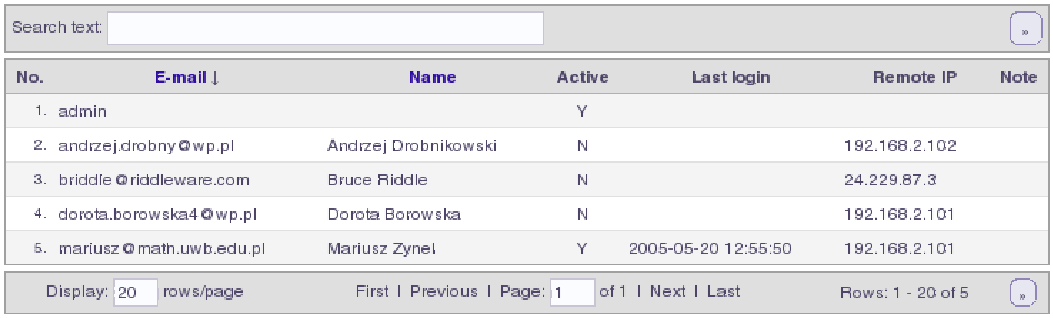
Pod menu znajduje się ramka z napisem `Administrator`. Jest to informacja o tym, kto jest zalogowany.

2.3.1 Lista użytkowników

Na tej stronie umieszczono spis wszystkich użytkowników, którzy zarejestrowali się do tej pory i zostali dodani do naszej bazy danych. Na liście znajduje się e-mail, imię i nazwisko danego użytkownika, informacja o tym czy jego konto jest aktywne, data ostatniego logowania oraz adres komputera, z którego dany użytkownik się loguje. Na górze tabeli umieszczona jest wyszukiwarka użytkowników według imienia, nazwiska lub adresu e-mail.

Dolna listwa tabeli pełni te same funkcje jak w tabeli z listą komputerów czy też w przeglądarce patchy, możemy tu określić ilość wierszy na stronie oraz numer strony.

Listę użytkowników możemy sortować według e-maila bądź nazwy użytkownika (nazwisko i imię). Z sortowania korzystamy, klikając odpowiedni nagłówek. Wygląd tabeli z użytkownikami przedstawiony jest na rys. 2.7.


Search text:

No.	E-mail ↓	Name	Active	Last login	Remote IP	Note
1.	admin		Y			
2.	andrzej.drobny@wp.pl	Andrzej Drobnikowski	N		192.168.2.102	
3.	briddle@riddleware.com	Bruce Riddle	N		24.229.87.3	
4.	dorota.borowska4@wp.pl	Dorota Borowska	N		192.168.2.101	
5.	mariusz@math.uwb.edu.pl	Mariusz Zynel	Y	2005-05-20 12:55:50	192.168.2.101	

Display: 20 rows/page First | Previous | Page: 1 of 1 | Next | Last Rows: 1 - 20 of 5

Rysunek 2.7: Lista użytkowników

Kiedy administrator kliknie w wybranego użytkownika, przejdzie na stronę zawierającą szczegółowe informacje o nim. W tym momencie może modyfikować dane użytkownika np. zmienić mu konto z aktywnego na nieaktywne badź odwrotnie. Administrator może także usunąć danego użytkownika.

2.3.2 Dodanie użytkownika

Nowego użytkownika dodajemy wypełniając specjalny formularz. Musimy w nim wypełnić takie pola jak: imię, nazwisko, adres e-mail i dwukrotnie hasło. Możemy także określić, czy konto nowego użytkownika ma być aktywne czy nie. Jeśli administrator ma jakieś uwagi odnośnie tego użytkownika, może je wpisać w polu `Note`.

Rozdział 3

Implementacja

3.1 Struktura aplikacji

W przypadku nawet niedużej aplikacji internetowej mamy do czynienia z wieloma różnymi plikami. Wśród nich znajdują się dokumenty HTML z podstawową treścią, pliki CSS odpowiadające za styl strony, obrazy GIF, PNG lub JPG umieszczane na stronie oraz skrypty JavaScript. W większych aplikacjach, w których wykorzystujemy narzędzia typu PHP, czy Meta-HTML dochodzą pliki biblioteczne zawierające wielokrotnie używany kod. Logiczne rozmieszczenie wszystkich plików, może nie ma wpływu na funkcjonalność aplikacji, ale ma ogromne znaczenie dla autorów. W aplikacji o przejrzystej strukturze dużo łatwiej odnaleść potrzebne pliki. Układ katalogów nabiera jeszcze większego znaczenia, gdy dostęp do aplikacji jest kontrolowany i mamy kilka poziomów dostępu, tak jak w naszej aplikacji.

W naszej aplikacji założone mamy następujące katalogi:

`admin`

Jak sama nazwa wskazuje, katalog `admin` dotyczy części administracyjnej aplikacji. Są w nim umieszczone skrypty PHP, które odpowiadają za wygląd i funkcjonowanie części adminisracyjnej aplikacji. Oto spis tych skryptów:

`contents.php`	`host.php`	`patch.php`
`host-browse.php`	`index.php`	`user-browse.php`
`host-delete.php`	`passwd-save.php`	`user-delete.php`
`host-edit.php`	`passwd.php`	`user-edit.php`
`host-save.php`	`patch-browse.php`	`user-save.php`

W tym katalogu znajduje się jeszcze ukryty plik `.htaccess`. Jest on częścią konfiguracji serwera Apache i odpowiada za kontrolę dostępu do tego katalogu przez Internet.

Skrypt `contents.php` opisuje zawartość menu. Seria skryptów: `host-browse.php`, `host-delete.php`, `host-edit.php`, `host-save.php` i `host.php` związana jest z danymi na temat komputera użytkownika i dotyczy odpowiednio: przeglądarki, usuwania, edycji, zapisywania zmian oraz głównej strony informacji o tym komputerze. Skrypt `index.php` odpowiada na stronę główną po zalogowaniu administratora. Skrypty `passwd.php` i `passwd-save.php` dotyczą hasła użytkownika. Pierwszy z nich odpowiada za formularz zmiany hasła, drugi zapisuje nowe hasło. Informacje na temat poprawek są wyświetlane na stronach generowanych dzięki skryptom `patch-browse.php` i `patch.php`. Pierwszy dotyczy przeglądarki patchy a drugi informacji o konkretnym patchu. Ostatnia seria skryptów dotyczy informacji o użytkowniku. Skrypty: `user-browse.php`, `user-delete.php`, `user-edit.php` i `user-save.php` odpowiadają kolejno za: przeglądarkę użytkowników, usuwanie użytkownika, formularz edycji i zapis danych z tego formularza.

`assets`

W tym katalogu znajdują się grafiki umieszczane na stronach aplikacji. Mamy tutaj tylko jeden plik `pwrdsx86.png` – logo serwisu

`http://solaris-x86.org`.

`bin`

Ten katalog zawiera skrypty perlowe i shellowe, stanowiące integralną część naszej aplikacji:

```
PatchDB.pm          insert-pkginfo.pl   insert-xref.pl
getdate-xref.pl     insert-showrev.pl   update-xref.sh
```

Skrypty `insert-pkginfo.pl` i `insert-showrev.pl` wołane są przy dodawaniu nowego komputera do bazy, natomiast `update-xref.sh` wraz z `insert-xref.pl` uruchamiane są cyklicznie w celu aktualizacji bazy poprawek.

Wszystkie powyższe skrypty zostały napisane przez pana Andrzeja Drobnikowskiego w ramach jego pracy dyplomowej.

`include`

Jest to bardzo ważny katalog gdyż zawiera najczęściej wykorzystywane pliki. Umieszczone są w nim skrypty PHP, wstawiane bezpośrednio do właściwych dokumentów:

```
browser.php  done.php     footer.php   header.php
home.php     init.php     patch.php
```

Każda strona naszej aplikacji składa się z 5 następujących części: inicjalizacji (`init.php`), nagłówka (`header.php`), części głównej, stopki (`footer.php`) i zakończenia (`done.php`). Wszystkie składowe poza częścią główną są wspólne i dlatego umieszczono je tutaj. Dokładny opis budowy strony aplikacji przedstawiony jest w 3.5.

Znajdują się tutaj również większe, powtarzające się elementy aplikacji takie jak: strona domowa (`home.php`), przeglądarka patchy (`browser.php`) i opis patcha (`patch.php`).

`lib`

W tym katalogu umieszczone są biblioteki z często wykorzystywanymi funkcjami:

```
config.php  stdlib.php  util.php
```

`script`

Skrypty napisane w JavaScript, które wykorzystujemy w naszej aplikacji umieszczone są w tym katalogu:

```
forgot.js       host-update.js  passwd.js
host-add.js     main.js         register.js
```

Poza skryptem `main.js` zawierającym wspólne funkcje dla innych skryptów, pozostałe służą do sprawdzenia, czy odpowiednie formularze w aplikacji są dobrze wypełnione.

`style`

W tym katalogu mamy plik `main.css` zawierający style CSS, opisujące formatowanie i dobór kolorów poszczególnych elementów aplikacji takich jak: przyciski, elementy formularzy, tabele i tekst. CSS odpowiada za wygląd strony. Jeśli będziemy chcieli zmienić wygląd strony, wystarczy, że zmienimy CSS.

`user`

Katalog ten ściśle wiąże się z częścią aplikacji, przeznaczoną dla użytkownika. Zawiera on skrypty PHP, które generują strony po zalogowaniu użytkownika:

```
contents.php      host-save.php     passwd.php
host-browse.php   host.php          patch-browse.php
host-delete.php   index.php         patch.php
host-edit.php     passwd-save.php
```

Podobnie jak w katalogu `admin`, tutaj też znadjuje się ukryty plik `.htaccess`.

Poszczególne skrytpy mają podobne znaczenie jak w katalogu `admin`.

Na korzeniu aplikacji oprócz wyżej wymienionych katalogów katalogów znajdują się także następujące skrypty PHP:

`activate.php`	`forgot.php`	`patch.php`
`contents.php`	`index.php`	`register-save.php`
`forgot-send.php`	`patch-browse.php`	`register.php`

Te skrypty generują dokumenty dostępne bez ograniczeń. Skrypt `activate.php` odpowiada za aktywację konta, `contents.php` za wygląd menu w tej części aplikacji, `forgot.php` i `forgot-send.php` odpowiadają za wysłanie nowego hasła użytkownikowi, który je zapomniał. Za stronę główną, która wygeneruje się po wpisaniu adresu naszego serwisu, odpowiada `index.php`. Skrypty `patch-browse.php` i `patch` dotyczą jak poprzednio stron z informacją o patchach. Dwa ostatnie skrypty `register.php` i `register-save.php` odpowiadają za stronę z formularzem do rejestracji i za przetwarzanie tego formularza.

3.2 Baza danych

Stworzona przez nas aplikacja stanowi interfejs do bazy danych poprawek na system Solaris. Struktura tej bazy danych została zaprojektowana we współpracy z panem Andrzejem Drobnikowskim. Jako serwer bazy danych wykorzystujemy MySQL.

Baza poprawek nosi nazwę `patchdb`. Znajdują się w niej następujące tabele:

`hosts` – komputery użytkowników,

`pkginfo` – pakiety oprogramowania zainstalowanego na poszczególnych komputerach,

`showrev` – patche zainstalowane na poszczególnych komputerach,

`users` – użytkownicy,

`xref` – aktualne dane o najnowszych wersjach patchy,

`xrefarch` – lista platform dla danego patcha,

`xrefcon` – lista patchy konfliktowych z danym patchem,

`xrefhistory` – dane o wszystkich wersjach patchy,

`xrefpkg` – lista pakietów oprogramowania modyfikowanego przez danego patcha,

`xrefreq` – lista patchy wymaganych przez danego patcha,

`xrefwdrawn` – patche wycofane.

Najważniejszymi tabelami, z których korzysta nasza aplikacja są: `xref` i `xrefhistory`. Obie mają identyczną strukturę opisaną w tabeli 3.1. Zawarte w nich dane o patchach wyświetlane są na wielu stronach: strona główna, przeglądarka patchy, opis szczegółowy patcha oraz lista patchy zainstalowanych na danym komputerze. Aby uzyskać pełną informację na temat danej poprawki, należy sięgnąć do kilku tabel jednocześnie, a mianowicie do `xref`, `xrefarch`, `xrefcon`, `xrefpkg` i `xrefreq`.

Field	Type	Null	Key	Default	Extra
patchid	int(6) unsigned zerofill		PRI	000000	
patchrev	int(2) unsigned zerofill		PRI	00	
reldate	date	YES		NULL	
rflag	enum('Y','N')			N	
sflag	enum('Y','N')			N	
oflag	enum('Y','N')			N	
bflag	enum('Y','N')			N	
yflag	enum('Y','N')			N	
osarch	varchar(64)	YES	MUL	NULL	
synopsis	text	YES		NULL	

Tabela 3.1: Struktura tabeli `xref`.

Poszczególne kolumny mają następujące znaczenie:

`patchid` – identyfikator,

`patchrev` – wersja,

`reldate` – data wydania,

`rflag` – (recommended) patch jest rekomendowany do zainstalowania na naszym komputerze,

`sflag` – (security) dotyczy bezpieczeństwa systemu,

`oflag` – (obsoleted) został zastąpiony innym,

`bflag` – (bad patch) wycofany,

`yflag` – (year 2000) dotyczy problemów związanych z rokiem 2000,

`osarch` – wersja i architektura systemu,

`synopsis` – krótki opis.

Dane użytkowników przechowywane są w tabeli `users`. Jest to niezwykle ważna tabela dla naszej aplikacji. Na niej oparta jest kontrola dostępu, to z tej tabeli pobierane są identyfikator i hasło użytkownika przy logowaniu. Opis jej budowy znajduje się w tabeli 3.2.

Field	Type	Null	Key	Default	Extra
id	int(11)		PRI	NULL	auto_increment
email	varchar(64)		UNI		
password	varchar(16)	YES		NULL	
firstname	varchar(32)	YES		NULL	
lastname	varchar(32)	YES	MUL	NULL	
active	enum('Y','N')	YES		Y	
lastlogin	datetime	YES		NULL	
remoteip	varchar(16)	YES		NULL	
note	tinytext	YES		NULL	
updated	timestamp(14)	YES		NULL	
created	timestamp(14)	YES		NULL	

Tabela 3.2: Struktura tabeli `users`.

Kolumny tej tabeli mają znaczenie jak niżej:

`id` – identyfikator,

`email` – adres e-mail użytkownika,

`firstname` – imię,

`lastname` – nazwisko,

`password` – hasło w postaci zaszyfrowanej,

`active` – czy konto jest aktywne,

`lastlogin` – data ostatniego logowania,

`remoteip` – adres komputera, z którego użytkownik loguje się,

`note` – uwagi administratora,

`updated` – czas ostatniej modyfikacji,

`created` – czas utworzenia danego rekordu.

Podstawowe dane o komputerach użytkowników przechowywane są w tabeli `hosts`(tab. 3.3).

Field	Type	Null	Key	Default	Extra
id	int(11)		PRI	NULL	auto_increment
hostname	varchar(32)		MUL		
domain	varchar(64)				
osarch	varchar(64)		MUL		
user	int(11)		MUL	0	
synopsis	tinytext	YES		NULL	
updated	timestamp(14)	YES		NULL	
created	timestamp(14)	YES		NULL	

Tabela 3.3: Struktura tabeli `hosts`.

`id` – identyfikator,

`hostname` – nazwa hosta,

`domain` – domena do której przypisany jest komputer danego użytkownika,

`osarch` – architektura danego komputera,

`user` – identyfikator użytkownika,

`synopsis` – wersja systemu Solaris zainstalowanego na danym komputerze,

`updated` – czas ostatniej modyfikacji,

`created` – czas utworzenia danego rekordu.

Tabela `hosts` jest ściśle powiązana z tabelami `showrev` i `pkginfo`. Zapytania łączone tych tabel dają pełną informację o tym, jakie patche i jakie pakiety są zainstalowane na danym komputerze. Na podstawie tych danych oraz danych zawartych w tabeli `xref` możemy sporządzić listę patchy aktualnych, nieaktualnych oraz tych, które jeszcze nie są zainstalowane.

3.3 Obsługa błędów

W aplikacjach takich jak nasza, gdzie ma miejsce częste odwoływanie się do bazy danych oraz interakcja z użytkownikiem poprzez formularze, bardzo ważna jest obsługa sytuacji krytycznych. Do takich sytuacji zaliczamy błędne wprowadzenie danych przez użytkownika, błąd podczas komunikacji z bazą

danych oraz inne błędy, które powodują, że kontunuacja wykonywania skryptu nie ma sensu. Jedyne co możemy wtedy zrobić, to przekazać informację o błędzie i przerwać wykonanie skryptu. Zadanie to realizuje funkcja `error()` z biblioteki `stdlib.php`.

```
function error($msg) {
    global $error;

    $error = $msg;

    @require_once ROOT_DIR . '/include/footer.php';
    @require_once 'footer.php';
    @require_once ROOT_DIR . '/include/done.php';
    @require_once 'done.php';
    exit;
}
```

Jej zadanie jest bardzo proste, ustawia ona globalną zmienną `$error` na przekazany do niej jako argument tekst komunikatu o błędzie i przerywa wykonanie skryptu. Ładowane jest zakończenie strony `done.php`, w którym realizowane jest wyświetlenie komunikatu o błędzie. Zawartość `done.php` znajduje się w A.4.

Poza samą obsługą sytuacji wyjątkowych, z punktu widzenia autora aplikacji, bardzo ważne jest aby mieć możliwość śledzenia skryptów i podejrzenia wartości poszczególnych zmiennych w trakcie ich wykonywania. Jest to potrzebne przy diagnostyce błędów.

W biliotece `stdlib.php` znajduje się funkcja `debug()`, która przekazany do niej jako argument ciąg znaków, dopisuje do globalnej tablicy `$debug_output`. Zawartość tej tablicy jest wyświetlana na końcu strony jeśli skrypt wykonywany jest w trybie *debug*. Poniżej przedstawiamy kod funkcji `debug()`:

```
function debug($msg) {
    global $debug_output;

    if (DEBUG) {
        array_push($debug_output, $msg);
    }
}
```

3.4 Komunikacja z bazą danych

Aby umożliwić dostęp do bazy danych `patchdb` zostało utworzone konto na sewerze MySQL o tej samej nazwie co baza i nadano mu wszelkie możliwe uprawnienia, łącznie z tworzeniem i usuwaniem tabel.

W naszej aplikacji odwołania do bazy danych są bardzo częste i odgrywają zasadniczą rolę. Aby mieć możliwość łątwej diagnostyki oraz błędów przy tych odwołaniach opracowaliśmy swój własny interfejs do bazy danych, wykorzystujący oczywiście standardowe funkcje PHP. W jego skład wchodzi 6 funkcji. Umieszczone one zostały w bibliotece `stdlib.php`.

Funkcja `db_connect()` realizuje połączenie z bazą poprawek. Korzysta ona z globalnych zmiennych: `$db_host`, `$db_user`, `$db_pass`, `$db_name` oraz `$dbh`, w których przechowywane są odpowiednio: nazwa hosta, nazwa użytkownika, hasło, nazwa bazy i identyfikator połączenia.

```
function db_connect() {
  global $db_host, $db_user, $db_pass, $db_name, $dbh;

  if (! ($dbh = mysql_connect($db_host, $db_user, $db_pass))) {
      error("MySQL: Unable to connect to database: " . mysql_error());
  }

  if (! mysql_select_db($db_name, $dbh)) {
      error("MySQL: Unable to select database: " . mysql_error());
  }

  // Older versions of MySQL may not like that.

  if (! mysql_query("/*!40101 set names latin1 */", $dbh)) {
      error("MySQL: Unable to set character encoding: " . mysql_error());
   }
}
```

Zaczynamy od nawiązania połączenia przez `mysql_connect()`. Sprawdzamy czy się ono powiodło, jeśli nie, to zgłaszany jest stosowny błąd. W zmiennej `$dbh` przechowywany jest identyfikator połączenia z bazą.

Dalej za pomocą `mysql_select_db()` wybieramy bazę. Jeśli wystąpił błąd, zgłaszamy go.

Następnie ustalane jest kodowanie znaków na `latin1`. Konieczne jest to w nowszych wersjach MySQl, począwszy od wersji 4.01.01. Kodowanie znaków ma wpływ na sposób sortowania porównywania wyrażeń.

Rozłączenie z bazą realizuje funkcja `db_disconnect` przedstawiona poniżej.

```
function db_disconnect() {
    global $dbh;

    mysql_close($dbh);
}
```

Za pomocą poniższej funkcji możemy kierować nasze zapytania do bazy danych. Jeśli funkcja `mysql_query` zwróci błąd, na ekran zostanie wypisany

odpowiedni komunikat. W trybie diagnostycznym treść zapytania będzie można obejrzeć na końcu strony.

```
function db_query($query) {
    global $dbh;

    debug($query);

    if (! ($result = mysql_query($query, $dbh))) {
        error("MySQL: Query failed: " . mysql_error() . ":\n$query\n");
    }
    return $result;
}
```

Funkcja `db_free` zwalnia pamięć, zajmowaną przez wynik ostatniej kwerendy.

```
function db_free($result) {
    mysql_free_result($result);
}
```

Bardzo często potrzebujemy pobrać z bazy pojedynczy wiersz. Ułatwia to funkcja `db_select`. Jako argument podajemy zapytanie, w wyniku uzyskujemy tablicę asocjacyjną, indeksowaną nazwami kolumn.

```
function db_select($query) {
    $result = db_query($query);
    $assoc = mysql_fetch_assoc($result);
    db_free($result);
    return $assoc;
}
```

Ostatnią funkcją jest `db_simple_select()`, która ułatwia odczytanie wyniku zapytania do bazy danych. Najpierw za pomocą funkcji `db_query()` wysyłane jest zapytanie a następnie przy pomocy `mysl_fetch_row()` pobierany jest pierwszy wiersz wyniku. Jako rezultat funkcja zwraca wartość pierwszej kolumny z wiersza odpowiedzi.

```
function db_simple_select($query) {
    $result = db_query($query);
    $row = mysql_fetch_row($result);
    db_free($result);
    return $row[0];
}
```

Ta funkcja ma zastosowanie gdy np. chcemy odczytać ilość rekordów spełniających pewne kryteria. Interesuje nas wtedy tylko jedno, pierwsze pole wyniku.

3.5 Budowa strony

Wszystkie strony w aplikacji generowane są według określonego szablonu. Wyróżnić na nich można stałe elementy takie jak: nagłówek zawierający pasek tytułu oraz wyszukiwarkę, menu, stopkę z odnośnikami do podstron i datą ostatniej aktualizacji bazy. Są to elementy wizualne, natomiast są jeszcze pewne powtarzające się procedury, które trzeba wykonać generując każdą ze stron aplikacji, a których efektu nie będziemy mogli zobaczyć. Ponieważ na każdej ze stron korzystamy z bazy danych, więc przed rozpoczęciem tworzenia strony musimy z tą bazą nawiązać połączenie, a po wygenerowaniu strony rozłączyć się. Widać zatem, że wszytskie strony aplikacji mają pewne logiczne części wspólne. Te powtarzające się fragmenty stron umieściliśmy w osobnych skryptach w katalogu `include`.

Inicjalizacja strony – `init.php`

Za zaincjowanie strony odpowiada skrypt `init.php`. Zaczynamy tutaj od załadowania poprzez `require_once()` konfiguracji aplikacji `config.php` oraz standardowej biblioteki `stdlib.php`. W `config.php` znajdują się stałe określające nazwę hosta, nazwę bazy, nazwę użytkownika oraz jego hasło niezbędne do podłączenia się do bazy danych. Umieszczenie tego w jednym miejscu usprawnia nam nanoszenie zmian, gdyż wystarczy zmodyfikować dane w tym właśnie pliku. W bibliotece `stdlib.php` zdefiniowane są stałe oraz funkcje potrzebne w dalszej części aplikacji. Tutaj znajduje się zestaw funkcji do komunikacji z bazą danych.

Ten skrypt odpowiedzialny jest również za wygenerowanime nagłówka strony HTML. W nagłówku tym określamy zestaw znaków, jakich używamy w aplikacji. Jest to ISO-8859-1. Ponadto umieszczamy tutaj słowa kluczowe, opis, informację dla robotów oraz tytuł strony, który będzie się pojawiał na górnej listwie okna przeglądarki. W nagłówku HTML podłączamy styl CSS używany w naszej aplikacji oraz podstawowy zestaw funkcji JavaScript, odpowiednio z plików `main.css` oraz `main.js`.

Na koniec skryptu inicjowane jest połączenie do bazy danych poprzez `db_connect()`. Po podłączeniu do bazy, odczytywane są dane zalogowanego użytkownika.

Fragment strony wygenerowany podczas inicjalizacji nie zawiera żadnnych wizualnych elementów. Wykonywane są tutaj procedury przygotowujące do reszty dokumentu.

Kod skryptu `init.php` umieszczony jest w A.1.

Nagłówek – `header.php`

Skrypt `header.php` odpowiada za pasek tytułowy z wyszukiwarką i za menu. Zawartość menu pobierana jest z listy asocjacyjnej o nazwie `$contents`.

Jako indeksy na tej liście występują nazwy plików, natomiast jako wartości nazwy podstron odpowiadających tym plikom. Te właśnie nazwy widzimy jako pozycje w menu. Menu jest listą zwykłych odnośników HTML odpowiednio sformatowanych.

Jeżeli jest zalogowany użytkownik, to pod menu będzie także wyświeltlone jego imię i nazwisko, a w przypadku administratora napis `Administator`.

Kod skryptu `header.php` umieszczony jest w A.2.

Stopka – `footer.php`

Skrypt `footer.php` generuje fragment strony znajdujący się na samym dole. Mamy tam zestaw odnośników do podstron, taki sam jak w menu. Dodatkowo mamy odnośnik do góry strony i strony poprzedniej. Umieszczone one zostały tutaj dla wygody użytkowników i usprawnienia nawigacji. W prawym dolnym rogu umieszczona jest data ostatniej aktualizacji bazy poprawek. Jest to data ostatniego z wgranych plików `patchdiag.xref`.

Kod skryptu `footer.php` umieszczony w A.3.

Zakończenie strony – `done.php`

W skrypcie `done.php` na samym początku rozłączamy się z bazą danych za pomocą `db_disconnect()`. Jeśli wcześniej wystąpił błąd wyświetlany jest tutaj komunikat o tym błędzie. Błąd może wystąpić wewnątrz konstrukcji HTML, np. tabeli. Nie będziemy mogli wtedy prawidłowo domknąć tej konstrukcji. Nie mamy zatem pewności, w jakim miejscu strony jesteśmy. Dlatego też komunikat o błędzie umieszczany jest a tabeli zajmującej całe okienko przeglądarki i przesuniętej do jej skrajnego, lewego, górnego rogu. W ten sposób przykrywamy wygenerowaną do tej pory stronę. Pod komunikatem o błędzie jest przycisk do cofnięcia się na stronę poprzednią.

W tym skrypcie wypisywane są również dane diagnostyczne zebrane podczas generowania całej strony, w szczególności mogą to być wszystkie zapytania do bazy danych. Jeśli jesteśmy w trybie diagnostycznym to na końcu strony wyświetlona zostanie zawartość listy `$debug_output` wiersz po wierszu.

Kod skryptu `done.php` umieszczony jest w A.4.

Generowanie strony

Aby wszystkie opisane wcześniej procedury zostały wykonane w odpowiednim momencie, każda strona aplikacji musi się do nich odwoływać. Każda podstrona zaczyna się od wczytania pliku `contents.php` oraz `header.php` poprzez `require_once()`. Nagłówek `header.php` wczytuje inicjalizację `init.php`. W ten sposób mamy gwarancję na to, że na początku każdej podstrony zostaną wykonane odpowiednie czynności. Teraz możemy przystąpić do realizacji zadania danej podstrony. Może to być wypisanie formularza, zapis do bazy

danych albo inne zadanie. Po jego zakończeniu należy wczytać i uruchomić skrypt `footer.php`. Z jego poziomu uruchomiony zostanie `done.php`.

Opisywane tutaj składowe strony stanowią szkielet aplikacji. Jest on wypełniany przez poszczególne podstrony. Przy takiej organizacji aplikacji łatwo możemy modyfikować zachowanie wszystkich jej podstron. Szybko możemy dodawać, usuwać i zmieniać wygląd stałych elementów aplikacji.

Dodatek A

Szkielet strony aplikacji

A.1 Inicjalizacja strony – `init.php`

```
<?php
        require_once ROOT_DIR . '/lib/config.php';
        require_once ROOT_DIR . '/lib/stdlib.php';

?>

<html>
<head>
  <meta http-equiv="content-type" content="text/html;
  charset=iso-8859-1">
  <meta name="keywords" content="Solaris, patch, database, manage,
  pkgadd, pkgrm, patchdiag.xref">
  <meta name="description" content="Solaris Patch Database">
  <meta name="robots" content="index,nofollow">
  <title>
    Solaris Patch Database
    <?
        foreach ($contents as $key => $value) {
            if (strcmp($key, THIS_DOC) == 0) {
                echo ' - ' . $value;
                break;
            }
        }
    ?>
 </title>
  <link rel="stylesheet" type="text/css" href="<? echo
  url_style('main.css') ?>">
  <link rel="shortcut icon" href="<? echo
  url_asset('favicon.ico') ?>">
  <script language="javascript" src="<? echo
```

```
  url_script('main.js') ?>"></script>
</head>

<body>
<a name="top"></a>

<?
    db_connect();

    if ($_SERVER['REMOTE_USER']) {
        $user = db_select("select * from users where
        email = '" . $_SERVER['REMOTE_USER'] . "'");
    }
?>
```

A.2 Nagłówek – header.php

```
<?php

    require_once 'init.php';

?>

<table class=page width=925 height=100% border=0 cellpadding=0
 cellspacing=0 align=center>
<tr valign=middle>
  <td colspan=2 class=pageheader>
    <table width=100% height=100% border=0 cellpadding=0
    cellspacing=0>
    <tr>
      <td><h1>Solaris Patch Database</h1></td>
      <td align=right>
        <form method=get action="patch-browse.php">
          <input type=text class=text name=search
          value="Search patches" size=20 onFocus="clearSearch(this)">
          <input type=submit class=button value="&raquo;">
        </form>
      </td>
    </tr>
    </table>
  </td>
</tr>
<tr valign=top>
  <td class=pagemenu>
    <table width=100% height=100% border=0 cellpadding=0
```

```
cellspacing=0>
    <tr valign=top>
      <td class=pagemenucont>
        <?
            foreach ($contents as $key => $value) {
                echo '<a href="' . $key . '">' . $value . '</a>';
            }

            if ($user{id}) {
                print "<div class=logged>\n";
                if ($user{email} == 'admin') {
                    print "Logged in:<br>Administrator\n";
                } else {
                    print "Logged in:<br>" . $user{firstname} .
                    " " . $user{lastname} . "\n";
                }
                print "</div>\n";
            }
        ?>
      </td>
    </tr>
    <tr valign=bottom>
      <td align=center><a href="http://solaris-x86.org"
      target=_blank><img src="<? echo url_asset('pwrdsx86.png')?>">
      </a></td>
    </tr>
    </table>
  </td>
  <td class=pagebody>
```

A.3 Stopka – `footer.php`

```
 </td>
</tr>
<tr valign=middle>
  <td colspan=2 class=pagefooter>
    <table width=100% height=100% border=0 cellpadding=0
    cellspacing=0>
    <tr valign=midddle>
      <td class=pagefootercont>
        <?
            foreach ($contents as $key => $value) {
                echo '<a href="' . $key . '">' . $value .
                     '</a>  ';
            }
```

```
        ?>
      </td>
      <td class=pagefooternav>
        <a href="#top">Top</a>
        <a href="javascript: history.back()">Back</a>
      </td>
    </tr>
    <tr valign=top>
      <td class=pagefootercopy>
        &copy; Copyright 2005 Riddleware. All rights reserved.
      </td>
      <td class=pagefooterdate>
        Last updated:
            <?
                print db_simple_select("select max(reldate)
                      from xref");
            ?>
      </td>
    </tr>
    </table>
  </td>
</tr>
</table>


<?

    require_once 'done.php';

?>
```

A.4 Zakończenie strony – done.php

```
<?php

    db_disconnect();

    if ($error) {
        ?>

        <table id=error class=error width=100% height=100% border=0
        cellpadding=0 cellspacing=0>
        <tr valign=middle>
          <td align=center>
            <form method=get action="javascript: history.back()">
```

```
            <table class=dialog width=450 height=100 border=0
            cellpadding=0 cellspacing=0>
            <tr>
              <td>
                <?
                      print "$error\n";
                      print "<pre>\n";
                      foreach ($error_output as $error) {
                          print "$error\n";
                      }
                      print "</pre>\n";
                ?>
              </td>
            </tr>
            <tr>
              <td height=60 align=right>
                <input type=submit class=button value="Back">
              </td>
            </tr>
            </table>
          </form>
        </td>
      </tr>
      </table>

      <?
  }

  if (DEBUG) {
      print "<pre>\n";
      foreach ($debug_output as $line) {
          print "$line\n";
      }
      print "</pre>\n";
  }
?>

</body>
</html>
```

Bibliografia

[1] Welling Luke, Thomson Laura *PHP i MySQL Tworzenie stron WWW*, Helion, 2002.

[2] Aktinson Leon, *Core MySQL*, Helion, 2003.

[3] Podręcznik PHP, `http://pl2.php.net/manual/pl/`

[4] MySQL, Dokumentacja techniczna, `http://www.mysql.com/`

[5] Kursy i artykuły o PHP i MySQL, `http://webcity.pl/webcity/`

[6] SunSolve, Centrum informacji o poprawkach dla Solaris `http://sunsolve.sun.com/`