

AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

Wydział Inżynierii Metali i Informatyki Przemysłowej



PRACA DYPLOMOWA INŻYNIERSKA

pt.

„Projekt i implementacja aplikacji
webowej do przechowywania i analizy
wyników badań laboratoryjnych”

Imię i nazwisko dyplomanta:

Sebastian Wiewióra

Kierunek studiów:

Informatyka Stosowana

Nr albumu:

262180

Promotor:

dr Dorota Byrska-Wójcik

Recenzent:

dr inż. Barbara Mrzyglód

Podpis dyplomanta:

Podpis promotora:

Kraków 2018

„Uprzedzony o odpowiedzialności karnej na podstawie art. 115 ust. 1 i 2 ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t.j. Dz.U. z 2006 r. Nr 90, poz. 631 z późn. zm.): „Kto przywłaszcza sobie autorstwo albo wprowadza w błąd co do autorstwa całości lub części cudzego utworu albo artystycznego wykonania, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 3. Tej samej karze podlega, kto rozpowszechnia bez podania nazwiska lub pseudonimu twórcy cudzy utwór w wersji oryginalnej albo w postaci opracowania, artystyczne wykonanie albo publicznie zniekształca taki utwór, artystyczne wykonanie, fonogram, wideogram lub nadanie.”, a także uprzedzony o odpowiedzialności dyscyplinarnej na podstawie art. 211 ust. 1 ustawy z dnia 27 lipca 2005 r. Prawo o szkolnictwie wyższym (t.j. Dz. U. z 2012 r. poz. 572, z późn. zm.) „Za naruszenie przepisów obowiązujących w uczelni oraz za czyny uchybiające godności studenta student ponosi odpowiedzialność dyscyplinarną przed komisją dyscyplinarną albo przed sądem koleżeńskim samorządu studenckiego, zwanym dalej "sądem koleżeńskim"”, oświadczam, że niniejszą pracę dyplomową wykonałem(-am) osobiście i samodzielnie i że nie korzystałem(-am) ze źródeł innych niż wymienione w pracy.”

Kraków, dnia

Podpis dyplomanta.....

Spis treści

1. Wstęp	5
2. Cel i zakres pracy.....	6
I. Część studialna	7
1. Opis wykorzystywanych technologii.....	8
1.1. Platforma LAMP.....	8
1.1.1. System operacyjny	9
1.1.2. Serwer WWW.....	10
1.1.3. System zarządzania relacyjną bazą danych	10
1.2. Język PHP	12
1.3. Framework do aplikacji webowych.....	13
1.3.1. Symfony.....	13
2. Statyczna próba rozciągania	15
2.1. Wskaźniki wytrzymałości.....	16
2.2. Wskaźnik plastyczności	17
3. Statyczna próba ściskania	19
II. Część projektowa Wykonanie aplikacji do przechowywania i analizy wyników badań laboratoryjnych	21
1. Przedmiot opracowania.....	22
1.1. Ogólny opis.....	23
1.1.1. Funkcje programu	23
1.1.2. Charakterystyka użytkowników	24
1.1.3. Ogólne ograniczenia	24
1.2. Szczegółowe wymagania	24
1.3. Wykonanie obliczeń	27
1.3.1. Obliczenie pola przekroju poprzecznego próbki walcowej S	27
1.3.2. Obliczenie naprężenia z eksperymentu S_{exp}	27
1.3.3. Obliczenie odkształcenia ε	28
2. Implementacja.....	29

2.1. Wprowadzanie danych i analiza wyników eksperymentów wykonanych na maszynie Zwick	30
2.1.1. Klasa Zwick	30
2.1.2. Klasa ZwickData.....	32
2.1.3. Kontroler	33
2.1.4. Odczytywanie danych z plików	35
2.1.5. Generowanie raportu.....	38
2.2. Dodawanie materiałów i przypisywanie ich do zadań.....	39
2.3. Grupowanie materiałów i zadań w projekty	40
2.4. Obsługa kont użytkownika.....	40
2.5. Interaktywne menu.....	41
3. Testowanie	43
4. Podsumowanie	52
5. Bibliografia	54
6. Spis rysunków	55
7. Spis tabel.....	56

1. Wstęp

Analiza i przechowywanie wyników badań laboratoryjnych jest nieodłączną częścią każdego laboratorium. Na każde przeprowadzone badanie składają się procesy:

- określenie celu, ustalenie parametrów,
- wykonanie próby,
- pobranie wyników z urządzeń pomiarowych,
- odczytanie wyników, analiza,
- wykonanie dodatkowych obliczeń,
- interpretacja wyników,
- archiwizacja danych.

Do każdego badania należy uwzględnić dodatkowy nakład pracy na uporządkowanie wyników, ich odpowiednie przechowywanie, oznaczenie, a także analizę i interpretację wyników. Automatyzacja chociaż części tego procesu może skrócić czas wykonania zadania, zmniejszyć ryzyko błędów oraz umożliwić łatwy dostęp do wcześniej wykonanych badań do celów statystycznych.

Aplikacja wspomagająca analizę wyników badań może mieć zastosowanie w przemyśle, do celów naukowych oraz dydaktycznych. Przedsiębiorstwo zlecające wykonanie pomiarów, mogłoby uzyskać zdalny dostęp do wyników badań już w momencie ich wykonywania. Pracownicy naukowcy mogą korzystać z narzędzia w celu uporządkowania dużych ilości przeprowadzonych pomiarów i skrócić czas potrzebny na ich analizę oraz uzyskać dodatkowe możliwości analizy zbiorczej serii prób. Studenci w uczelnianych laboratoriach mieliby możliwość wykonywania ćwiczeń laboratoryjnych w nowoczesny i ujednolicony sposób, a pracownicy dydaktyczni zyskaliby narzędzie do szybkiego wykrycia błędów w wykonywanych zadaniach.

2. Cel i zakres pracy

Celem pracy jest automatyzacja i uporządkowanie procesu analizy i przechowywania wyników badań wytrzymałościowych materiałów przeprowadzonych w laboratorium. W tym celu przeprowadzona zostanie analiza sposobu przeprowadzania badań laboratoryjnych, przechowywania wyników uzyskanych z urządzeń pomiarowych oraz metody przeprowadzania analizy. Zidentyfikowane zostaną wymagania procesu technologicznego oraz oczekiwane funkcjonalności.

Przeprowadzony zostanie przegląd dostępnych technologii i wybór narzędzia, które można wykorzystać do realizacji zadania. Następnie zostanie zaimplementowany system obsługi zdolny do wykonywania postawionych zadań, a więc gromadzenia, analizy i archiwizacji wyników badań laboratoryjnych.

Praca obejmuje badania właściwości mechanicznych metali: rozciągania i ściskania przeprowadzonych na maszynie Zwick na różnego rodzaju materiałach, w różnych zakresach prędkości oraz temperatur.

I. Część studialna

1. Opis wykorzystywanych technologii

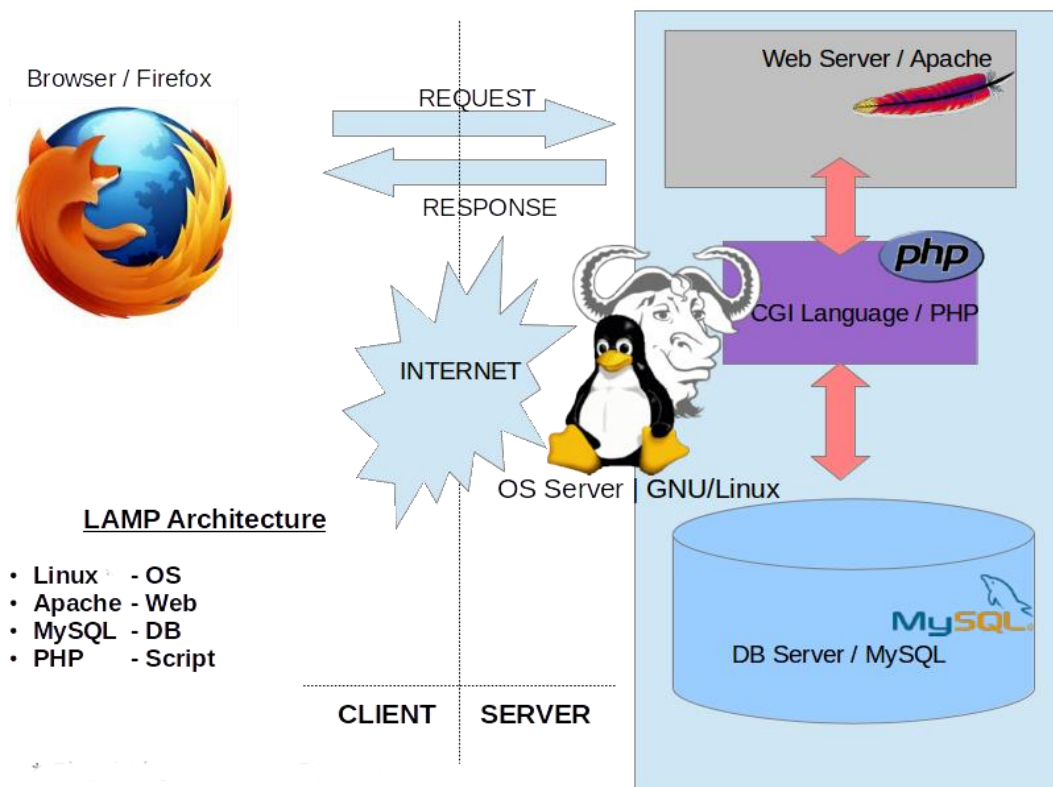
W niniejszym rozdziale omówiono technologie oraz narzędzia, które zostały wykonane w trakcie realizacji pracy, a także opisano statyczną próbę rozciągania i ściskania.

1.1. Platforma LAMP

LAMP jest archetypowym modelem pakietu oprogramowania serwisów WWW. Jego nazwa jest akronimem nazw jego czterech komponentów [1]:

- systemu operacyjnego *Linux*,
- serwera HTTP *Apache*,
- systemu zarządzania relacyjną bazą danych *MySQL*,
- języka skryptowego *PHP*.

Pomimo, że żaden z tych elementów nie został stworzony specjalnie do współdziałania z pozostałymi, taki zestaw oprogramowania jest bardzo popularny ze względu na niski koszt i dostępność wszystkich komponentów. Elementy modelu LAMP są zamienne i nie ograniczają się do pierwotnego wyboru. Pakiet LAMP może być przystosowany do różnych systemów operacyjnych. Rysunek 1 przedstawia zależności między poszczególnymi elementami modelu a klientem, czyli przeglądarką internetową. Serwer bazy danych służy do przechowywania danych w sposób trwały. Interpreter języka PHP wykonuje rozkazy na podstawie informacji z serwera WWW oraz bazy danych. Tylko web serwer komunikuje się z klientem za pośrednictwem Internetu w celu dostarczenia oczekiwanej treści i obsługi żądań. W niniejszej pracy wykorzystany został pakiet XAMPP, będący wieloplatformowym odpowiednikiem LAMP.



Rys. 1. Schemat powiązań między komponentami pakietu LAMP

Źródło: [https://en.wikipedia.org/wiki/LAMP_\(software_bundle\)#/media/File:LAMPP_Architecture.png](https://en.wikipedia.org/wiki/LAMP_(software_bundle)#/media/File:LAMPP_Architecture.png)

1.1.1. System operacyjny

W Internecie dominującymi systemami operacyjnymi wśród serwerów są uniksowe dystrybucje open source, oparte na Linuksie i FreeBSD. Serwery z systemem Windows Server również mają bardzo duży udział. Własne systemy operacyjne, takie jak z/OS i serwer MacOS są również wdrażane, ale w znacznie mniejszej liczbie.

Specjalistyczne systemy operacyjne zorientowane na serwer mają tradycyjnie takie funkcje, jak [2]:

- możliwość rekonfiguracji i aktualizacji zarówno sprzętu, jak i oprogramowania w pewnym zakresie, bez ponownego uruchamiania,
- zaawansowane mechanizmy tworzenia kopii zapasowych umożliwiające regularne i częste przechowywanie krytycznych danych online,
- przejrzysty transfer danych między różnymi wolumenami lub urządzeniami,
- elastyczne i zaawansowane funkcje sieciowe,

- możliwości automatyzacji, takie jak demony w systemie UNIX i usługi w Windows,
- ścisłe bezpieczeństwo systemu z zaawansowaną ochroną użytkowników, zasobów, danych i pamięci,
- zaawansowane wykrywanie i alarmowanie w takich warunkach, jak przegrzanie, awaria procesora i awaria dysku,
- w praktyce wiele systemów operacyjnych na komputerach stacjonarnych i serwerach ma podobne bazy kodowe, różniące się głównie konfiguracją.

1.1.2. Serwer WWW

Serwer internetowy to system komputerowy, który przetwarza żądania za pośrednictwem protokołu HTTP, podstawowego protokołu sieciowego wykorzystywanego do rozpowszechniania informacji w sieci WWW. Termin może odnosić się do całego systemu lub do oprogramowania, które akceptuje i nadzoruje żądania HTTP [3].

1.1.3. System zarządzania relacyjną bazą danych

System zarządzania relacyjnymi bazami danych to system zarządzania bazami danych oparty na modelu relacyjnym. Większość baz danych w powszechnym użyciu opiera się na modelu relacyjnej bazy danych.

Model relacyjny (RM) do zarządzania bazami danych to podejście do zarządzania danymi przy użyciu struktury i języka zgodnego z logiką predykatów pierwszego rzędu, gdzie wszystkie dane reprezentowane w postaci encji, połączone są w relacje. Baza danych zorganizowana pod kątem modelu relacyjnego jest relacyjną bazą danych. W modelu relacyjnym powiązane wpisy są połączone ze sobą "kluczem".

Celem modelu relacyjnego jest dostarczenie metody określania danych i zapytań przy użyciu deklaracji: użytkownicy bezpośrednio określają, jakie informacje zawiera baza danych i jakich informacji od niej chcą, i pozwalają oprogramowaniu zadbać o opisanie struktury danych do przechowywania i procedur odczytywania w celu odpowiedzi na zapytania [4].

Większość relacyjnych baz danych korzysta z definicji danych SQL i języka zapytań. Systemy te wdrażają to, co można uznać za inżynierskie przybliżenie modelu relacyjnego. Tabela w schemacie bazy danych SQL odpowiada zmiennej nazwowej,

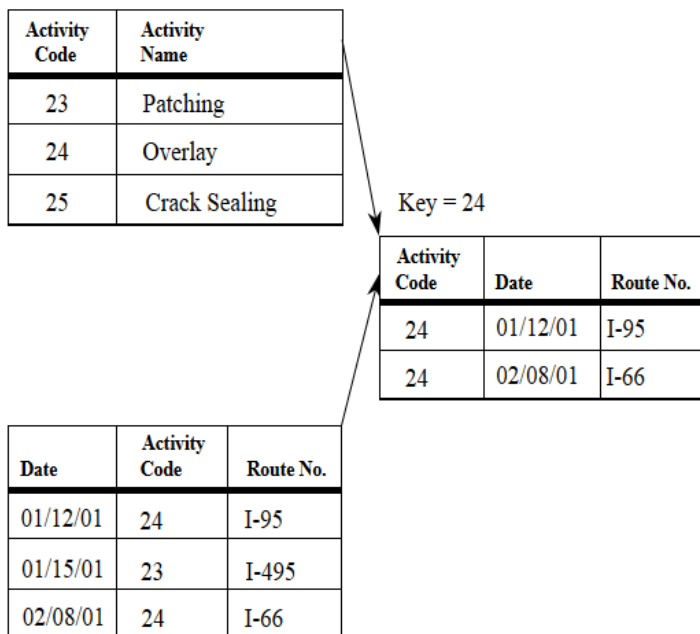
zawartość tabeli odpowiada relacji, kluczowe powiązania i zapytania SQL odpowiadają predykatom. W terminologii dotyczącej relacyjnych baz danych, pojawia się często wiele pojęć z różnych płaszczyzn, modeli. Tabela 1 przedstawia nomenklaturę podstawowych obiektów bazodanowych w różnych terminologiach. Od czystej teorii w pierwszej – do wdrożenia w ostatniej kolumnie.

Tabela 1. Nomenklatura podstawowych obiektów bazodanowych w różnych terminologiach

Teoria relacyjna	Model ER	Relacyjne bazy	Aplikacje
Relacja	Encja	Tabela	Klasa
Krotka	Instancja	Wiersz	Instancja klasy (obiekt)
Atrybut	Atrybut	Kolumna	Właściwość, atrybut
Dziedzina	Dziedzina/Typ	Typ danych	Typ danych

Źródło: <http://www.sqlpedia.pl/relacyjne-bazy-danych-pojecia-podstawowe/>

Rys. 2 przedstawia przykładową bazę danych w modelu relacyjnym. Elementy danych zorganizowano jako zbiór formalnie opisanych tabel, z których można uzyskać dostęp do danych oraz wybrać poszczególne elementy na wiele sposobów, bez konieczności reorganizacji tabel bazy danych. Każda tabela zawiera jedną lub więcej kategorii danych w kolumnach. Każdy wiersz zawiera unikalną instancję danych dla kategorii zdefiniowanych przez kolumny.



Rys. 2. Przykład bazy danych w modelu relacyjnym

Źródło: Data Integration Glossary, US: Department of Transportation, August 2001. Język skryptowy

1.2. Język PHP

Język skryptowy jest językiem programowania obsługującym skrypty: programy napisane dla specjalnego środowiska wykonawczego, które automatyzują wykonywanie zadań. Alternatywnie mogą być wykonywane jeden po drugim przez człowieka. Języki skryptów są często interpretowane (a nie kompilowane). Środowiska, które można zautomatyzować za pomocą skryptów, obejmują aplikacje, strony internetowe w przeglądarce internetowej, korzystanie z powłok systemów operacyjnych (OS), systemów wbudowanych, a także wiele gier. Języki skryptów są czasami określane jako języki programowania wysokiego poziomu, ponieważ działają na wysokim poziomie abstrakcji. [5].

PHP jest językiem skryptowym po stronie serwera przeznaczonym do tworzenia stron internetowych, ale również wykorzystywanym jako język programowania ogólnego przeznaczenia. Implementacja referencyjna PHP jest obecnie produkowana przez *The PHP Group* [6].

Kod PHP może być osadzony w kodzie HTML lub może być używany w połączeniu z różnymi systemami szablonów stron internetowych, systemami zarządzania treścią WWW i ramami internetowymi. Kod PHP jest zwykle przetwarzany przez interpreter PHP zaimplementowany jako moduł na serwerze WWW lub jako plik wykonywalny

Common Gateway Interface (CGI). Serwer łączy w sobie wyniki zinterpretowanego i wykonanego kodu PHP, którym może być dowolny typ danych, w tym obrazy, z wygenerowaną stroną internetową [7].

1.3. Framework do aplikacji webowych

Framework (ang. struktura, fundament, podstawa) do aplikacji webowych jest rodzajem specjalnie zaprojektowanego szkieletu do ułatwienia programistom budowy aplikacji webowych. Frameworki zapewniają podstawową funkcjonalność, wspólną dla większości aplikacji internetowych i gotowe narzędzia ogólnego przeznaczenia, które wyręczają programistę od implementacji powtarzalnych funkcjonalności.

Przykład programu „Hello World!” napisanego w kodzie PHP wewnątrz dokumentu HTML:

```
<!DOCTYPE html>
<html>
<head>
<title>PHP Test</title>
</head>
<body>
<?php echo '<p>Hello World</p>'; ?>
</body>
</html>
```

1.3.1. Symfony

Symfony to zestaw komponentów PHP oraz framework dla aplikacji internetowych. Symfony zawiera 30 autonomicznych bibliotek rozszerzających funkcjonalność języka PHP, które można wykorzystywać niezależnie od Symfony framework do budowy aplikacji PHP. Symfony framework realizuje dwa podstawowe zadania [8]:

- 1) zapewnienie możliwości wyboru komponentów i dodatkowych bibliotek,
- 2) umożliwienie konfiguracji oraz połączenia bibliotek w całość.

Celem frameworka jest zintegrowanie wielu niezależnych narzędzi w jeden spójny interfejs programistyczny. Symfony dostarcza zestaw narzędzi do szybkiego tworzenia aplikacji internetowych, bez narzucania programiście rozwiązań w zakresie funkcjonalności

aplikacji. Zwykły użytkownik może rozpocząć programowanie, stosując określoną dystrybucję Symfony, która dostarcza framework z sensownymi domyślnymi ustawieniami.

Główne cechy Symfony to [9]:

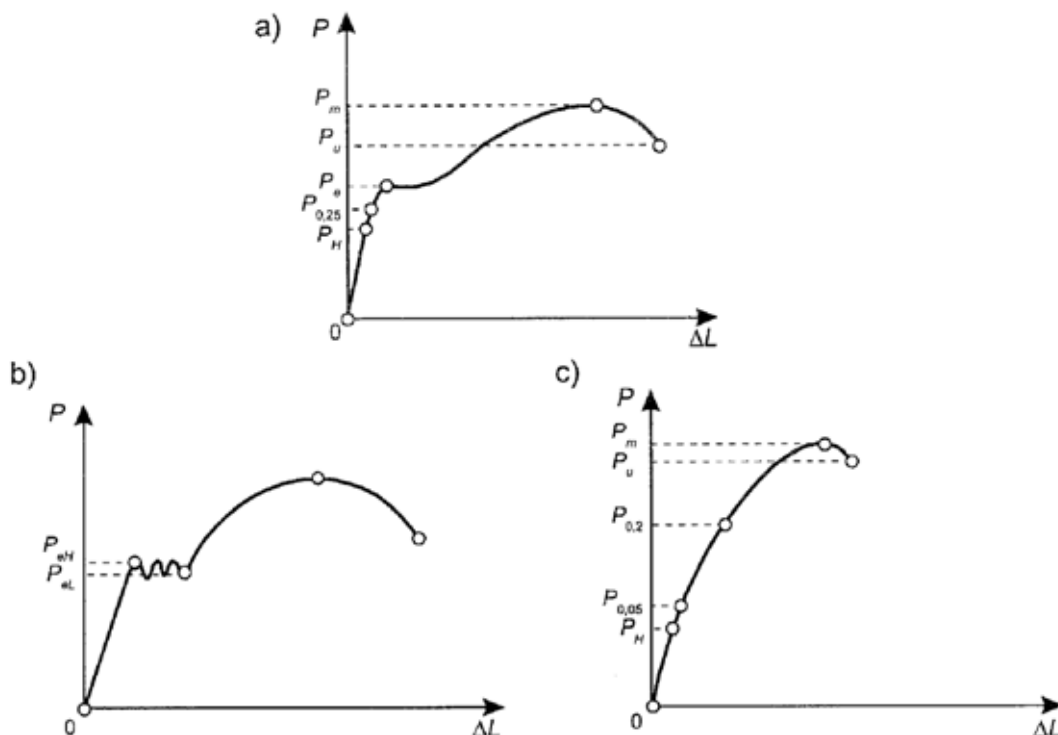
- bazuje na wzorcu projektowym MVC,
- programowanie zorientowane obiektowo,
- łatwość w instalacji oraz konfiguracji na większości platform,
- niezależność od systemu bazodanowego,
- zgodność z najlepszymi standardami oraz wzorcami budowy aplikacji internetowych,
- walidacja formularzy i treści,
- zarządzanie sesjami,
- łatwość rozbudowy oraz możliwość integracji z innymi bibliotekami,
- wykorzystanie technologii scaffoldingu,
- wbudowana możliwość tłumaczenia na wiele języków,
- wbudowana ochrona przed atakami CSRF oraz XSS.

2. Statyczna próba rozciągania

Celem próby rozciągania jest określenie podstawowych parametrów charakteryzujących własności wytrzymałościowe i plastyczne badanych materiałów. Dla każdej próbki wyznacza się obciążenie (siłę), wydłużenie oraz przewężenie odpowiadające poszczególnym etapom rozciągania.

Statyczną próbę rozciągania przeprowadza się w warunkach laboratoryjnych zbliżonych do rzeczywistych warunków pracy elementów maszyn różnego typu, a sposób jej przeprowadzania jest ujęty w normach [10].

Bezpośrednim wynikiem próby rozciągania jest wykres przedstawiający siłę F w funkcji wydłużenia ΔL . Stosowanie układu współrzędnych: siła – wydłużenie bezwzględne, czyli $F - \Delta L$, jest w praktyce niewygodne z tego względu, że porównywanie wyników może się odbywać tylko wtedy, gdy przekroje stosowanych próbek są jednakowe. Dlatego też normalizuje się wykres próby rozciągania ze względu na wymiary próbki. Normalizacja polega na tym, że obciążenie F zastępuje się naprężeniem nominalnym ($\sigma_n = F / S_0$). Rysunek 3 przedstawia przebieg wykresów dla trzech rodzajów materiałów: wykazujących wyraźną granicę plastyczności, górną i dolną granicę plastyczności oraz bez wyraźnej granicy plastyczności.



Rys. 3. Wykresy rozciągania dla materiałów:
a) wykazujących wyraźną granicę plastyczności,
b) wykazujących górną i dolną granicę plastyczności,
c) bez wyraźnej granicy plastyczności

Źródło: http://home.agh.edu.pl/~jakubjac/cwlabwm.htm#_Toc32678819

W wyniku statycznej próby rozciągania uzyskuje się wskaźniki wytrzymałości i plastyczności.

2.1. Wskaźniki wytrzymałości

Wskaźniki wytrzymałościowe oblicza się przy wykorzystaniu wzoru na napężenie σ będące ilorazem siły w dowolnej chwili badania i początkowej powierzchni przekroju poprzecznego S_0 próbki:

$$\sigma = \frac{F}{S_0} \left[\frac{N}{mm^2} \right]. \quad (1)$$

Do wskaźników wytrzymałościowych uzyskanych ze statycznej próby rozciągania zaliczamy:

- 1) Wytrzymałość na rozciąganie, R_m

$$R_m = \frac{F_m}{S_0} \left[\frac{N}{mm^2} \right]. \quad (2)$$

- 2) Wyraźną granicę plastyczności R_e
- a) Górną granicę plastyczności R_{eH}

$$R_{eH} = \frac{F_{eH}}{S_0} \left[\frac{N}{mm^2} \right]. \quad (3)$$

- b) Dolną granicę plastyczności R_{eL}

$$R_{eL} = \frac{F_{eL}}{S_0} \left[\frac{N}{mm^2} \right]. \quad (4)$$

- 3) Umowną granicę plastyczności R_p – naprężenie graniczne określone podczas trwania próby, powodujące nieproporcjonalny przyrost wydłużenia równy umownemu procentowi początkowej długości pomiarowej próbki L_0 lub ekstensometru L_e . Symbol tej wielkości uzupełnia się wskaźnikiem określającym umowny procent przyrostu początkowej długości pomiarowej próbki lub ekstensometru, np. $R_{p0,2}$
- 4) Naprężenie graniczne R_r przy wydłużeniu trwałym – naprężenie przy którym po zdjęciu siły wydłużenie trwałe początkowej długości pomiarowej próbki L_3 lub przyrost trwały długości pomiarowej ekstensometru L_e jest równe umownej wartości. Symbol wielkości uzupełnia się wskaźnikiem określającym umowny procent trwałego wydłużenia lub przyrostu, np. $R_{r0,2}$. Naprężenie graniczne R_t przy przyroście całkowitym – naprężenie określone przy przyroście całkowitym, równym umownemu procentowi początkowej długości pomiarowej próbki L_0 lub ekstensometru L_e . Symbol wielkości uzupełnia się wskaźnikiem określającym umowny procent przyrostu początkowej długości pomiarowej próbki lub ekstensometru, np. $R_{t0,2}$

2.2. Wskaźniki plastyczności

Wskaźniki plastyczności w statycznej próbie rozciągania oblicza się na podstawie zmiany wymiarów próbki podczas próby, po jej odciążeniu lub rozerwaniu korzystając ze wzoru:

$$A = \frac{l - l_0}{l_0} \cdot 100 [\%]. \quad (5)$$

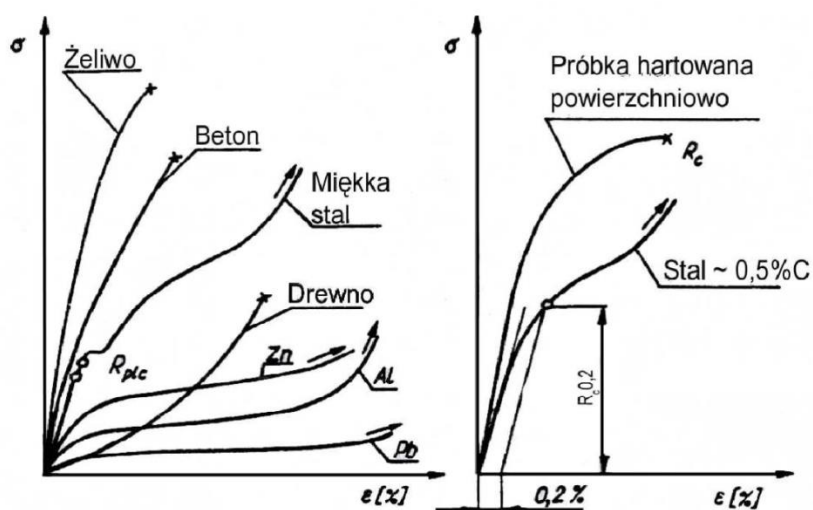
Do wskaźników plastyczności należą:

- 1) wydłużenie nieproporcjonalne procentowe A_g przy największej sile,
- 2) wydłużenie całkowite procentowe A_{gt} ,
- 3) wydłużenie procentowe A_t przy rozerwaniu.

3. Statyczna próba ściskania

Próba ściskania jest „odwrotnością” próby rozciągania – wykres ściskania niektórych metali jest symetryczny do wykresu rozciągania w zakresie ujemnych naprężeń i odkształceń. Jest to powodem istnienia jednoosiowego stanu naprężenia w próbce ściskanej i rozciąganej. Stąd też, wszystkie własności mechaniczne uzyskuje się w analogiczny sposób jak przy próbie rozciągania.

Próbka ściskana musi mieć odpowiednie wymiary długości do przekroju, w przeciwnym wypadku może dojść do wyboczenia. Próba musi być tak przeprowadzona, aby uzyskać osiowe ściskanie, a płaszczyzny ściskające próbkę (uchwyty maszyny wytrzymałościowej) mogły ustawić się równoległe do płaszczyzn podstaw próbki. W sąsiedztwie uchwytów poprzeczne wymiary próbki nie mogą się swobodnie zwiększać. Wywierają tu wpływ siły tarcia na powierzchni styku próbki z uchwytami [11]. Wpływ ten powoduje, że wartość naprężeń nie odpowiada założeniom o jednostajnym rozkładzie naprężeń w materiale. Przy zlikwidowaniu tarcia (również poprzez smarowanie powierzchni uchwytów) dla próbek kruchych powstają złomy rozdzielcze równoległe do kierunku działania siły.



Rys. 4. Typowe charakterystyki naprężenie - odkształcenie uzyskane z różnych typów materiałów

Źródło: <http://www.biomech.pwr.wroc.pl/wp-content/uploads/2016/09/pdcw4.pdf>

W wyniku statycznej próby ściskania uzyskuje się:

- 1) skrócenie przy zniszczeniu, $dL_{(kor.)}$,
- 2) skrócenie przy zniszczeniu, $dL_{(plast.)}$,

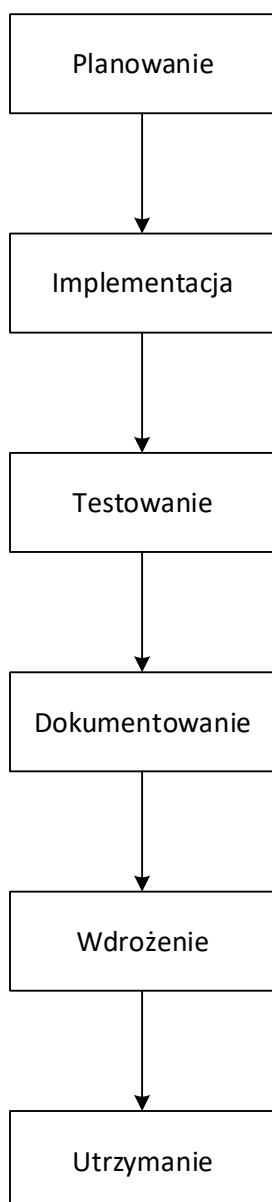
- 3) moduł Younga, E_{mod} ,
- 4) siłę przy plastycznym odkształceniu, F ,
- 5) maksymalną siłę, F_{max} ,
- 6) skrócenie przy F_{max} , $dL_{(\text{max.})}$.

II. Część projektowa

Wykonanie aplikacji do przechowywania i analizy wyników badań laboratoryjnych

1. Przedmiot opracowania

Tematem pracy dyplomowej jest „Projekt i implementacja aplikacji webowej do przechowywania i analizy wyników badań laboratoryjnych”. Aplikacja opracowana jest na podstawie badań wykonanych w laboratorium na maszynach wytrzymałościowych Zwick. Założenia do budowy aplikacji powstały w oparciu o dane wyjściowe z badań wykonanych na maszynie Zwick oraz oczekiwanej funkcjonalności. Aplikacja została stworzona według klasycznego modelu tworzenia oprogramowania, który przedstawiono na rysunku 5:



Rys. 5. Schemat procesu produkcji oprogramowania

1.1. Ogólny opis

Aplikacja będzie umożliwiała utworzenie konta użytkownika. Użytkownik, po zalogowaniu, będzie miał możliwość dodania danych za pomocą formularzy oraz wczytania plików z danymi. Na podstawie danych wejściowych wykonane zostaną obliczenia, na podstawie których wygenerowany zostanie raport zawierający wyniki w postaci liczbowej, tabelarycznej oraz wykresu. Aplikacja będzie oferowała możliwość archiwizacji danych oraz zarządzania nimi poprzez możliwość edycji i usunięcia.

1.1.1. Funkcje programu

Opracowany program będzie posiadał następujące funkcjonalności:

1. Dodawanie nowych zadań,
2. Modyfikacja istniejących zadań,
3. Dodanie informacji o materiałach,
4. Określenie maszyny wytrzymałościowej,
5. Przechowywanie informacji na temat wyników eksperymentów:
 - jaki materiał był testowany: nazwa stopu, skład chemiczny
 - dla danego materiału, informacje o wykonywanych badaniach:
 - jaka maszyna wytrzymałościowa,
 - parametry próbki: kształt, wymiary,
 - rodzaj próby (rozciąganie, ściskanie, itp.),
 - parametry eksperymentu: temperatura, prędkość.
 - Dla danej próby:
 - możliwość zapisania plików z maszyn wytrzymałościowych w formacie PDF oraz CSV,
 - wyniki testów z maszyny wytrzymałościowej (Moduł Younga, granica plastyczności, granica wytrzymałości, itp.).
 - zdjęcia próbki przed i po eksperymencie
6. Analiza wprowadzonych danych:
 - obliczenie naprężenia z eksperymentu,
 - obliczenie odkształcenia,
 - wykreślenie krzywej umocnienia.

1.1.2. Charakterystyka użytkowników

Każdy użytkownik posiada własne konto, zawierające login, adres e-mail i hasło. Użytkownik ma możliwość dodania wyników eksperymentów, edycji i usunięcia oraz przeglądania wcześniej wprowadzonych danych i wygenerowanych raportów. Użytkownik nie ma dostępu do danych wprowadzonych przez innego użytkownika.

Istnieje jeden użytkownik z uprawnieniami administratora, który ma możliwość korzystania z panelu administratora, na którym może przeglądać listę użytkowników, weryfikować ich konta lub usuwać je. Administrator ma możliwość przeglądania wszystkich danych w systemie.

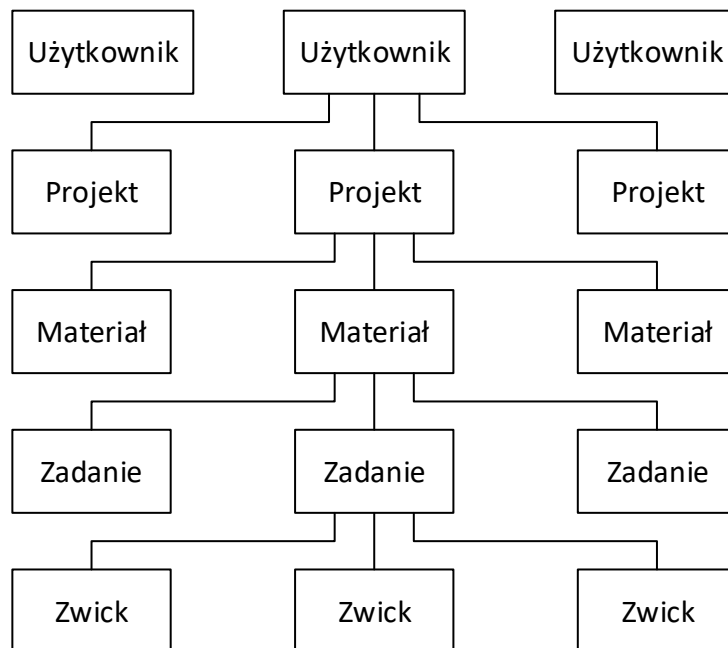
1.1.3. Ogólne ograniczenia

Przyjęto następujące założenia dotyczące dostępu do aplikacji:

- dostęp do aplikacji z dowolnego miejsca z dostępem do Internetu
- pełna kompatybilność z przeglądarkami Google Chrome (wersja 63), Mozilla Firefox (wersja 57 „Quantum”) oraz Microsoft Edge (wersja 41)
- możliwość korzystania z aplikacji na urządzeniach mobilnych
- możliwość pobrania zapisanych danych
- na jednym materiale można wykonywać wiele badań: ten sam materiał może być poddany badaniom na różnych maszynach wytrzymałościowych, może zostać poddany ścisaniu i rozciąganiu, może być poddany testom w różnych temperaturach i prędkościach

1.2. Szczegółowe wymagania

W celu spełnienia wszystkich postawionych wymagań przy zadanych ograniczeniach jako strukturę działania programu przyjęto model oparty na zadaniach. Użytkownik w pierwszym kroku tworzy projekt, następnie dodaje materiał, a w kolejnym kroku zadania. Następnie wybierany jest jeden z wcześniej dodanych materiałów, maszyna, na której wykonywano badanie, rodzaj próby, parametry próbki, wyniki eksperymentów oraz pliki z danymi. Zależności między poszczególnymi encjami przedstawia poniższy diagram (rysunek 6):



Rys. 6. Zależności między encjami bazy danych

Zawartość poszczególnych encji jest przedstawiona poniżej:

<p>Użytkownicy</p> <ul style="list-style-type: none">– Nazwa użytkownika,– Hasło,– Adres e-mail.
<p>Projekty</p> <ul style="list-style-type: none">– Nazwa projektu,– Użytkownik.
<p>Materiały</p> <ul style="list-style-type: none">– Nazwa stopu,– Skład chemiczny,– Projekt.

Zadania

- Nazwa zadania,
- Maszyna wytrzymałościowa,
- Kształt próbki (walcowy, wiosełka),
- Rodzaj próby (rozciąganie / ściskanie),
- d_0 – Średnica próbki,
- L_0 – Wysokość próbki,
- t_0 – Temperatura próby,
- v_0 – Prędkość próby,
- Materiał.

Zwick

- A_g – Wydłużenie / przewężenie,
- A_{gt} – Wydłużenie / przewężenie,
- A_t – Wydłużenie / przewężenie,
- E – Moduł Younga,
- $R_{p0.2}$ – Granica plastyczności,
- R_B – Granica wytrzymałości,
- F_m – Maksymalna siła,
- R_m – Naprężenie przy zerwaniu,
- Dane wejściowe – plik w formacie CSV,
- Raport – plik w formacie PDF,
- Zadanie.

ZwickData

- Czas badania,
- Droga standardowa,
- Siła standardowa,
- S – Pole przekroju poprzecznego,
- ε – odkształcenie,
- Zwick.

1.3. Wykonanie obliczeń

Po wprowadzeniu do bazy danych informacji o wykonanym eksperymencie możliwe jest wykonanie obliczeń, dzięki którym możemy otrzymać krzywą umocnienia danego materiału w zadanych warunkach.

1.3.1. Obliczenie pola przekroju poprzecznego próbki walcowej S

$$S = \pi \cdot \left(\frac{d_0}{2}\right)^2 \cdot \frac{h_0}{h_0 - \text{droga standardowa}}, \quad (6)$$

gdzie:

- d_0 – średnica próbki [mm],
- h_0 – wysokość próbki [mm],
- *droga standardowa* – wartość odczytana z pliku danych wejściowych [mm].

1.3.2. Obliczenie naprężenia z eksperymentu S_{exp}

$$S_{exp} = \frac{\text{Siła standardowa}}{S}, \quad (7)$$

gdzie:

- Siła standardowa – wartość odczytana z pliku danych wejściowych [N],
- S – pole przekroju poprzecznego próbki.

1.3.3. Obliczenie odkształcenia ε

$$\varepsilon = \ln \left(\frac{h_0}{droga\ standardowa} \right), \quad (8)$$

gdzie:

- h_0 – wysokość próbki,
- *droga standardowa* – wartość odczytana z pliku danych wejściowych [mm].

2. Implementacja

W celu realizacji przyjętych wymagań zastosowano następujące rozwiązania:

- Platforma: Aplikacja webowa zainstalowana na serwerze HTTP z obsługą skryptów PHP,
- Wersja PHP: 5.6.24, Zend Engine: 2.6.0, Xdebug: 2.4.1,
- Framework: Symfony 3.4.3,
- Baza danych: SQL MariaDB 10.1.16,
- Front end framework: Bootstrap 3,
- Szablon: SB Admin 2.

Tabela 2. Zestawienie najważniejszych komponentów wykorzystanych w projekcie

Komponent	Wersja	Cel użycia
doctrine/orm	v2.5.14	Model obiektowo-relacyjny bazy danych
friendsofsymfony/user-bundle	2.0.x-dev c617ec7	Obsługa kont użytkowników
knplabs/knp-menu	2.3.0	Dynamicznie generowane menu
symfony/symfony	v3.4.3	Symfony Framework
twig/twig	v1.35.0	Silnik szablonów stron PHP

Funkcjonalność aplikacji po stronie serwera zaimplementowano korzystając z języka HTML 5, stylów CSS 3 oraz skryptów JavaScript. Rozszerzenia TWIG pozwalały prezentować funkcjonalność serwera w przeglądarce klienta poprzez generowanie treści stron internetowych w zależności od zawartości bazy danych i ustawień użytkownika.

Zestawienie ważniejszych bibliotek JavaScript wykorzystanych w pracy:

Tabela 3. Zestawienie ważniejszych bibliotek JavaScript

Komponent	Wersja	Cel użycia
Bootstrap	3.3.7	HTML, CSS, JS framework do tworzenia responsywnych stron internetowych

jQuery	3.1	Manipulacja modelem obiekowym strony internetowej
metisMenu	1.1.3	Animowane, wielopoziomowe menu
jQuery flot	0.8.3	Tworzenie responsywnych wykresów
DataTables	1.10.12	Wyświetlanie tabel zawierających duże ilości pozycji

Implementację kodu przeprowadzono etapami. Oznacza to, że w pierwszej kolejności rozpoczęto realizację najważniejszych funkcji, a następnie wprowadzano kolejne do już funkcjonalnej aplikacji.

2.1. Wprowadzanie danych i analiza wyników eksperymentów

wykonanych na maszynie Zwick

2.1.1. Klasa Zwick

Pierwszym zadaniem było utworzenie struktury pozwalającej na wprowadzenie wszystkich wyników eksperymentu przeprowadzonego na maszynie wytrzymałościowej Zwick. W tym celu utworzono Klasę Zwick, która zawierała pola odpowiadające parametrom badania:

```
// src/AppBundle/Entity/Zwick.php

namespace AppBundle\Entity;

class Zwick
{
    $taskName    – nazwa zadania,
    $shape       – kształt próbki,
    $type        – typ próby (rozciąganie / ściskanie),
    $d0          – średnica próbki,
    $h0          – wysokość próbki,
    $t0          – temperatura próby,
    $v0          – prędkość rozciągania / ściskania,
    $ag          – przewężenie / wydłużenie,
    $agt         – przewężenie / wydłużenie,
    $at          – przewężenie / wydłużenie,
    $e           – moduł Younga,
    $r           – granica plastyczności,
    $fm          – maksymalna siła,
    $rm          – granica wytrzymałości,
    $rb         – naprężenie przy zerwaniu,
```

```
$fileTra      – nazwa pliku z rozszerzeniem TRA,  
$filePdf      – nazwa pliku z rozszerzeniem PDF.  
}
```

Ponieważ na tym etapie nie przewidziano możliwości dodawania wyników eksperymentów z maszyny Gleeble 3800, atrybuty encji „Zadanie” oraz „Zwick” umieszczono w jednej klasie.

Wszystkie pola klasy enkapsulowano, a więc dostęp do nich możliwy jest za pomocą *getterów* i *setterów*:

```
private $taskName;  
  
public function getTaskName()  
{  
    return $this->taskName;  
}  
  
public function setTaskName($taskName)  
{  
    $this->taskName = $taskName;  
}
```

Aby trwale przechować dane zapisane w określonym obiekcie klasy, wykorzystano bibliotekę *Doctrine ORM*, która umożliwia zmapowanie obiektowego modelu danych przedstawionego poniżej, do modelu relacyjnego bazy danych. Instrukcje do zmapowania pól klasy umieszczono w adnotacji, bezpośrednio nad opisującym je elementem.

Poniżej przedstawiono sposób powiązania tabeli w bazie danych SQL z klasą PHP. Adnotacja *@ORM\Entity* służy do określenia repozytorium, które ułatwia wykonywanie operacji *CRUD* (ang. *Create, Read, Update, Delete*) na bazie danych w skryptach PHP. Adnotacja *@ORM\Table* określa nazwę tabeli w modelu relacyjnym, która będzie odpowiadała poniższej klasie. W tym przypadku tabela *zwick* w bazie danych będzie odpowiadała klasie „Zwick” w modelu obiektowym.

```
use Doctrine\ORM\Mapping as ORM;  
  
/**  
 * @ORM\Entity(repositoryClass="AppBundle\Repository\ZwickRepository")  
 * @ORM\Table(name="zwick")  
 */  
class Zwick
```

Relacyjną reprezentacją pola przechowującego wartość liczbową jest kolumna tabeli z określonym zmiennoprzecinkowym typem danych (*float*):

```
/**
 * @ORM\Column(type="float")
 */
private $d0;
```

W celu przechowania wartości kształtu materiału oraz rodzaju próby wykorzystano binarny typ zmiennej

```
/**
 * false: cylinder, true: paddle
 * @ORM\Column(type="boolean")
 */
private $shape;
/**
 * false: compression, true: tension
 * @ORM\Column(type="boolean")
 */
private $type;
```

2.1.2. Klasa ZwickData

W celu przechowywania wartości z pliku CSV (ang. *Comma Separated Value*) stworzono klasę *ZwickData*, która odpowiada pojedynczemu wierszowi danych. Klasę rozszerzono o pola do przechowywania wyników obliczeń wykonywanych na danych wejściowych:

```
class ZwickData
{
    $test_time;           – Czas badania,
    $distance_standard;   – Droga standardowa,
    $load_measurement;    – Siła standardowa,
    $$;                   – Pole przekroju poprzecznego próbki,
    $Eps;                 – Odkształcenie,
    $Sexp;                – Naprężenie z eksperymentu.
}
```

Podobnie jak w klasie *Zwick*, pola enkapsulowano i dopisano odpowiednie adnotacje. W celu umożliwienia zapisania danych odczytanych z pliku o formacie rozdzielonym przecinkami, należało ustawić parametr „*nullable=true*” dla pól \$\$, \$Eps oraz \$Sexp, ponieważ przed wykonaniem obliczeń nie przechowywały żadnych danych:

```
/**
 * @ORM\Column(type="float", nullable=true)
 */
private $$;
```


Aby zwi za c poszczególne odczyty pochodz ce z jednego eksperymentu, Encje Zwick oraz *ZwickData* po łączono relacj  „jeden do wielu”. W tym celu w klasie Zwick dodano pole *\$data*, a w klasie *ZwickData* *\$zwick* oraz dodano adnotacje.

W klasie Zwick, parametr „*targetEntity*” atrybutu *@OneToMany* okre la docelow  klas  relacji, a „*mappedBy*” oznacza nazw  pola klasy *ZwickData* przeznaczonego do zmapowania relacji mi dzy klasami. W tym przypadku jest to pole *\$zwick*. Z kolei w klasie *ZwickData*, mamy adnotacj  *@ManyToOne*, w kt rym parametr „*targetEntity*” wskazuje na klas  Zwick, a „*inversedBy*”, b d cy odpowiednikiem parametru „*mappedBy*” w adnotacji *@OneToMany*, wskazuje na pole *\$data*. Adnotacja *@JoinColumn* zawsze towarzyszy *@ManyToOne* i jest potrzebna do okre lenia kolumny, w kt rej znajduje si  klucz obcy (parametr „*name*”) oraz do ustawienia nazwy kolumny zawieraj cej klucz g wny w klasie Zwick (parametr „*referencedColumnName*”).

```
// src/AppBundle/Entity/Zwick.php

use Doctrine\ORM\Mapping\OneToMany;

/**
 * @OneToMany(targetEntity="AppBundle\Entity\ZwickData",
 *     mappedBy="zwick",
 *     cascade={"remove"})
 */
private $data;

// src/AppBundle/Entity/Zwick.php

use Doctrine\ORM\Mapping\JoinColumn;
use Doctrine\ORM\Mapping\ManyToOne;

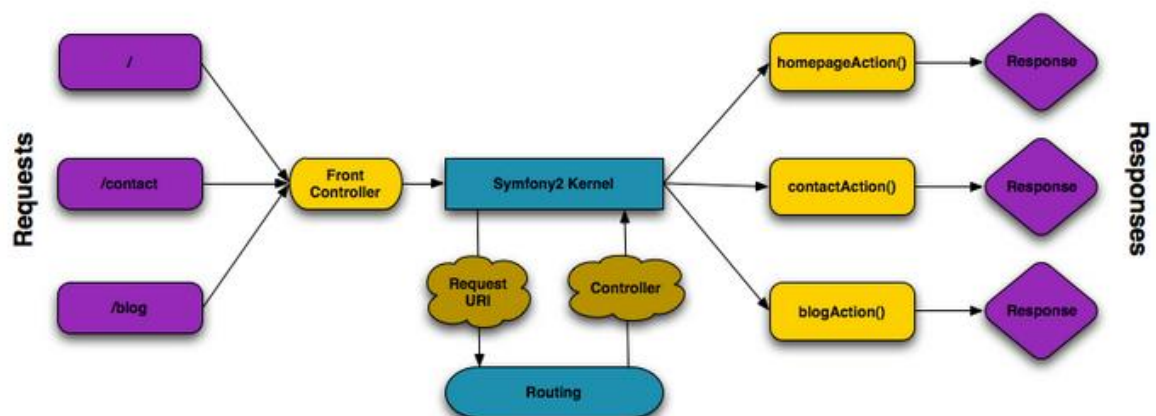
/**
 * @ManyToOne(targetEntity="AppBundle\Entity\Zwick", inversedBy="data")
 * @JoinColumn(name="id_input", referencedColumnName="id")
 */
private $zwick;
```

2.1.3. Kontroler

Po utworzeniu encji, zaimplementowano operacje, jakie maj  by  na nich wykonywane. Do realizacji zada n zwi zanych z tworzeniem nowych encji, modyfikacji, wy wietlania i usuwania istniej cych w Symfony s u y kontroler. Kontroler Zwick (*ZwickController*) sk ada si  z nast puj cych akcji:

- newAction,
- showAction,
- deleteAction,
- editAction,
- pdfAction,
- traAction.

Podobnie jak w przypadku encji, elementy kontrolerów konfigurowane zostały przy użyciu adnotacji. Najważniejszą adnotacją w kontrolerze jest `@Route`, służąca do konfiguracji trasowania (ang. *routing*), a więc adresu URL, pod którym zlokalizowana jest dana akcja. W ten sposób klient może się komunikować z serwerem. Wywołując odpowiedni adres strony internetowej, wywoływana jest przypisana do niego akcja. Na Rys. 7 przedstawiono schemat działania trasowania. Po wysłaniu zapytania, komponent Symfony „Routing” wyszukuje akcję ukrytą pod adresem i wywołuje ją. Każda akcja zwraca odpowiedź, która jest wysyłana z powrotem do klienta.

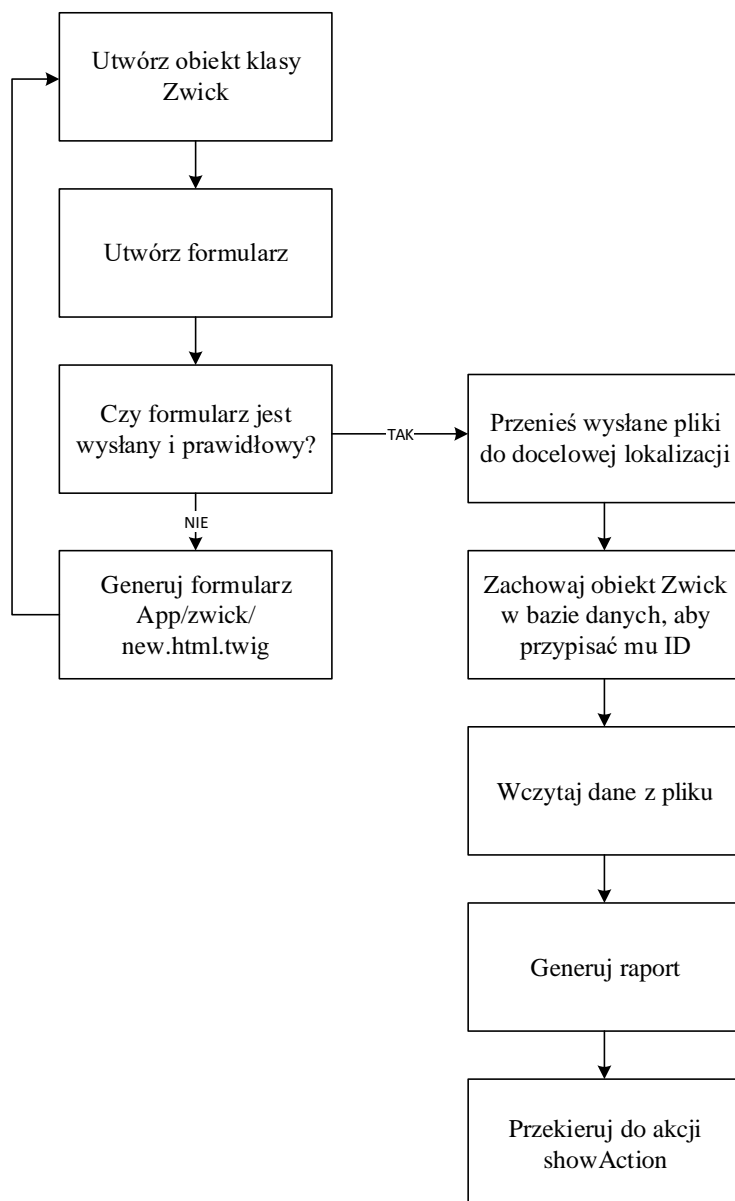


Rys. 7. Schemat działania warstwy trasowania.

Źródło: <http://symfony2-docs-pl.readthedocs.io/pl/latest/book/routing.html>

Kiedy w odpowiedzi na akcję otrzymuje się formularz, metoda wywoływana jest 2 razy: pierwszy raz po to, aby wygenerować formularz, a drugi, aby odczytać z niego dane i zapisać. Takie akcje w kontrolerze *Zwick* to: *new* oraz *edit*. Metoda *newAction* służy do tworzenia nowego zadania. Schemat działania akcji *new* przedstawiono na Rys. 8. Na początku tworzony jest obiekt klasy *Zwick* oraz formularz. Następnie sprawdzane jest, czy formularz został wysłany i czy nie zawiera błędów. Za pierwszym razem warunek nie zostanie spełniony i wygenerowana zostanie strona z formularzem. Za drugim razem, gdy wprowadzone w formularzu wartości nie będą zawierały błędów, wysłane pliki zostaną

przeniesione z lokalizacji tymczasowej, do docelowej lokalizacji na serwerze, dane zostaną wczytane z pliku, zostanie wygenerowany raport oraz nastąpi wywołana akcja „*show*”, która wyświetli wyniki przeprowadzonych operacji.



Rys. 8. Diagram przedstawiający operacje wykonywane w metodzie *newAction*.

2.1.4. Odczytywanie danych z plików

Obsługa plików wymagała odpowiedniej konfiguracji modelu obiektowo-relacyjnego a także wykorzystania biblioteki frameworku Symfony do obsługi plików i wykonywania operacji na systemie plików. W celu odczytania danych z pliku CSV, potrzebna była biblioteka do normalizacji obiektu, serializacji oraz enkoder CSV.

W klasie `Zwick` pola `$fileTra` oraz `$filePdf` skonfigurowano dopisując odpowiednie asercje w adnotacjach. Wymagane jest wykorzystanie zmiennej typu „string”. `@Assert\NotBlank` sprawdza, czy plik został wybrany przed wysłaniem formularza. Jeżeli warunek nie jest spełniony, wyświetlana jest wprowadzona w parametrze „message” wiadomość. `@Assert\File` sprawdza zgodność typu pliku, bez względu na jego rozszerzenie. W przypadku `$fileTra`, plik jest typu tekstowego („text/plain”), a w przypadku `$filePDF` jest to format PDF. Gdy nastąpi próba wysłania pliku o innym formacie, niż wskazany, aplikacja zwróci błąd.

```
// src/AppBundle/Entity/Zwick.php

/**
 * @ORM\Column(type="string")
 * @Assert\NotBlank(message="Proszę wybrać plik z rozszerzeniem TRA")
 * @Assert\File(mimeTypes={ "text/plain" })
 */
private $fileTra;

/**
 * @ORM\Column(type="string")
 * @Assert\NotBlank(message="Proszę wybrać plik PDF")
 * @Assert\File(mimeTypes={ "application/pdf" })
 */
private $filePdf;
```

W kontrolerze, najpierw zmieniono nazwę pliku na losową, z rozszerzeniem odpowiadającym rzeczywistemu typowi danych. Następnie przeniesiono go do odpowiedniej lokalizacji na serwerze. W programie lokalizacja ta jest ustawiona za pomocą globalnego parametru „csv_directory” w pliku konfiguracyjnym `app/config.yml`. Analogiczne operacje przeprowadzono dla pliku PDF.

```
// src/AppBundle/Controllers/ZwickController.php

use Symfony\Component\HttpFoundation\File as File;

public function newAction(Request $request)
{
    ...
    if ($form->isSubmitted() && $form->isValid()) {
        ...

        /** @var File\UploadedFile $file */
        $file = $input->getFileTra();
        $input->setFilePdf($fileName);
        $fileName = md5(uniqid()).'.'.$file->guessExtension();
        $file->move(
            $this->getParameter('csv_directory'),
            $fileName
        );
    }
}
```

```
);
$input->setFileTra($fileName);
...
```

Odczytanie danych z pliku i zapisanie ich do encji wykonano na dwa sposoby. Pierwotnie wykorzystano natywną metodę MySQL / MariaDB do szybkiego i efektywnego wczytywania danych z pliku w formacie CSV.

```
LOAD DATA INFILE :filename into table zwick_data
fields terminated by ','
lines terminated by '\r\n'
ignore 1 lines
(test_time, distance_standard, load_measurement)
```

Metoda *LOAD DATA INFILE* nie jest zaimplementowana w Doctrine ORM, więc komendę wprowadzono wpisując ją bezpośrednio w języku SQL. Metoda ta jest polecana ze względu na optymalny sposób wczytywania danych z pliku do bazy danych, natomiast wadą jej jest niepopularność, przez co nie jest oferowana przez niektórych dostawców serwerów baz danych. Drugim problemem jest konieczność wykonania dodatkowej operacji wczytywania tak umieszczonych danych w bazie z powrotem do modelu obiektowego, co wiąże się z dodatkowym narzutem.

W celu zapewnienia kompatybilności aplikacji z bazą danych SQLite 3, zrezygnowano z tego rozwiązania i odczytywanie danych z pliku CSV wykonano korzystając z bibliotek *Symfony Serializer*, *ObjectNormalizer* oraz *CsvEncoder*. Metodą *decode* odczytano zawartość pliku i umieszczono go w tabeli dwuwymiarowej. Pierwszy wymiar odpowiadał numerowi wiersza pliku liczonego od 0, z pominięciem wiersza nagłówka. Drugi wymiar zawierał kolejne wartości wiersza oddzielone przecinkami. Tak odczytane dane zapisywano do kolejnych encji zbioru *ZwickData*:

```
use Symfony\Component\Serializer\Encoder\CsvEncoder;
use Symfony\Component\Serializer\Normalizer\ObjectNormalizer;
use Symfony\Component\Serializer\Serializer;

$serializer = new Serializer([new ObjectNormalizer()], [new
CsvEncoder()]);
$data = $serializer->decode(file_get_contents(
$this->getParameter('csv_directory').'/'.$input->getFileTra()), 'csv');
$inputData = [];
foreach ($data as $row) {
    $dataRow = new ZwickData();
    $dataRow->setZwick($input);
    $dataRow->setTestTime(reset($row) );
    $dataRow->setDistanceStandard(next($row) );
    $dataRow->setLoadMeasurement(next($row) );
    array_push($inputData, $dataRow);
}
```

```
$em->persist($dataRow);
}
```

Akcje *pdf* i *tra* obsługują możliwość pobrania plików przez użytkownika. W pierwszym kroku pobiera się binarną reprezentację pliku:

```
$response = new BinaryFileResponse($this->
    getParameter('pdf_directory').'/'.$pdfFilename);
```

Następnie ustawia się wartość nagłówek odpowiedzi serwera na metodę GET:

```
$response = new BinaryFileResponse($this->
    getParameter('pdf_directory').'/'.$pdfFilename);
```

W kolejnym kroku określa się nazwę pliku oraz sposób, w jaki sposób ma być zwrócona zawartość pliku. „DISPOSITION INLINE” oznacza, że plik zostanie wyświetlony bezpośrednio w przeglądarce. „DISPOSITION_ATTACHMENT” używa się w celu pobrania pliku na komputer odbiorcy i taki parametr wykorzystano w przypadku akcji „*traAction*”.

```
use Symfony\Component\Filesystem\Filesystem;
```

```
$response->setContentDisposition(
    ResponseHeaderBag::DISPOSITION_INLINE,
    "report.pdf"
);
```

Podczas wywołania akcji „*deleteAction*” służącej do usuwania zadania, następuje także usunięcie plików z serwera. W tym celu wykorzystano metodę *remove* biblioteki *FileSystem*:

```
$fs = new Filesystem();
$fs->remove($this->getParameter('csv_directory').'/'.$zwick->
    getFileTra());
```

2.1.5. Generowanie raportu

Generowanie raportu realizowane jest poprzez wywołanie metody *generateReport*. Metoda wywołuje komendę do i zapisuje do bazy danych wyniki obliczeń. Same obliczenia natomiast napisano w odrębnej klasie *ZwickCalculations*. Poniżej przedstawiono implementację w języku PHP równań opisanych w punkcie 1.3:

Odczytanie danych:

```
$d0 = $this->zwick->getD0();
$h0 = $this->zwick->getH0();
```

```
$this->time[$i] = $next_data_row->getTestTime();
$this->distance_standard[$i] = $next_data_row->getDistanceStandard();
$this->load_measurement[$i] = $next_data_row->getLoadMeasurement();
```

gdzie:

- `$d0` – średnica próbki [mm],
- `$h0` – wysokość próbki [mm],
- `$this->distance_standard[$i]` – droga standardowa [mm],
- `$this->load_measurement[$i]` – siła standardowa [mm],
- `$i` – i-ta encja zbioru ZwickData.

Obliczenie pola przekroju poprzecznego próbki walcowej S

```
$this->S[$i] = pi() * pow($d0,2) * $h0 / (4 * ($h0 - $this->distance_standard[$i]));
```

Obliczenie naprężenia z eksperymentu S_{exp}

```
$this->Sexp[$i] = $this->load_measurement[$i] / $this->S[$i];
```

Obliczenie odkształcenia ε

```
$this->Eps[$i] = log($h0 / ($h0 - $this->distance_standard[$i]));
```

Zapisanie wyników do ZwickData

```
$data_row->setS($this->S[$i]);
$data_row->setEps($this->Eps[$i]);
$data_row->setSexp($this->Sexp[$i]);
```

Taka budowa aplikacji pozwalała na wykonywanie przy użyciu programu podstawowego zadania, do którego jest przeznaczony, a więc przechowywania i analizy wyników badań laboratoryjnych.

2.2. Dodawanie materiałów i przypisywanie ich do zadań

W drugim kroku iteracyjnym wprowadzono możliwość dodawania materiałów. Zgodnie z określonymi wymaganiami jeden materiał może być poddany badaniom na różnych maszynach oraz może być poddany na tej same maszynie różnego rodzaju próbom, tj. ściskania, rozciągania.

Modelem najlepiej spełniającym powyższe założenia jest utworzenie relacji „jeden do wielu” między encją *Material* a *Zwick*. Implementacja takiej relacji przebiegała analogicznie do relacji między *Zwick* a *ZwickData*.

W modelu obiektowym, klasa *Material* zawiera pola „nazwa stopu” oraz „skład chemiczny”. Ze względu na konieczność konfiguracji relacji, dodano także „wskaźnik” na obiekt klasy *Zwick*, w postaci pola *\$tasks* z adnotacją *@OneToMany*. Należało dodać także odpowiednie pole w klasie *Zwick* *\$material* z adnotacjami *@ManyToOne* oraz *@JoinColumn*.

2.3. Grupowanie materiałów i zadań w projekty

W kolejnym etapie pogrupowano poszczególne zestawy materiałów i zadań w jeden projekt. W tym celu utworzono klasę *Project*, która zawierała jedynie nazwę projektu i wskaźnik *\$materials* na obiekty klasy *Material*. Implementacja i utworzenie relacji „jeden do wielu” między klasą *Project* a *Material* przebiegała analogicznie jak tworzenie relacji *Material–Zwick*.

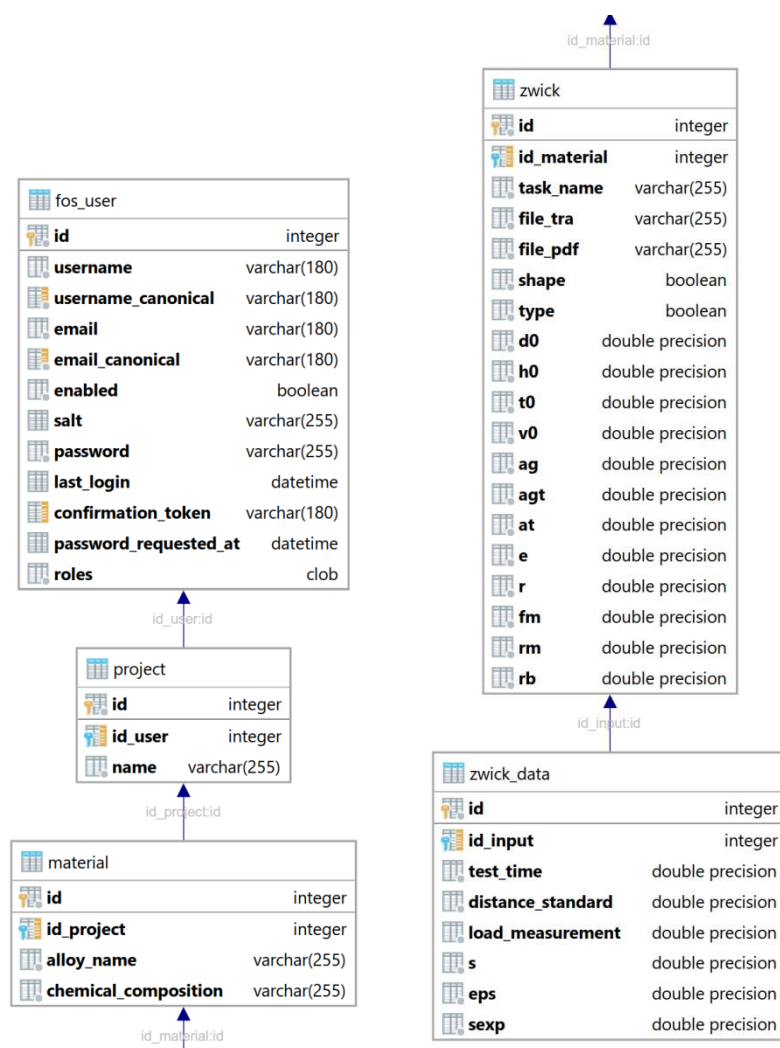
2.4. Obsługa kont użytkownika

W tym etapie wprowadzono obsługę kont użytkownika. W tym celu zastosowano zewnętrzną bibliotekę *FOS User Bundle* nie należącą do *Symfony*, a utworzoną przez społeczność użytkowników. Biblioteka ta ułatwia zarządzanie kontami użytkowników i zarządzanie prawami dostępu. Implementację funkcjonalności biblioteki umieszczono w osobnej bibliotece *User Bundle* obok dotychczas wykorzystywanej *App Bundle*. Klasę *User* rozszerzono o wskaźnik *\$projects* w celu utworzenia relacji „jeden do wielu” między użytkownikiem a projektami. Utworzono także własny kontroler *Login* w celu zaimplementowania własnej logiki logowania. Zmodyfikowano także domyślne formularze logowania i rejestracji, a także wprowadzono możliwość akceptacji użytkowników przez administratora systemu.

Po implementacji powyższych funkcjonalności poczynając od punktu 2.1, wywołano z konsoli komendę:

```
php /bin/console doctrine:database:create
```

dzięki której otrzymano strukturę danych przedstawioną na Rys. 9:



Rys. 9. Diagram bazy danych.

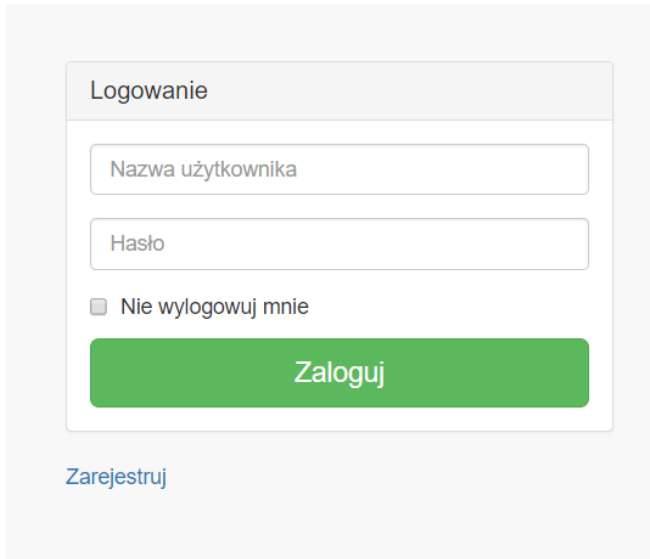
2.5. Interaktywne menu

Do utworzenia menu, którego zawartość zmieniała się będzie w zależności od stanu aplikacji wykorzystano kolejną zewnętrzną bibliotekę: *KNP Menu Bundle*. Biblioteka ta pozwoliła na utworzenie menu metodą obiektową poprzez utworzenie klasy *MenuBuilder*, w której utworzono metodę *mainMenu*. W celu umożliwienia korzystania z bazy danych, zarejestrowano *MenuBuilder* jako usługę dodając odpowiednią pozycję w pliku `app/config/services.yml` oraz podając parametr `@doctrine.orm.entity_manager`. W celu wyświetlenia pozycji związanych tylko z kontem zalogowanego użytkownika, przesłano do usługi parametr `@security.token_storage`. W celu wyświetlenia administratorowi linku do panelu administratora, przesłano do usługi parametr: `@security.authorization_checker`. W ten sposób można było z korzystać z następujących bibliotek: *Entity Manager*, *Token Storage* oraz *Authorization Checker*. Korzystając z ich możliwości stworzono 3-poziomowe

interaktywne menu, którego zawartość zmienia się wraz z zawartością konta użytkownika. Po przejściu na konkretną podstronę, która jest obecna w gałęzi menu, zostaje ona automatycznie rozwinięta i podświetlona, co ułatwia nawigację po aplikacji.

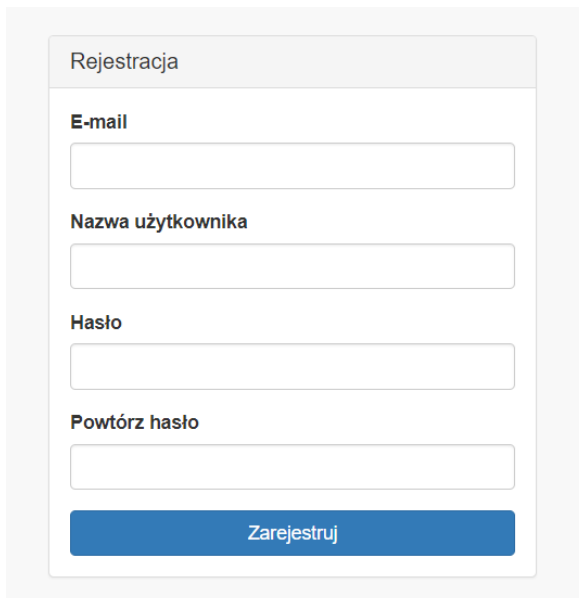
3. Testowanie

Po uruchomieniu aplikacji po raz pierwszy zostanie wyświetlona strona logowania przedstawiona na rys. 10:



Rys. 10. Okno logowania.

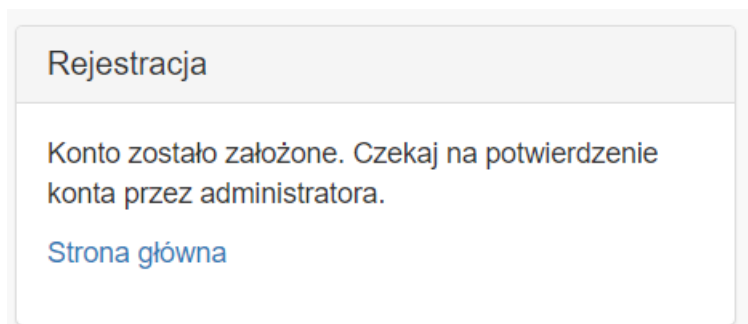
Nowy użytkownik, w celu uzyskania dostępu do systemu, musi założyć nowe konto wypełniając formularz rejestracyjny (rys. 11):



Rys. 11. Okno rejestracji.

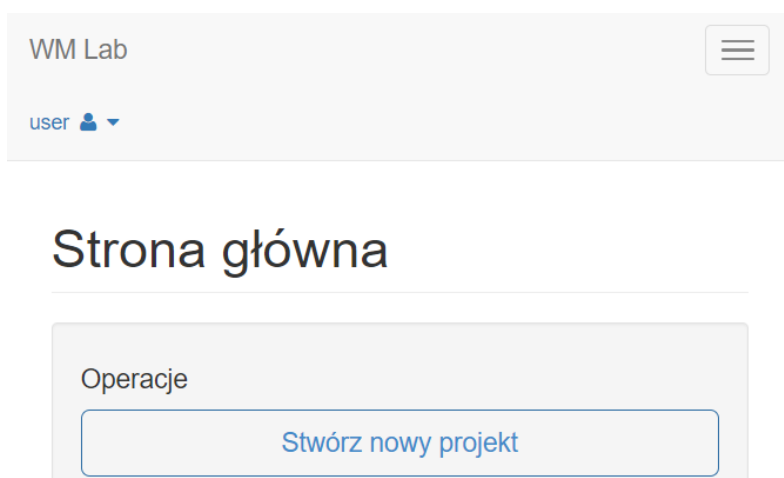
Rysunek 12 przedstawia informację o pomyślnym założeniu nowego konta. Rejestracja nowego konta musi zostać potwierdzona przez administratora. Do tego momentu

zalogowanie się i korzystanie z aplikacji będzie niemożliwe. Użytkownik z uprawnieniami administratora poprzez dedykowany odnośnik w menu bocznym ma dostęp do kokpitu, z którego może przydzielać dostęp do aplikacji określonym użytkownikom.



Rys. 12. Potwierdzenie utworzenia nowego konta.

Po zweryfikowaniu konta, użytkownik ma możliwość zalogowania się. Po zalogowaniu zostanie wyświetlona strona główna, przedstawiona na rysunku 13, na której znajdują się dynamicznie generowane menu, menu użytkownika, z którego możliwa jest edycja profilu lub wylogowanie oraz lista projektów (jeżeli istnieją). W oknie edycji profilu użytkownik ma możliwość edycji swojej nazwy użytkownika, adresu e-mail oraz hasła.



Rys. 13. Fragment strony głównej.

Po kliknięciu na przycisk „Stwórz nowy projekt” użytkownik zostanie poproszony o podanie nazwy nowego projektu. Po utworzeniu projektu, użytkownik ma możliwość dodania nowego materiału poprzez podanie jego nazwy i opcjonalnie składu chemicznego, co przedstawia rysunek 14. Okno edycji istniejącego materiału wygląda analogicznie, z dodatkowym przyciskiem do usunięcia materiału.

WM Lab

user

Dodawanie nowego materiału

Nazwa stopu

Skład chemiczny

Dodaj materiał

Przerwij

Rys. 14. Formularz dodawania nowego materiału.

Dopiero po dodaniu przynajmniej jednego materiału, użytkownik będzie miał możliwość dodania nowego zadania. W menu bocznym pojawi się pozycja z odnośnikiem do szczegółów nowo utworzonego materiału. Z poziomu widoku projektu użytkownik ma możliwość zobaczenia wszystkich materiałów i zadań należących do danego projektu, dodania nowego materiału lub zadania, edycji istniejących oraz usunięcia całego projektu łącznie ze wszystkimi utworzonymi w nim materiałami oraz zadaniami. Rysunek 15 przedstawia widok okna projektu.

WM Lab

user

Laboratorium

Nowy projekt

nowy materiał

Nowy projekt

Materiały

1. nowy materiał

Edytuj

Nowy materiał

Zadania

Nowe zadanie

Zmień nazwę projektu

Usuń Projekt i wszystkie dane

Rys. 15. Okno projektu.

Formularz dodawania zadania, składa się z dwóch części: „Parametry zadania” oraz „wyniki próby”. W części pierwszej, widocznej na rysunku 16, wprowadza się nazwę zadania oraz ustawia parametry wejściowe zadania, takie jak maszyna wytrzymałościowa, na której wykonany został test, materiał, z pośród wcześniej dodanych, kształt próbki, rodzaj próby wykonanej na wybranej maszynie, wymiary próbki, temperatura oraz prędkość próby. Pola „maszyna wytrzymałościowa”, „materiał”, „kształt próbki” oraz „rodzaj próby” są polami wyboru. Pola „maszyna wytrzymałościowa” i „kształt” są zablokowane, ponieważ w obecnej formie aplikacja nie pozwala na wybranie innych ustawień.

WM Lab user

Laboratorium

Nowy projekt

nowy materiał

Nowe zadanie

Parametry badania

Nazwa zadania

Maszyna wytrzymałościowa Zwick

Materiał nowy materiał

Kształt próbki Walec

Rodzaj próby Rozciąganie

d_0 Średnica próbki mm

h_0 Wysokość próbki mm

t_0 Temperatura próby °C

v_0 Prędkość próby mm/min

Wyniki próby

Rys. 16. Formularz tworzenia nowego zadania – „Parametry badania”.

Rysunek 17 przedstawia sekcję „wyniki próby”. W tej części formularza wprowadzane są parametry odczytane z maszyny wytrzymałościowej po zakończeniu próby. Wpisanie wartości w pola liczbowe nie jest wymagane, natomiast wymagane jest wybranie pliku TRA.

WM Lab

user

Laboratorium

Nowy projekt

nowy materiał

Nowe zadanie

Parametry badania

Wyniki próby

A_g	Wydłużenie nieproporcjonalne procentowe	%
A_{gt}	Wydłużenie całkowite procentowe	%
A_t	Wydłużenie całkowite procentowe	%
E	Moduł Younga	GPa
$R_{p0.2}$	Granica plastyczności	MPa
F_m	maksymalna siła	kN
R_m	Wytrzymałość na rozciąganie	MPa
R_b	Naprężenie przy zerwaniu	MPa

Dane wejściowe (TRA)
 Wybierz plik Nie wybrano pliku

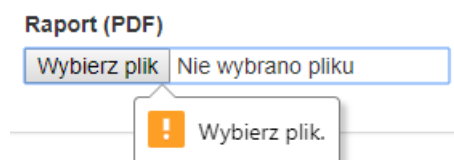
Raport (PDF)
 Wybierz plik Nie wybrano pliku

Rys. 17. Formularz tworzenia nowego zadania – „Wyniki próby”.

Po wprowadzeniu danych następuje sprawdzenie, czy pola formularza są kompletne, a wprowadzone wartości poprawne. Jeżeli wartości nie są poprawne zostanie wyświetlony opis błędu pod określonym elementem. Na Rys. 18 przedstawiono przykłady błędów polegających na wprowadzeniu liter w polu, gdzie spodziewana jest wartość liczbowo oraz braku wybrania wymaganego pliku.

A_g	aaa	%
-------	-----	---

! Ta wartość jest nieprawidłowa.

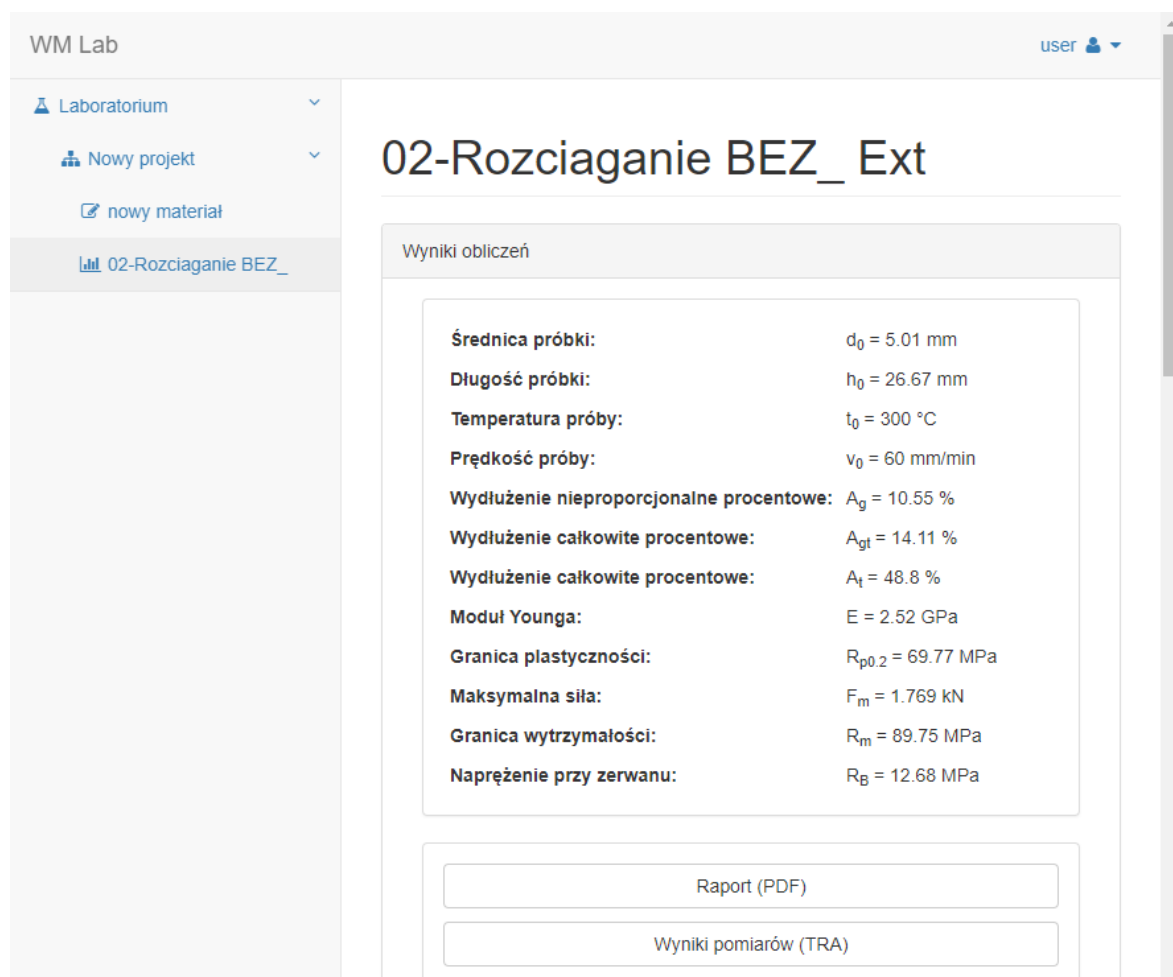


Rys. 18. Przykłady błędów walidacji formularza: niewłaściwy format danych oraz brak pliku.

Po kliknięciu na przycisk „Przejdź dalej”, jeżeli nie zostaną wykryte błędy, użytkownikowi zostanie wyświetlony wygenerowany raport składający się z kilku sekcji (rys. 19):

- wprowadzonych danych,
- odnośników do pobrania uprzednio wysłanych plików,
- tabeli z danymi odczytanymi z pliku TRA oraz wynikami obliczeń,
- wykresu zależności $\sigma - \epsilon$ dla zmierzonych wartości.

Do menu bocznego zostanie dodana pozycja wskazująca na utworzone zadanie i zostanie automatycznie podświetlona:



Rys. 19. Okno raportu - dane wejściowe, załączniki.

Rysunek 20 przedstawia interaktywną tabelę w oknie raportu. W celu umożliwienia przeglądania wyników na urządzeniach mobilnych oraz ułatwienia analizy dużej liczby rekordów, tabela z wynikami obliczeń zawiera szereg udogodnień:

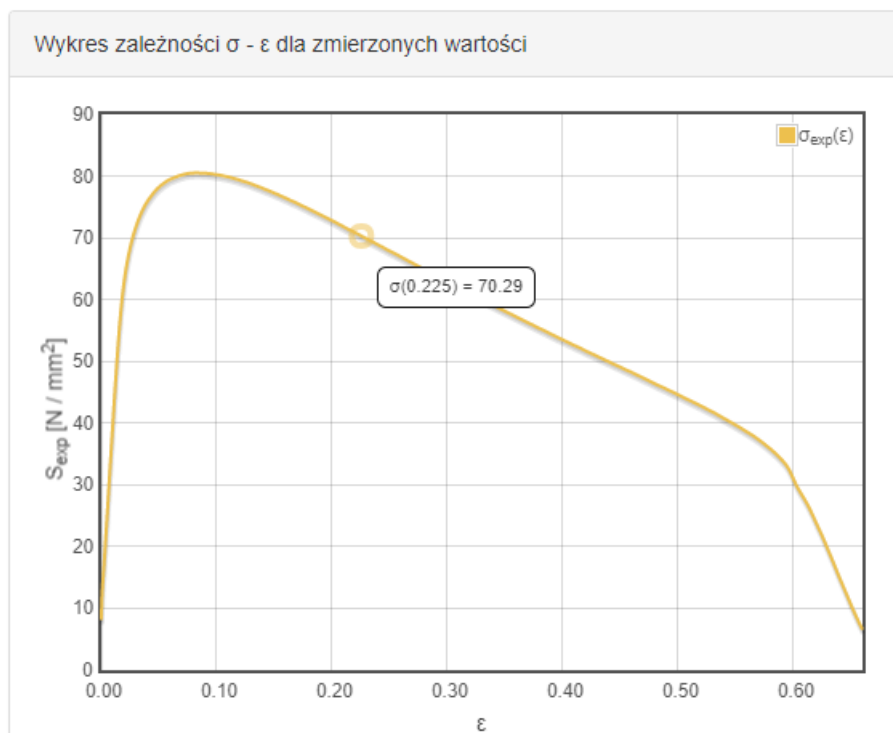
- wyświetlanie rekordów stronami,
- przeszukiwanie zawartości całej tabeli (np. w celu wyświetlenia określonego wiersza, wystarczy wpisać jego numer, a w celu znalezienia wiersza zawierającego wartość mieszczącą się w pewnym zakresie, należy rozpocząć wpisywanie w polu wyszukiwania i obserwować zmiany w tabeli),
- sortowanie według dowolnej kolumny,
- ukrywanie kolumn niemieszczących się na mniejszych wyświetlaczach.

Show entries
Search:

#	Czas badania	Droga standardowa [mm]	Siła standardowa [N]
1	0.000	-0.00000	180.69
<div>S [mm2]: 19.714</div> <div>Sexp [N / mm2]: 9.166</div> <div>ε: -0.00000</div>			
2	0.020	-0.00001	179.45
3	0.024	-0.00001	179.18

Rys. 20. Okno raportu - interaktywna tabela.

Wygenerowany wykres zawiera dwie opisane osie, siatkę z oznaczonymi wartościami i rzędnymi, wykreśloną krzywą oraz legendą. Po najechnaniu kursorem lub kliknięciu palcem na urządzeniu mobilnym na dowolny punkt krzywej, zostanie wyświetlona informacja ze współrzędnymi punktu na wykresie. Wykres jest w pełni responsywny i dostosowuje się do wielkości okna przeglądarki, co umożliwia przeglądanie go na urządzeniach mobilnych. Na rysunku 21 przedstawiono przykład interaktywnego wykresu:



Rys. 21. Okno raportu - interaktywny wykres.

4. Podsumowanie

Tworzenie aplikacji do wspomagania analizy wyników badań laboratoryjnych wymaga przeprowadzenia dokładnej analizy docelowej funkcjonalności aplikacji wraz ze stałą współpracą z osobami, dla których takie oprogramowanie jest przeznaczone. Implementację należy przeprowadzać w dwóch płaszczyznach: poprzez tworzenie funkcjonalności specyficznych dla danego rodzaju próby oraz dostarczenie obsługi systemowej, zapewniającej uniwersalność aplikacji. Dzięki temu rozszerzenie funkcjonalności aplikacji poprzez dodanie obsługi kolejnych rodzajów badań i maszyn, na których przeprowadza się pomiary realizowany będzie błyskawicznie. Należy również pamiętać o przemyślanej implementacji analizy poszczególnych prób, co pozwoli na maksymalną automatyzację powtarzających się operacji i tym samym skrócenie całkowitego czasu potrzebnego na uzyskanie oczekiwanych wyników.

Aplikacje webowe są bardzo złożonymi narzędziami. Konieczność implementacji warstwy klienta, serwera, danych oraz komunikacji między nimi wymaga utrzymania dyscypliny i dobrego przemyślenia architektury projektu już na etapie planowania. Niewątpliwymi zaletami takiego rozwiązania są:

- możliwość dostępu do danych z każdego miejsca,
- niezależność od systemu operacyjnego komputera i jego specyfikacji technicznej,
- łatwość wdrażania nowych wersji,
- spójność danych,
- możliwość jednoczesnej obsługi wielu użytkowników.

Korzyści te okupione są jednak koniecznością korzystania z wielu technologii w jednym projekcie i stałego dbania o utrzymanie kompatybilności poszczególnych modułów. Stopień skomplikowania aplikacji webowych jest znacznie większy od programów przeznaczonych na komputery lub urządzenia mobilne i wymaga znacznie większych nakładów czasowych i finansowych. Z tego powodu jedna osoba jest w stanie stworzyć i utrzymać jedynie proste aplikacje. Większe projekty tworzone są przez całe zespoły programistów, z których jedna część odpowiedzialna jest tylko za utrzymanie tzw. front-endu, a więc technologii uruchamianych w przeglądarce internetowej, a inny za back-end, a więc skrypty uruchamiane po stronie serwera, kolejny za obsługę baz danych, administrację,

testowanie, itd. Należy również rozważyć koszty utrzymania dostępu aplikacji do Internetu, a więc kosztów serwera i domeny internetowej. Jednak rozwój technologii, a przede wszystkim postępujące upowszechnianie się dostępu do Internetu sprawia, że liczba aplikacji webowych, ze względu na swoje zalety, dające im przewagę nad programami komputerowymi, stale rośnie w coraz większym tempie.

5. Bibliografia

- [1] Techtarget.com, *LAMP – Linux, Apache, MySQL, PHP*, <http://whatis.techtarget.com/definition/LAMP-Linux-Apache-MySQL-PHP>, 29.01.2018.
- [2] Q-Success, *Usage statistics and market share of Linux for websites*, <https://w3techs.com/technologies/details/os-linux/all/all>, 29.01.2018.
- [3] www.webdevelopersnotes.com, *What is web server – a computer OR a program?*, <https://www.webdevelopersnotes.com/what-is-web-server>, 29.01.2018.
- [4] U.S. Department of Transportation, 2001, *DATA INTEGRATION GLOSSARY*, <https://www.fhwa.dot.gov/infrastructure/asstmgmt/010394.pdf>, 29.01.2018.
- [5] Wall L., 2007, *Programming is Hard, Let's Go Scripting...*, <https://www.perl.com/pub/2007/12/06/soto-11.html/>, 29.01.2018.
- [6] The PHP Group, *Preface*, <http://php.net/manual/en/preface.php>, 29.01.2018.
- [7] The PHP Group, *History of PHP*, <http://php.net/manual/en/history.php.php>, 29.01.2018.
- [8] Potencier F., *Symfony website*, <https://symfony.com>, 29.01.2018.
- [9] Symfony-docs-pl, *Symfony: Buduj swoja aplikacje a nie swoje narzedzia*, https://symfony2-docs-pl.readthedocs.io/pl/latest/book/http_fundamentals.html#symfony-buduj-swoja-aplikacje-a-nie-swoje-narzedzia, 29.01.2018.
- [10] Wolny S., 2004 *Wytrzymałość materiałów, Część 4*, Wydawnictwa AGH, Kraków.
- [11] Sułkowski M., Ciesielska M, Jurczak – Kaczor P., 2016, *Analiza krzywych rozciągania w kształceniu technicznym*, Zeszyty Naukowe Wydziału Elektroniki i Automatyki Politechniki Gdańskiej, Nr 48, s. 85-92
- [12] Kusiak J., Danielewska-Tutecka A., Oprocha P., 2009, *Optymalizacja. Wybrane metody z przykładami zastosowań*. Wydawnictwo Naukowe PWN, Kraków.
- [13] Piątkowski T., 2015, *Wytrzymałość materiałów: laboratorium*, Wydawnictwa Uczelniane Uniwersytetu Technologiczno-Przyrodniczego, Bydgoszcz.

6. Spis rysunków

Rys. 1 Schemat powiązań między komponentami pakietu LAMP	9
Rys. 2 Przykład bazy danych w modelu relacyjnym	12
Rys. 3 Wykresy rozciągania dla materiałów: a) wykazujących wyraźną granicę plastyczności, b) wykazujących górną i dolną granicę plastyczności, c) bez wyraźnej granicy plastyczności	16
Rys. 4 Typowe charakterystyki naprężenie - odkształcenie uzyskane z różnych typów materiałów	19
Rys. 5 Schemat procesu produkcji oprogramowania.....	22
Rys. 6 Zależności między encjami bazy danych	25
Rys. 7 Schemat działania warstwy trasowania	34
Rys. 8 Diagram przedstawiający operacje wykonywane w metodzie <i>newAction</i>	35
Rys. 9 Diagram bazy danych	41
Rys. 10 Okno logowania.....	43
Rys. 11 Okno rejestracji.....	43
Rys. 12 Potwierdzenie utworzenia nowego konta	44
Rys. 13 Fragment strony głównej	44
Rys. 14 Formularz dodawania nowego materiału	45
Rys. 15 Okno projektu	46
Rys. 16 Formularz tworzenia nowego zadania – „Parametry badania”	47
Rys. 17 Formularz tworzenia nowego zadania – „Wyniki próby”	48
Rys. 18 Przykłady błędów walidacji formularza: niewłaściwy format danych oraz brak pliku	49
Rys. 19 Okno raportu - dane wejściowe, załączniki.....	49
Rys. 20 Okno raportu - interaktywna tabela	50
Rys. 21 Okno raportu - interaktywny wykres	51

7. Spis tabel

Tabela 1. Nomenklatura podstawowych obiektów bazodanowych w różnych terminologiach	11
Tabela 2. Zestawienie najważniejszych komponentów wykorzystanych w projekcie	29
Tabela 3. Zestawienie ważniejszych bibliotek JavaScript	29