
M4GiK
FILE
SERVER
Autor : Michał Szczygieł
DOCUMENTATION

DOKUMENTACJA PROJEKTU – SERWER WYMIANY PLIKÓW

Poniższy dokument zawiera :

Projektowanie Aplikacji Internetowych
Temat: Serwer wymiany plików

Michał Szczygieł, grupa 35, WFMIS

Spis Treści

Spis Rysunków

1. EXECUTIVE SUMMARY

Dokument ten jest specyfikacją projektu o nazwie „Server Wymiany Plików”, której autorem jest Michał Szczygieł student III roku Informatyki. Specyfikacja zwiera ścisłe informacje dotyczące projektu począwszy od wymagań funkcjonalnych po wymaganie niefunkcjonalne.

2. KONCEPCJA SYSTEMU

Praca z serwerem wymiany plików logistycznie odpowiadać będzie etapom:

- Jakie wykonywać będzie użytkownik, aby móc skorzystać z systemu.
- Jakie wykonywać będzie administrator w czasie pracy.

W systemie wyróżniać będziemy 3 różne grupy użytkowników:

- Administratorzy
- Użytkownicy
- Goście

Będą oni mieli przyznany dostęp do poszczególnych modułów systemu na różnych poziomach uprawnień, wynikających z ich kompetencji i potrzeb.

Serwis będzie przeznaczony do publicznego użytku. Jest to idealne rozwiązanie dla osób, które będą chciały mieć kopię swoich danych lub mieć do nich zdalny dostęp przy pomocy przeglądarki internetowej. Dane, będzie można upublicznić, przez co każdy będzie mógł je pobrać lub przeznaczyć tylko do użytku prywatnego.

Dane w systemie przechowywane będą na dwóch różnych serwerach:

- Baza danych
  - Będą tu przechowywane informację na temat użytkowników, plików, praw dostępu etc.
- Repozytorium
  - Będą tu przechowywane wszystkie pliki użytkowników, do których będą mieli dostęp przez serwis.

Do korzystania z serwisu będzie wymagane posiadania przeglądarki internetowej (minimum: IE 6 lub Mozilla Firefox 4) i dostępu do internetu. Szybkość wymiany plików pomiędzy użytkownikiem a serwerem będzie ograniczona przez prędkość łącza użytkownika jak i maksymalną przepustowość serwera. Łatwość obsługi zapewni prosty i przyjemny interfejs.

Grupy użytkowników w systemie:

Gość to osoba korzystająca z serwisu, od której niewymagane jest zalogowanie się do systemu (założenie konta). Będzie ona miała jedynie możliwość przeglądania i pobierania plików różnego typu udostępnionych od innych użytkowników (pliki o statusie publicznym).

Użytkownik to osoba korzystająca z serwisu, która będzie miała możliwość założenia konta, do którego będzie miała dostęp prywatny. Dzięki temu, będzie ona miała rozszerzony zasób możliwości w porównaniu do niezarejestrowanej osoby:

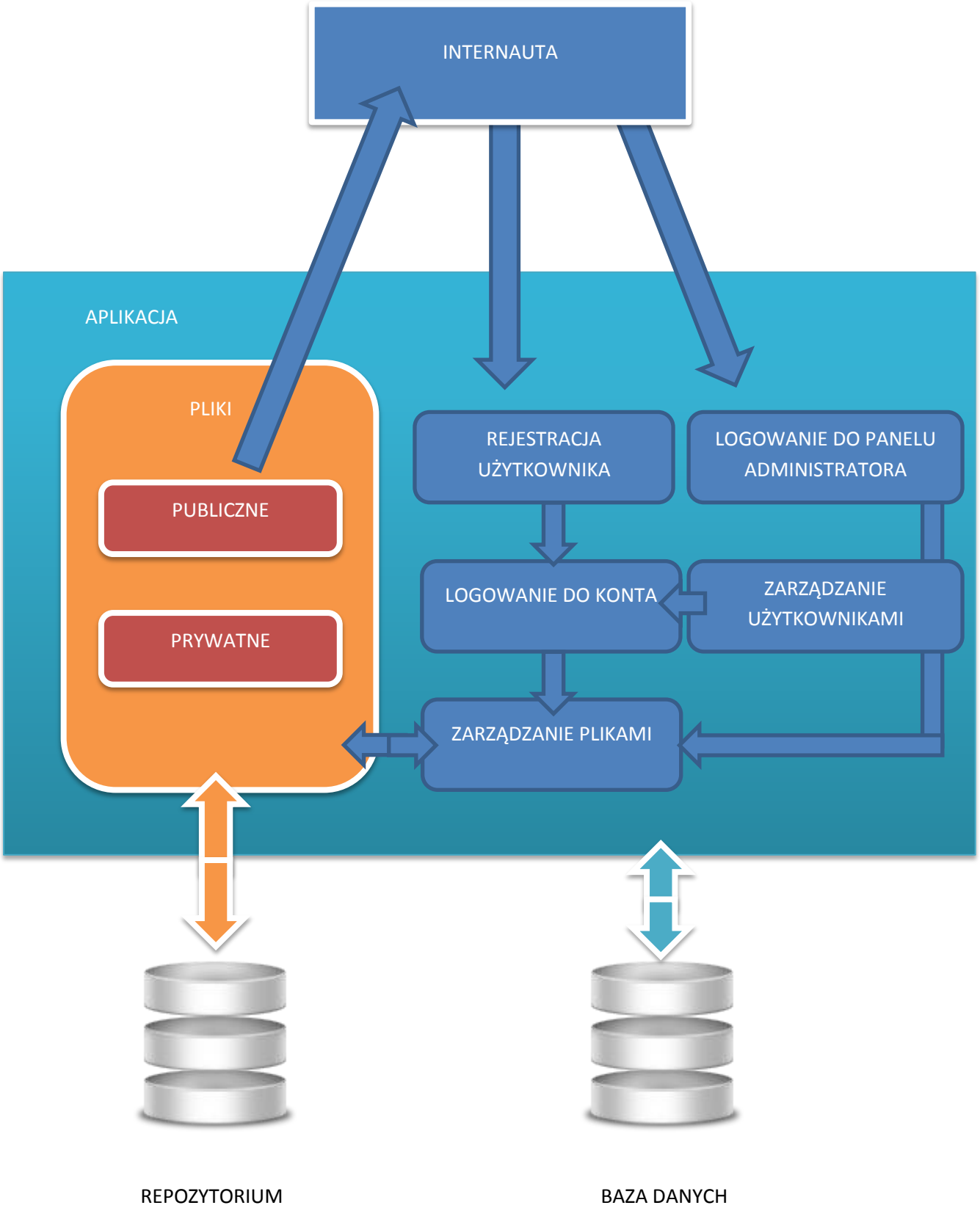
- Będzie posiadała pewną początkową określoną ilość przestrzeni dyskowej znajdującej się na serwerze, którą będzie mogła wykorzystywać do gromadzenia plików różnego typu.
- Będzie mogła dodawać/usuwać oraz ustawiać tryb plików (status publiczny lub prywatny).
- Będzie miała możliwość wysyłania/odbierania wiadomości do/od innych użytkowników/administratorów.
- Będzie miała możliwość wyszukiwania/pobierania plików od innych użytkowników, które zostały oznaczone, jako publiczne.

Administrator to osoba zarządzająca systemem oraz serwerami przechowywującymi dane:

- Będzie miała możliwość edytowania danych użytkowników.
- Będzie miała możliwość edytowania/pobierania/usuwania wszystkich plików (status publiczny i prywatny)
- Będzie miała możliwość wysyłania/odbierania wiadomości do/od innych użytkowników/administratorów.

3. ANALIZA WYMAGAŃ

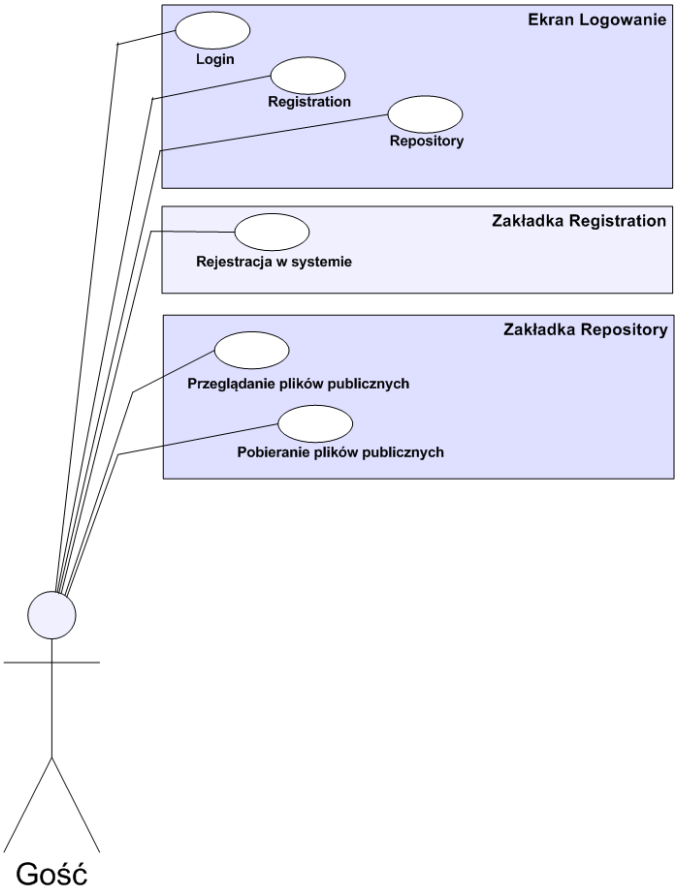
1. Model przepływu informacji w systemie.



4. WYMAGANIA FUNKCJONALNE

Wymagania funkcjonalne opisują funkcje wykonywane przez system. Powinny one dotyczyć wyłącznie zewnętrznych funkcji systemu.

1. Użytkownik będzie miał możliwość skorzystania z serwisu o uprawnieniu **gość**.
   1.1. Gość nie musi rejestrować ani logować się do systemu.
   1.2. Będzie miał możliwość przeglądania i pobierania publicznych plików.



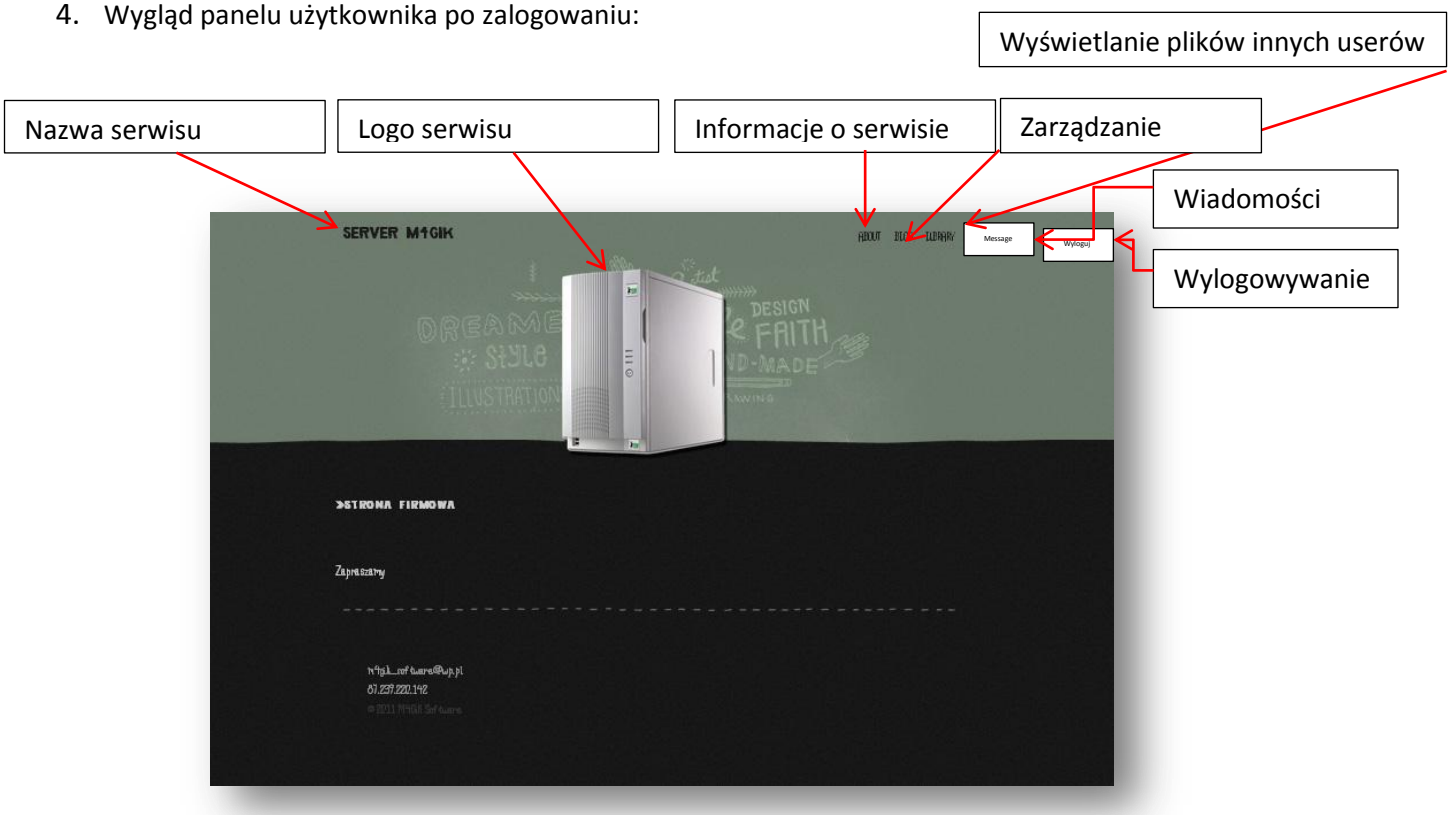
Rysunek 3: Przypadek użycia - Gość

2. Użytkownik chcący zmienić swoje uprawnienia z poziomu gość będzie miał możliwość stworzenia swojego konta i zalogowania się do serwisu.
   2.1. Rejestracja użytkownika będzie przebiegała przez podanie emaila, loginu i hasła.
      2.1.1. Wprowadzanie tych danych będzie odbywało się poprzez wprowadzenie danych z klawiatury w trzech osobnych polach tekstowych i zaakceptowaniu poprzez wciśnięcie guzika „rejestracja”.
         2.1.1.1. Dane powinny przejść kontrolę przed wprowadzeniem do bazy danych.
      2.1.2. Dane te będą przechowywane w bazie danych.

3. Możliwość zalogowania użytkownika do serwisu.
   3.1. Logowanie użytkownika będzie odbywało się poprzez podanie loginu i hasła.
      3.1.1. Wprowadzanie tych danych będzie odbywało się poprzez wprowadzenie danych z klawiatury w dwóch osobnych tekstowych i zaakceptowaniu poprzez wciśnięcie guzika „zaloguj".
      3.1.2. Po wprowadzeniu danych, aplikacja sprawdzi czy taki użytkownik znajduje się w bazie danych oraz czy podane hasło zgadza się z tym znajdującym się w bazie.
   3.2. Po zalogowaniu użytkownik będzie cały czas zalogowany aż do zakończenia sesji (*odnośnik do wylogowania*).

Rysunek 4 : Panel logowania

4. Wygląd panelu użytkownika po zalogowaniu:



Rysunek 5 : Panel Użytkownika

4.1. Panel użytkownika będzie posiadał 4 linki do podstron:
   4.1.1. Link **informacje o serwisie** (5).
   4.1.2. Link **zarzadzanie plikami** (6).
   4.1.3. Link **wyświetlanie plików innych użytkowników** (7).
   4.1.4. Link **wiadomości** (8)**.**

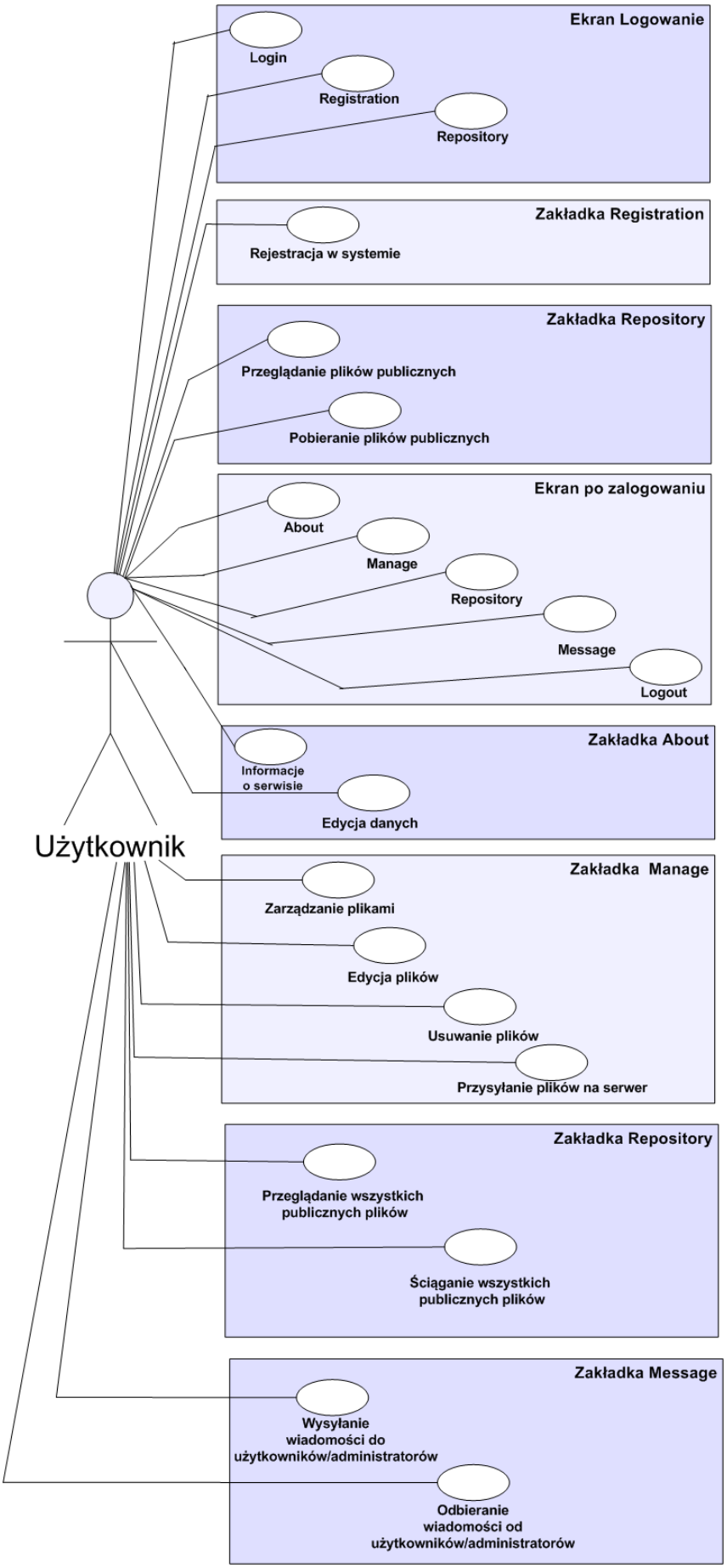
5. Podstrona **informacje o serwisie** powinna zawierać najważniejsze informację dotyczące serwisu.
   5.1. Opcjonalna nazwa linku „about”.
   5.2. Użytkownik będzie mógł tu znaleźć informacje jak używać serwis.
      5.2.1. Użytkownik będzie miał dostęp do przeglądu/edycji swoich danych.
   5.3. Wygląd tej podstrony będzie podobny do strony głównej.

6. Podstrona **zarządzanie plikami**.
   6.1. Opcjonalna nazwa linku „manage”.
   6.2. Użytkownik będzie mógł przeglądać/edytować/usuwać pliki.
   6.3. Użytkownik będzie mógł wysyłać pliki na serwer (*o ile dysponować będzie wystarczającą przestrzenią dyskową*) oraz nadać mu odpowiedni tryb (status publiczny lub prywatny).

7. Podstrona wyświetlająca **pliki innych użytkowników**
   7.1. Opcjonalna nazwa linku „repository”.
   7.2. Użytkownik będzie mógł przeglądać pliki dostępne do pobrania.

8. Podstrona **wiadomości**.
   8.1. Opcjonalna nazwa linku „message”.
   8.2. Użytkownik będzie mógł przeglądać/wysyłać/usuwać wiadomości do innych zalogowanych użytkowników/administratorów.

9. Dla opcji wylogowanie, użytkownik zostanie przeniesiony na stronę główną, gdzie jest okno logowania (3).
   9.1. Sesja użytkownika zostanie zakończona, po wylogowaniu nie można wrócić na poprzednią podstronę ani żadną z innych podstron.



Rysunek 6: Przypadek użycia – Użytkownik

10. Administrator systemu.
    - 10.1. Logowanie administratora będzie odbywało się poprzez podanie loginu i hasła.
        - 10.1.1. Wprowadzanie tych danych będzie odbywało się poprzez wprowadzenie danych z klawiatury w dwóch osobnych tekstowych i zaakceptowaniu poprzez wciśnięcie guzika „zaloguj".
        - 10.1.2. Po wprowadzeniu danych, aplikacja sprawdzi czy taki administrator znajduje się w bazie danych oraz czy podane hasło zgadza się z tym znajdującym się w bazie.
    - 10.2. Po zalogowaniu administrator będzie cały czas zalogowany aż do zakończenia sesji (*odnośnik do wylogowania*).
    - 10.3. Będzie miał dostęp do panelu administratora, gdzie będzie miał dostęp do edytowania danych użytkowników, edytowania/pobierania/usuwania wszystkich plików (status publiczny i prywatny), wysyłania/odbierania/usuwania wiadomości do/od innych użytkowników/administratorów.

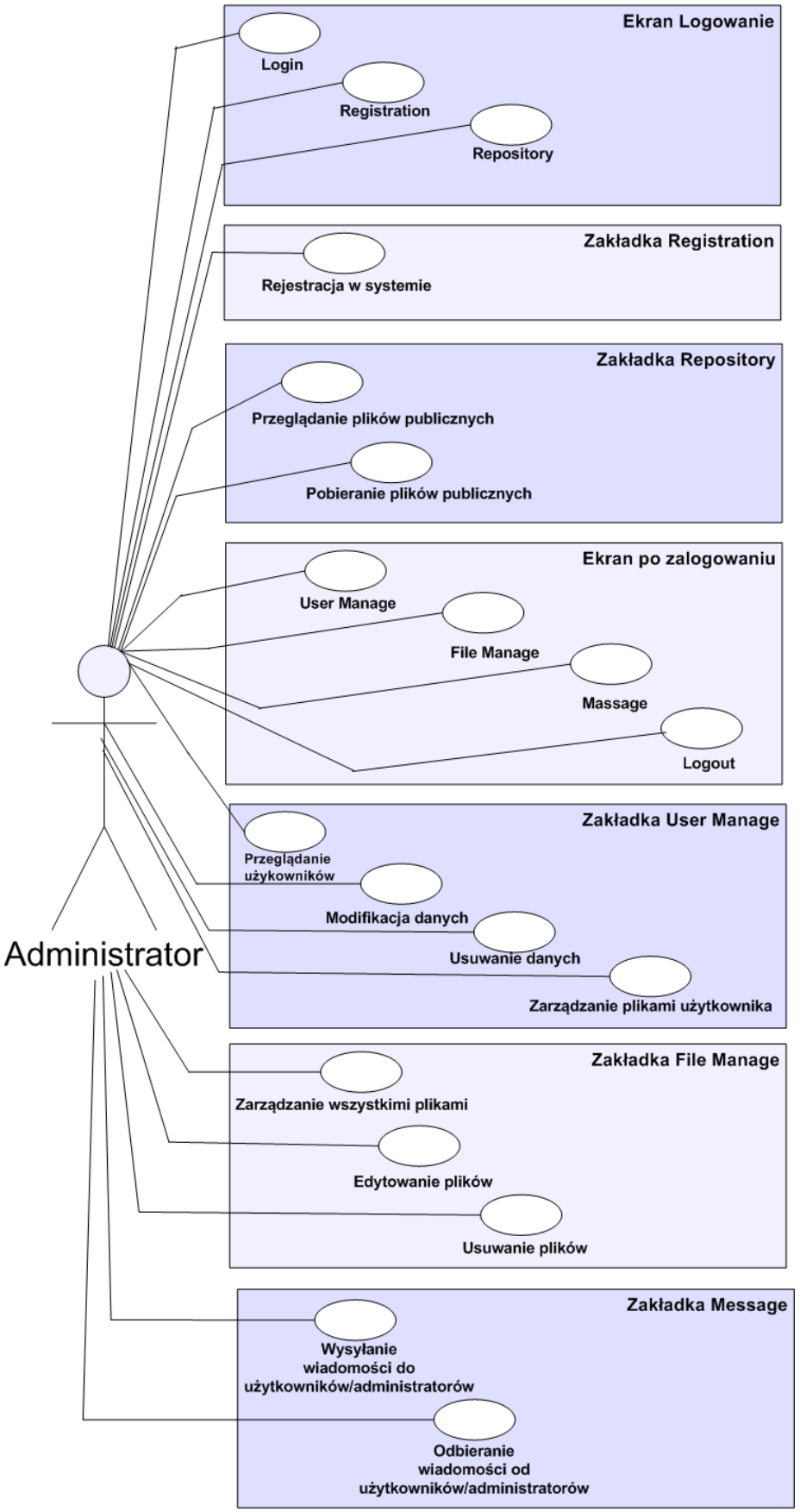
11. Panel administratora będzie posiadał 3 linki do podstron :
    - 11.1. Link **zarzadzanie użytkownikami** (12).
    - 11.2. Link **zarzadzanie plikami** (13).
    - 11.3. Link **wiadomości** (14)**.**

12. Podstrona **zarządzanie użytkownikami**.
    - 12.1. Opcjonalna nazwa linku „user manage".
    - 12.2. Administrator będzie mógł przeglądać/edytować/usuwać użytkowników.
    - 12.3. Administrator będzie mógł zarządzać plikami danego użytkownika indywidualnie.
        - 12.3.1. Administrator mógł przeglądać/edytować/usuwać/pobierać pliki uzytkownika.

13. Podstrona **zarządzanie plikami.**
    - 13.1. Opcjonalna nazwa linku „file manage".
    - 13.2. Administrator będzie mógł przeglądać/edytować/usuwać wszystkimi plikami.

14. Podstrona **wiadomości**.
    - 14.1. Opcjonalna nazwa linku „message".
    - 14.2. Administrator będzie mógł przeglądać/wysyłać/usuwać wiadomości do innych zalogowanych użytkowników/administratorów.

15. Dla opcji wylogowanie, użytkownik zostanie przeniesiony na stronę główną, gdzie jest okno logowania (3).
    - 15.1. Sesja administrator zostanie zakończona, po wylogowaniu nie można wrócić na poprzednią podstronę ani żadną z innych podstron.



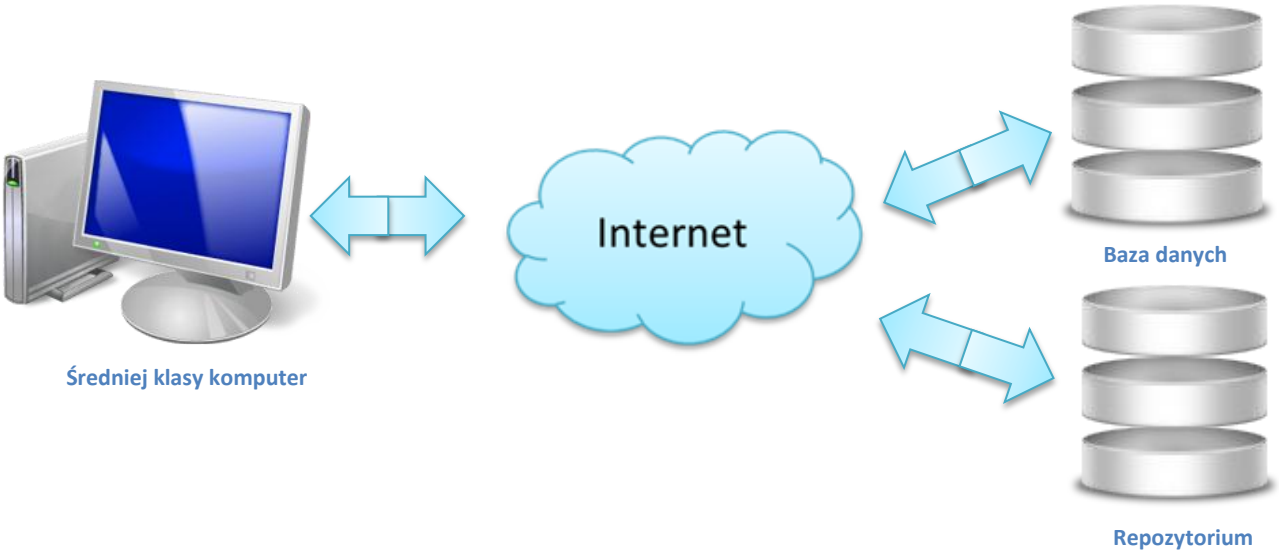
Rysunek 7: Przypadek użycia – Administrator

5. WYMAGANIA NIEFUNKCJONALNE

Wymagania niefunkcjonalne opisują ograniczenia, przy których system musi realizować swoje funkcje. Wynikają z potrzeb użytkownika, ograniczeń budżetowych, konieczności współpracy z innymi systemami sprzętu lub oprogramowania, czynników zewnętrznych

Wymagania sprzętowe

Do prawidłowej pracy system wymaga dostarczenia odpowiedniego sprzętu oraz infrastruktury sieciowej. Najważniejsze z punktu widzenia gościa/użytkownika/administratora jest to, że wystarczy mu średniej klasy komputer z dostępem do Internetu oraz zainstalowanym ogólnodostępnym oprogramowaniem (przeglądarka WWW). Z punktu widzenia systemu kluczową rolę będzie miało zagwarantowanie przestrzeni dyskowej w bazach danych oraz stałe szybkie łącze do internetu.



Bezpieczeństwo systemu

System zarządza danymi użytkowników, zatem wymaga zastosowania szczególnych środków bezpieczeństwa.

a) Panel administratora

Strona administracyjna będzie dostępna jedynie dla wybranej grupy osób, dlatego też zostanie zaimplementowany odpowiedni mechanizm kontroli dostępu do stron szczególnie istotnych dla funkcjonowania systemu. Administrator będzie logował się do systemu na stronie logowania. Umożliwi mu to dostęp do zabezpieczonej części systemu (panelu administratora). Gdy administrator zaloguje się do systemu, aplikacja obsługująca panel będzie przechowywała jego dane do czasu, a się on nie wyloguje. Wszystkie zabezpieczone strony wchodzące w skład panelu, przed udostepnieniem administratorowi swej zawartości będą sprawdzały czy jest on zalogowany.

b) Panel użytkownika

Użytkownik będzie logował się do systemu na stronie logowania. Umożliwi mu to dostęp do zabezpieczonej części systemu (panelu użytkownika). Gdy użytkownik zaloguje się do systemu, aplikacja obsługująca panel będzie przechowywała jego dane do czasu, a się on nie wyloguje. Wszystkie zabezpieczone strony wchodzące w skład panelu, przed udostepnieniem użytkownikowi swej zawartości będą sprawdzały czy jest on zalogowany.

Hasła będą przechowywane w bazie, tzn. przechowywany będzie jedynie kod haszujący hasło. W celu sprawdzenia poprawności hasła wprowadzonego przez użytkownika/administratora będzie generował się jego kod haszujący i porównywał się go z kodem przechowywanym w bazie danych.

6. SŁOWNIK

Login – nazwa użytkownika przechowywanego w bazie danych, który chce edytować własne notatki.

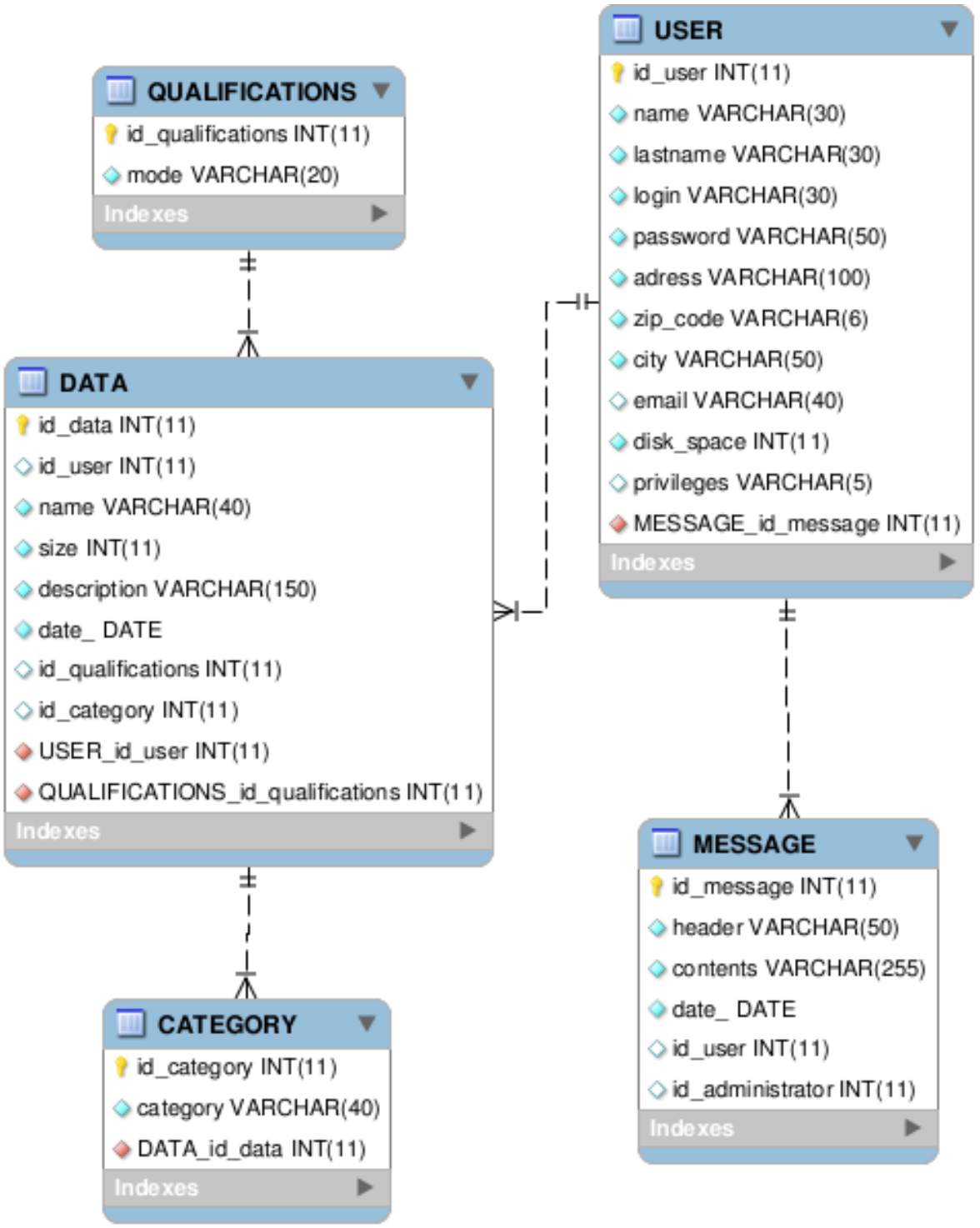
Hasło – hasło użytkownika, które przechowywane jest w bazie danych.

Zaloguj – przycisk służący do zalogowania się użytkownika na podstawie loginu i hasła.

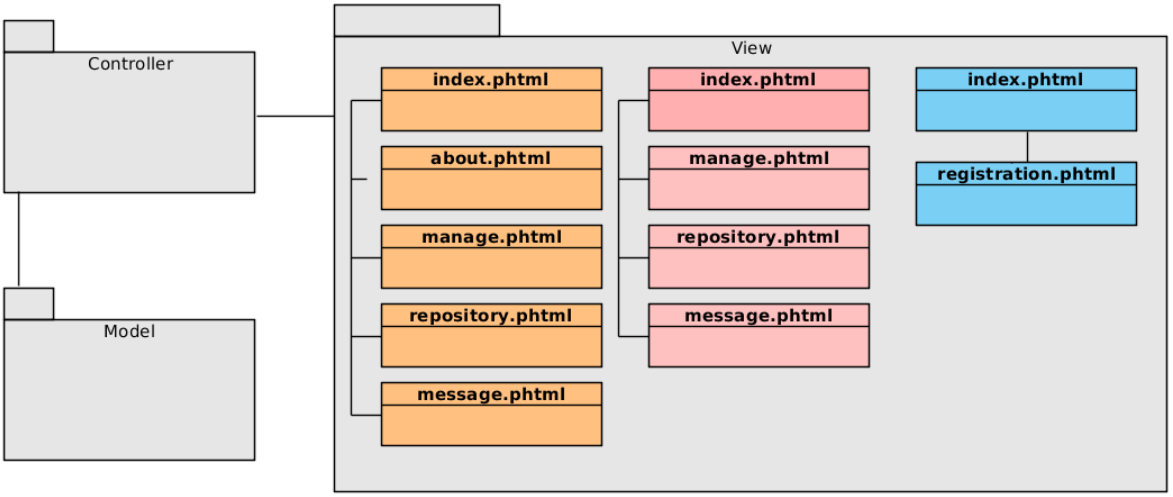
Zarejestruj – opcja umożliwiająca dodania nowego użytkownika do bazy danych.

Serwis – strona internetowa z pewnymi funkcjonalnościami.

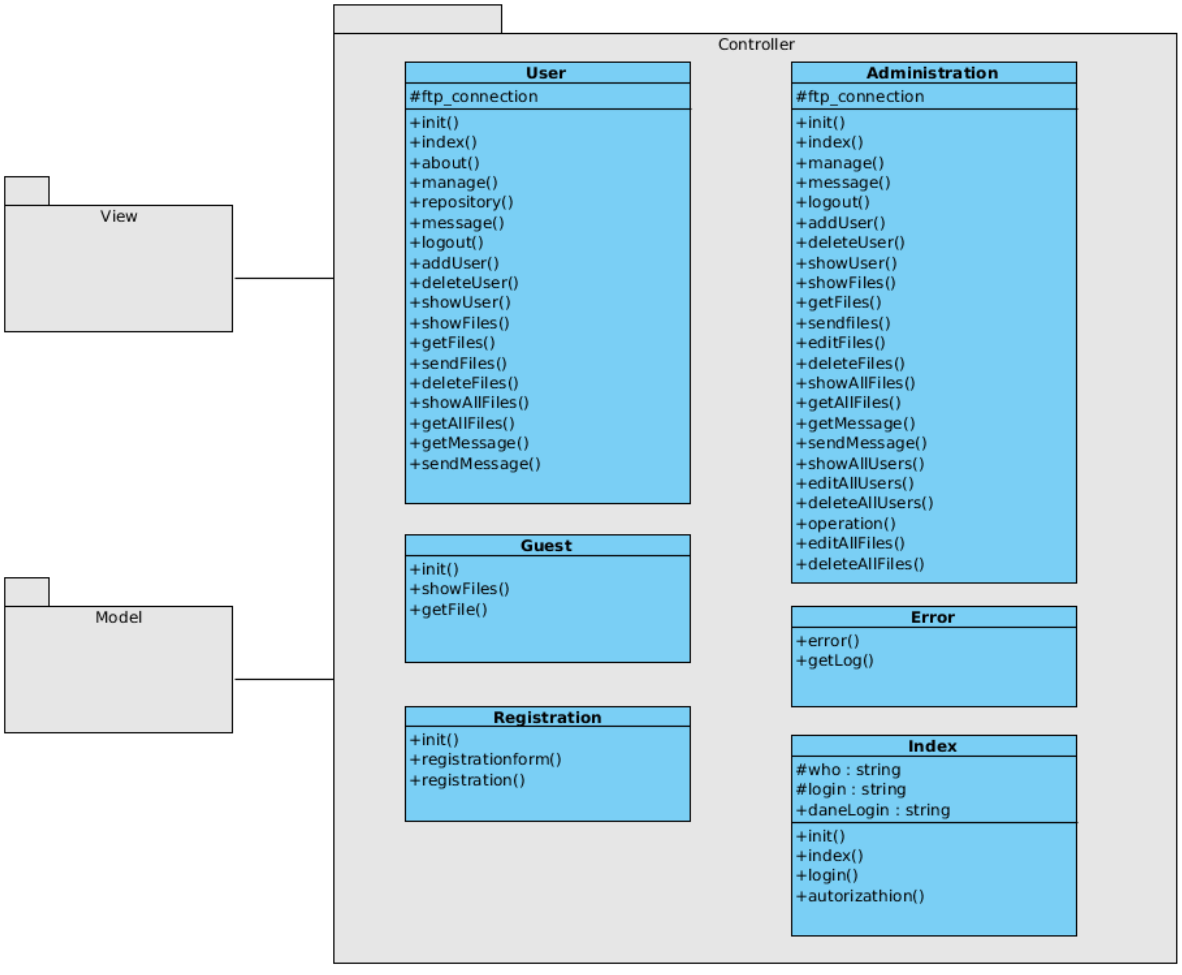
• SCHEMAT BAZY DANYCH



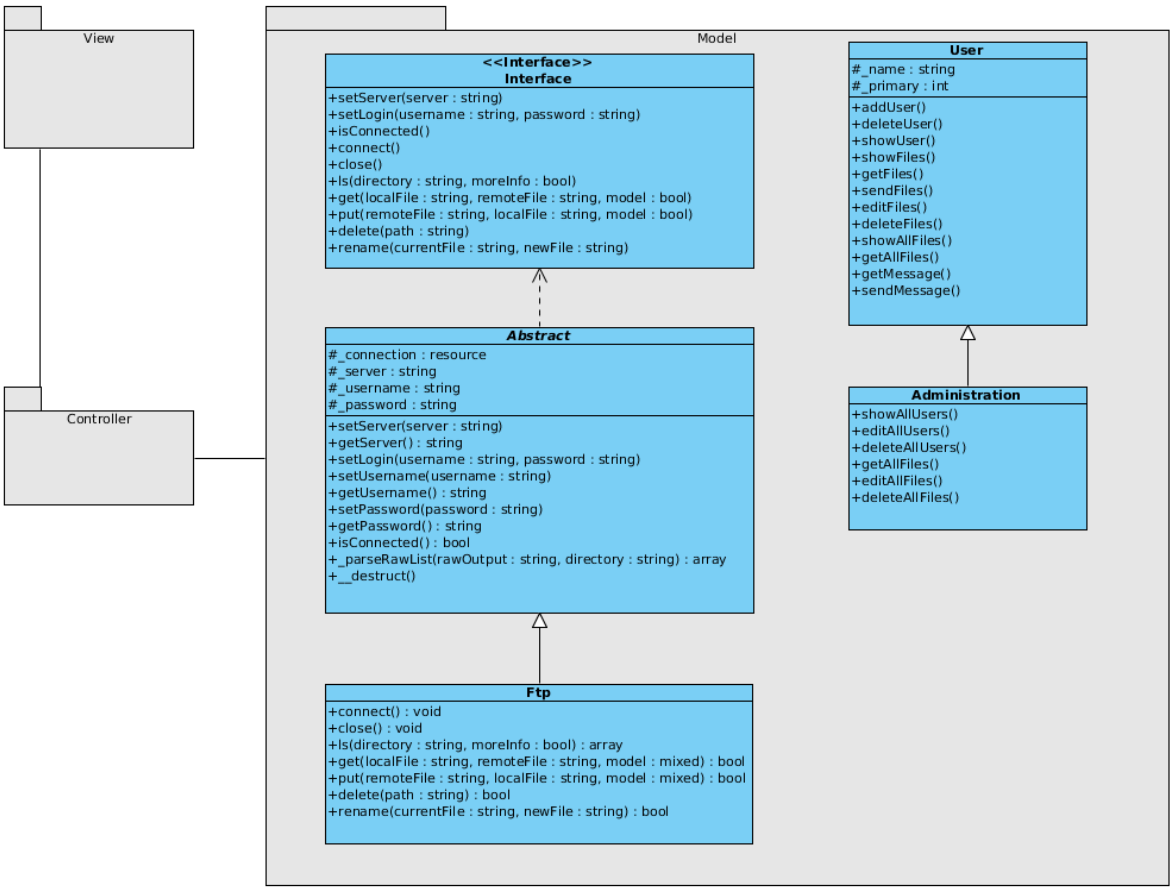
• DIAGRAMY KLAS



Rysunek 1 : Diagram klas przedstawiający warstwę widoku projektu, warstwa ta składa się z plików „phtml”

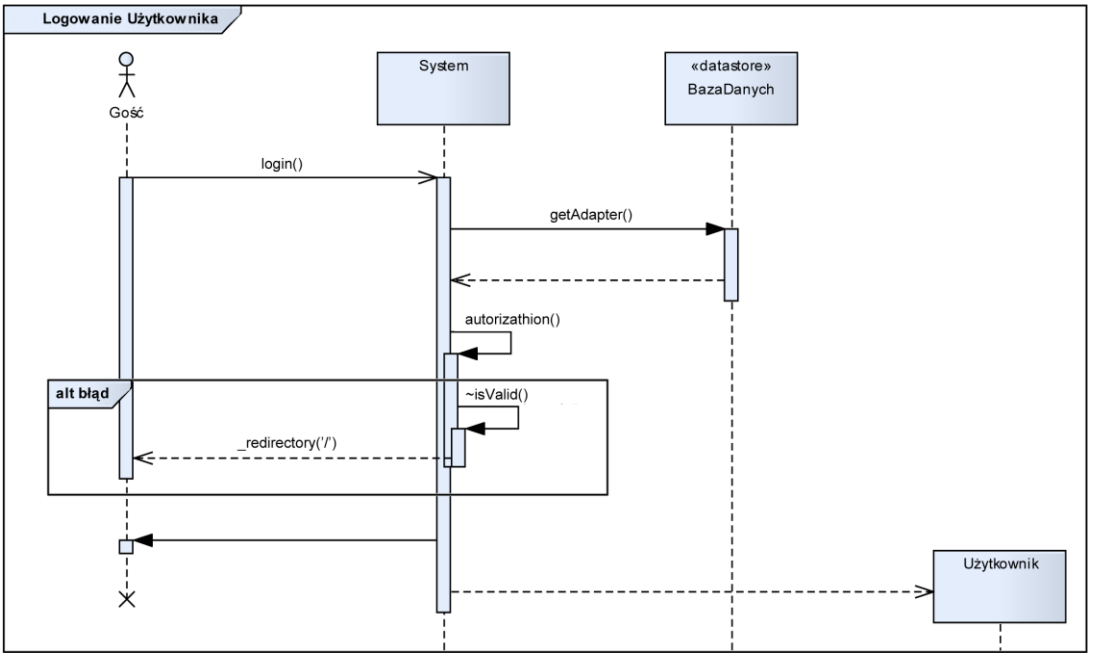


Rysunek 2 : Diagram klas przedstawiający warstwę kontrolera projektu, opracowany na bazie diagramów przypadków użycia

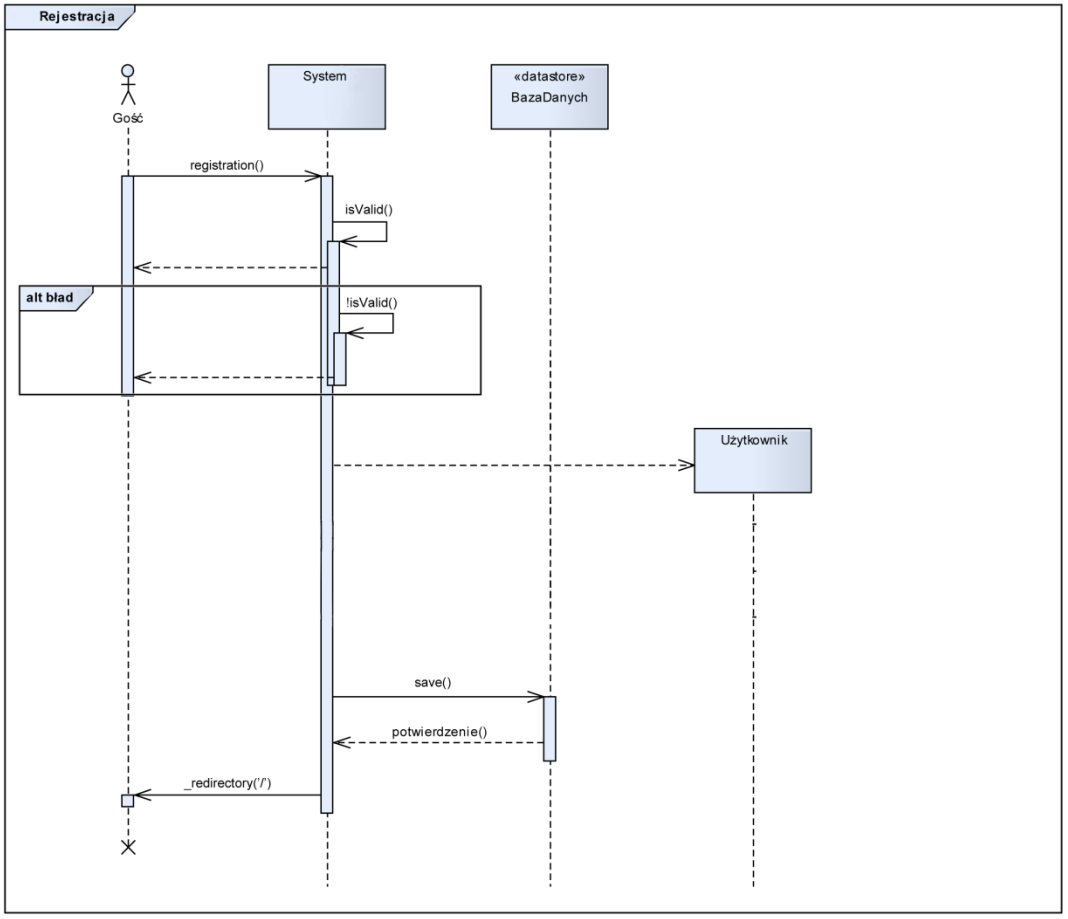


Rysunek 3 : Diagram klas przedstawiający logikę projektu opracowany na podstawie diagramów przypadków użycia

• DIAGRAMY SEKWENCJI



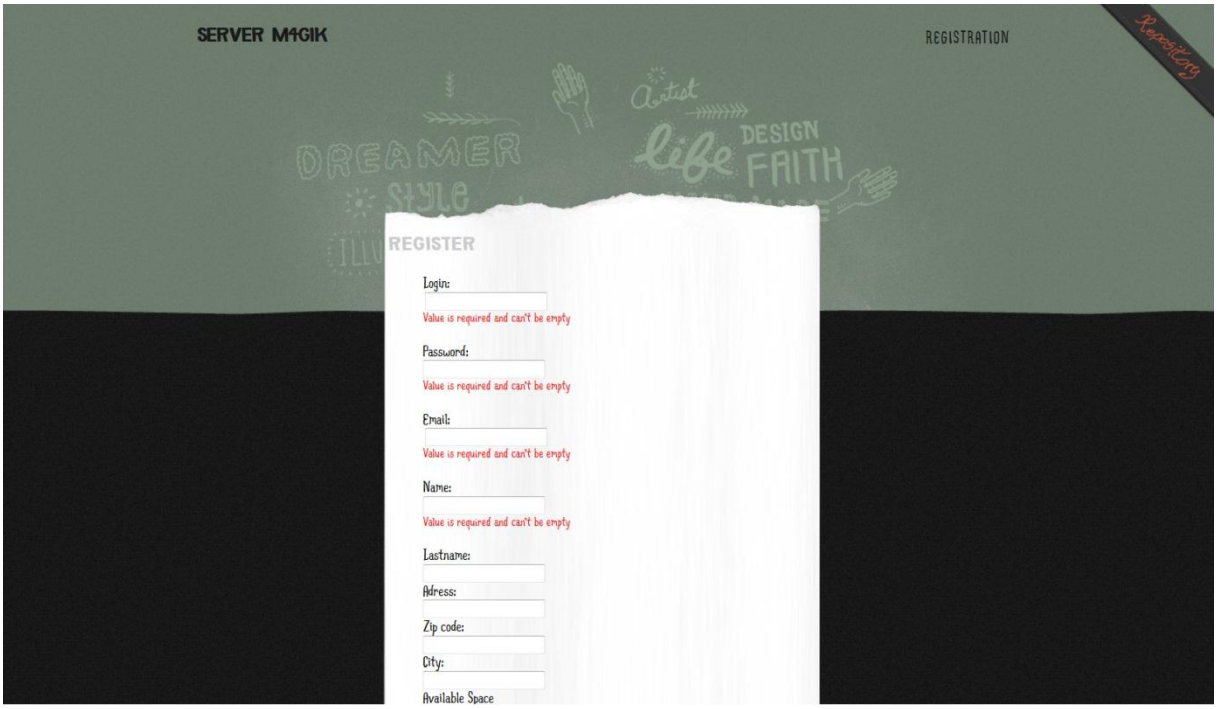
Rysunek 4 : Diagram sekwencji przedstawiający proces logowania



Rysunek 5 : Diagram sekwencji przedstawiający proces rejestracji

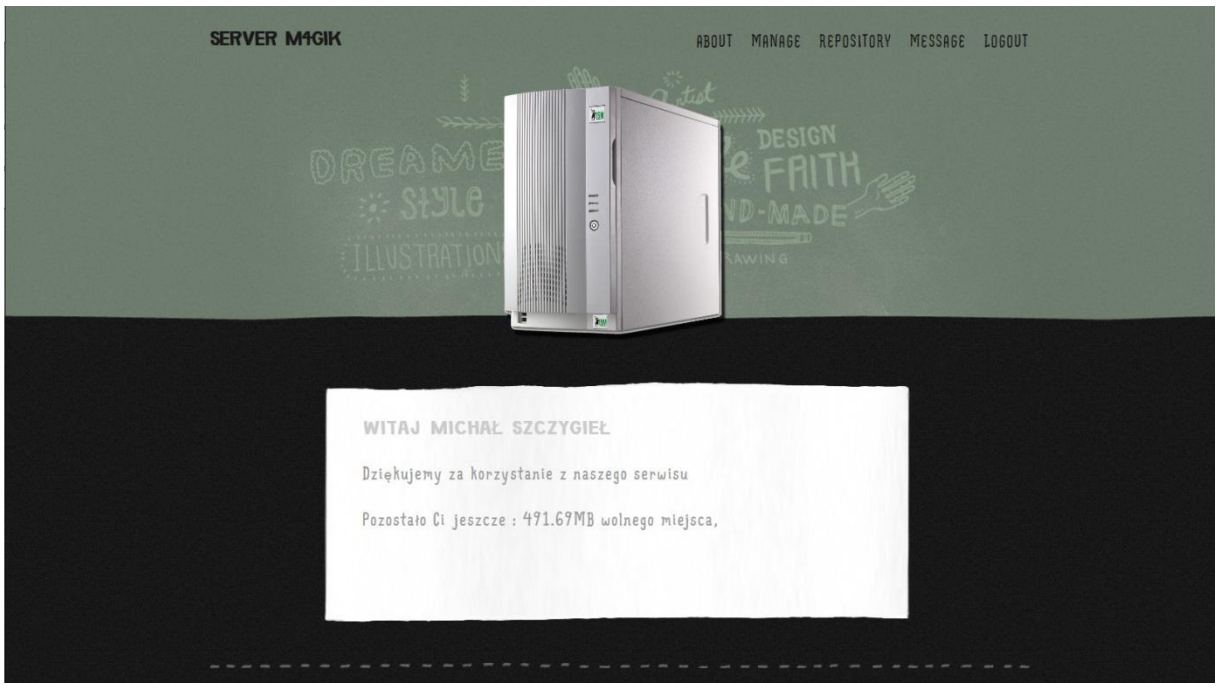
• REALIZACJA PROJEKTU

Rysunek 6 : Panel logowania



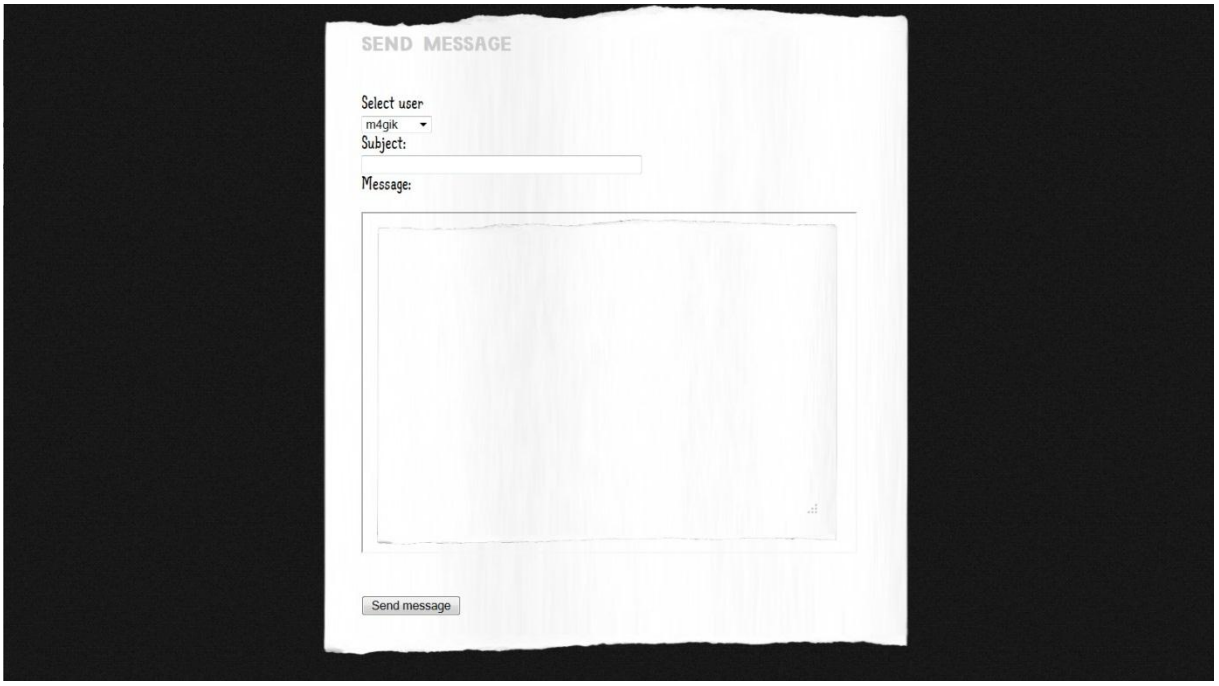
Rysunek 7 : Panel rejestracji

Rysunek 8 : Panel po zalagowaniu się administratora

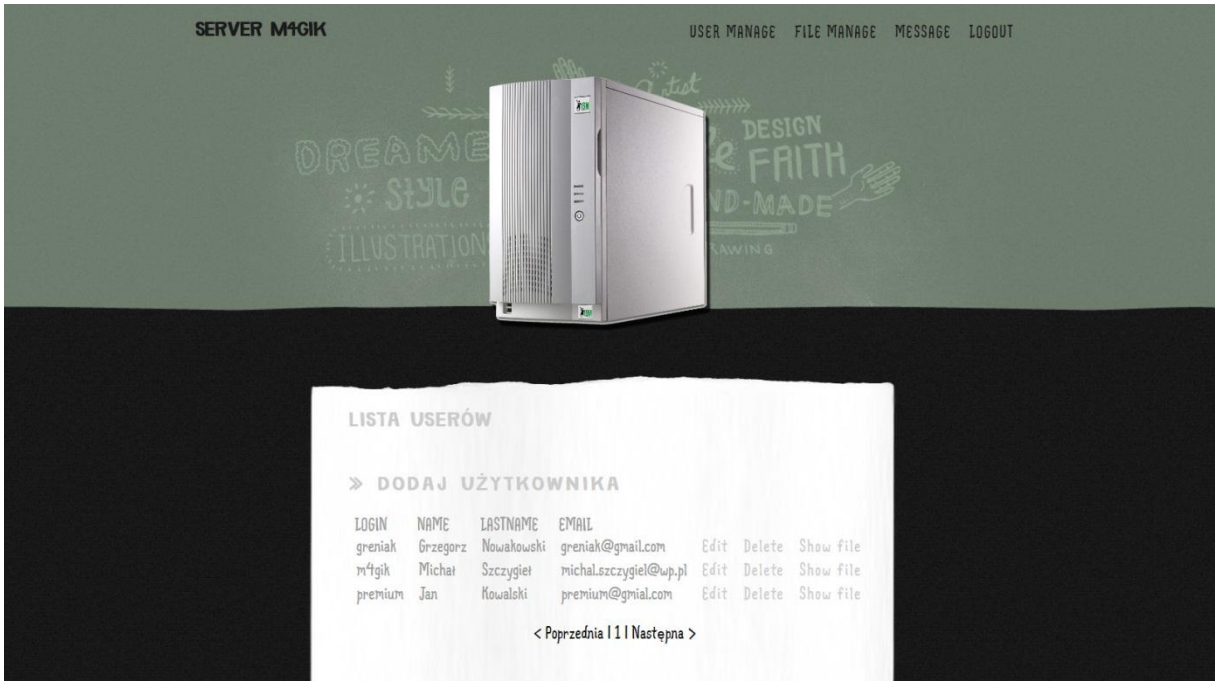


Rysunek 9 : Panel po zarejestrowaniu się użytkownika

Rysunek 10 : Panel wysyłania plików



Rysunek 11 : Panel wysyłania wiadomości

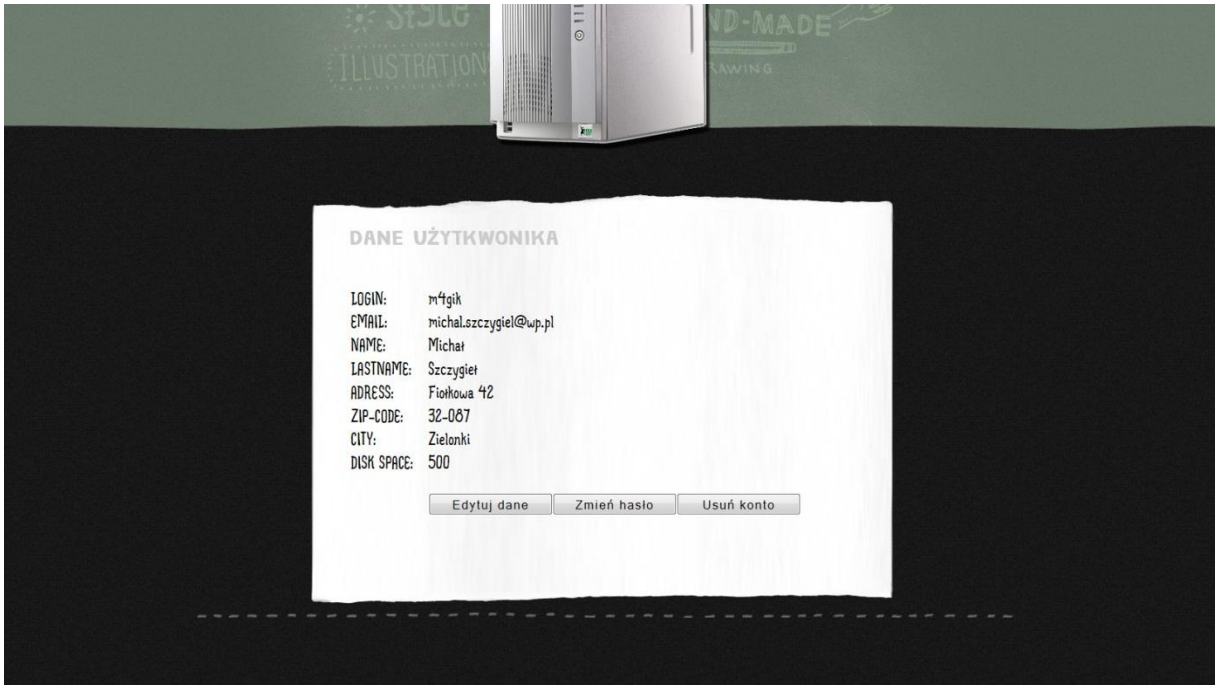


Rysunek 12 : Panel zarządzania użytkownikami

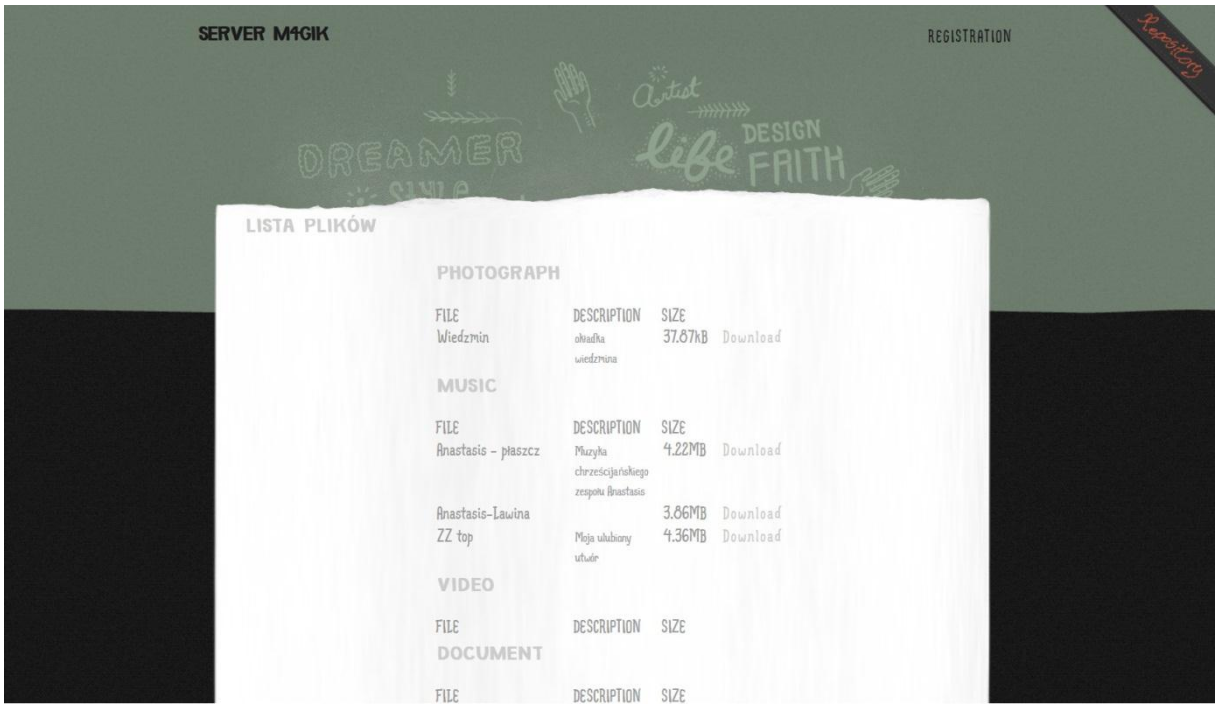
Rysunek 13 : Panel usuwania danego użytkownika

Rysunek 14 : Panel edycji danych użytkownika z poziomu administratora

Rysunek 15 : Zarządzanie wszystkimi plikami



Rysunek 16 : Panel zarządzania danymi użytkownika



Rysunek 17 : Główne repozytorium

• UNIT TESTY

KONFIGURACJA UNIT TESTÓW

```
<phpunit bootstrap="./TestHelper.php" colors="true">
    <testsuite name="Zend Framework Unit Test Demo">
        <directory>./</directory>
    </testsuite>

    <!-- Optional settings for filtering and logging -->
    <filter>
        <whitelist>
            <directory suffix=".php">../library/</directory>
            <directory suffix=".php">../application/</directory>
            <exclude>
                <directory suffix=".phtml">../application/</directory>
            </exclude>
        </whitelist>
    </filter>

    <logging>
        <log type="coverage-html" target="./log/report" charset="UTF-8"
         yui="true" highlight="true" lowUpperBound="50" highLowerBound="80"/>
        <log type="testdox-html" target="./log/testdox.html" />
    </logging>
</phpunit>
```

Rysunek 18 : phpunit.xml

```php
<?php
// set our app paths and environments
define('BASE_PATH', realpath(dirname(__FILE__) . '/../'));
define('APPLICATION_PATH', BASE_PATH . '/application');
define('TEST_PATH', BASE_PATH . '/tests');
define('APPLICATION_ENV', 'testing');

// Include path
set_include_path('.' . PATH_SEPARATOR . BASE_PATH . '/library'
    . PATH_SEPARATOR . get_include_path());

// Set the default timezone !!!
date_default_timezone_set('Europe/Warsaw');

require_once 'Zend/Application.php';
$application = new Zend_Application(APPLICATION_ENV,
    APPLICATION_PATH . '/configs/application.ini');
$application->bootstrap();
```

Rysunek 19 : TestHelper.php

```php
<?php
require_once 'Zend/Application.php';
require_once 'Zend/Test/PHPUnit/ControllerTestCase.php';

abstract class ControllerTestCase extends
Zend_Test_PHPUnit_ControllerTestCase
{
    protected function setUp()
    {
        // we override the parent::setUp() to solve an issue regarding not
        // finding a default module
    }
}
```

Rysunek 20 : ControllTestCase.php

```php
<?php
// file: tests/application/controllers/IndexControllerTest.php
require_once TEST_PATH . '/ControllerTestCase.php';

class IndexControllerTest extends ControllerTestCase
{
    public function testCanWeDisplayOurHomepage()
    {
        // go to the main page of the web application
        $this->dispatch('/');

        // check if we don't end up on an error page
        $this->assertNotController('error');
        $this->assertNotAction('error');

        // ok, no error so let's see if we're at our homepage
        $this->assertModule('default');
        $this->assertController('index');
        $this->assertAction('index');
        $this->assertResponseCode(200);
    }
}
```

Rysunek 21 : IndexControllerTest

Powyższy test jednostkowy testuje akcje index w kontrolerze index. Powyższy test przechodzi w 100 %.

```php
<?php
  require_once 'PHPUnit/Framework/TestCase.php';
  class Application_Model_Adminisration extends PHPUnit_Framework_TestCase
  {
  …
  }

  Becomes

  <?php
  require_once TEST_PATH . '/DatabaseTestCase.php';
  class Application_Model_Adminisration extends DatabaseTestCase
  {
  …
  }
```

Rysunek 22 : Model database

```php
<?php
  require_once 'Zend/Application.php';
  require_once 'Zend/Test/PHPUnit/DatabaseTestCase.php';
  require_once 'PHPUnit/Extensions/Database/DataSet/FlatXmlDataSet.php';

  abstract class DatabaseTestCase extends Zend_Test_PHPUnit_DatabaseTestCase
  {
      private $_dbMock;
      private $_application;

      protected function setUp()
      {
          $this->_application = new Zend_Application(
              APPLICATION_ENV, APPLICATION_PATH . '/configs/application.ini');
          $this->bootstrap = array($this, 'appBootstrap');
          parent::setUp();
      }
      public function appBootstrap()
      {
          $this->application->bootstrap();
      }
      protected function getConnection()
      {
          if (null === $this->_dbMock) {
              $bootstrap = $this->application->getBootstrap();
              $bootstrap->bootstrap('db');
              $connection = $bootstrap->getResource('db');
              $this->_dbMock = $this->createZendDbConnection($connection,'in2it')
              Zend_Db_Table_Abstract::setDefaultAdapter($connection);
          }
          return $this->_dbMock;
      }
      protected function getDataSet()
      {
          return $this->createFlatXMLDataSet(
              dirname(__FILE__) . '/_files/initialDataSet.xml');
      }
  }
```

Rysunek 23 : DatabaseTestCase.php

```php
<?php
if (!defined('SETUP_TEST')) {
        // require_once '../../setupTest.php';
}

//require_once 'Application/Models/Ftp/Ftp.php';

class Applcation_Models_Ftp_FtpTest extends UnitTestCase
{
        /**
         * Please set Your personal options
         *
         * @var array
         */
        protected $ options = array(
                'server'   => '87.239.220.142',
                'username' => 'server',
                'password' => '*******'
        );

        public function setUp()
        {
        }

        public function tearDown()
        {
        }

        public function testGetProtocol()
        {
              $url = 'ftp://87.239.220.142';
              $result = parse_url($url, PHP_URL_SCHEME);
              $this->assertEqual('ftp', $result, 'Oczekiwany URL Sheme jest inny niż FTP');
        }

       public function testCreateFactory()
        {
              $ftp = Applcation_Models_Ftp_Ftp::factory('ftp');
            $this->assertIsA($ftp, 'KontorX_Ftp', 'Utworzony obiekt nie jest typu "Aplication_Models_Ftp"');
        }

        public function testCheckAdapterTypeFtp()
        {
                $ftp = Applcation_Models_Ftp_Ftp::factory('ftp');
                $adapter = $ftp->getAdapter();
            $this->assertIsA($adapter, 'KontorX_Ftp_Adapter_Ftp', 'Adapter nie jest typu " Aplication_Models__Adapter_Ftp"');
       }

        public function testConnectFailure()
        {
                $ftp = Applcation_Models_Ftp_Ftp::factory('ftp', array(
                        'server' => 'ftp.noexists.info',
                        'username' => 'usernamame',
```

```
54.                         'password' => 'password'
55.                 ));
56.
57.                 try {
58.                         $ftp->getAdapter()->connect();
59.                         $this->fail("Połączenie do serwera nie może zostać nawiązane");
60.                 } catch(Exception $e) {
61.                         $this->assertIsA($e, 'Aplication_Models_Ftp_Adapter_Exception', "Wyjątej nie jest instancją '
    Aplication_Models_Ftp_Adapter_Exception'!");
62.                 }
63.         }
64.
65.         public function testConnectSuccessLoginFailure()
66.         {
67.                 $ftp = Applcation Models Ftp Ftp::factory('ftp', array(
68.                         'server'   => '87.239.220.142',
69.                         'username' => 'non user',
70.                         'password' => 'non_password'
71.                 ));
72.
73.                 try {
74.                         $ftp->getAdapter()->connect();
75.                         $this->fail("Połączenie do serwera nie może zostać nawiązane");
76.                 } catch(Exception $e) {
77.                         $this->assertIsA($e, 'Aplication_Models_Ftp_Adapter_Exception', "Wyjątej nie jest instancją
    'KontorX_Ftp_Adapter_Exception'!");
78.                 }
79.         }
80.
81.         public function testConnectSuccessLoginSuccess()
82.         {
83.                 $ftp = Applcation_Models_Ftp_Ftp::factory('ftp', $this->_options);
84.
85.                 try {
86.                         $ftp->getAdapter()->connect();
87.                 } catch(Exception $e) {
88.                         $this->fail("Połączenie do serwera nie zostało nawiązane:", $e-
    >getMessage());
89.                 }
90.         }
91.
92.         public function testPutFile()
93.         {
94.                 $ftp = Applcation_Models_Ftp_Ftp::factory('ftp', $this->_options);
95.
96.                 $remoteFile = 'httpdocs/remoteTestPutFile.txt';
97.                 $localFile = 'testPutFile.txt';
98.
99.                 $result = $ftp->put($remoteFile, $localFile);
100.                         $this->assertTrue($result, 'Wystąpił błąd w trakcie wysyłania pliku na
    serwer');
101.                 }
102.
```

```
103.                    public function testGetFile()
104.                    {
105.                            $ftp = Applcation Models Ftp Ftp::factory('ftp', $this-> options);
106.
107.                            $localFile = 'testGetFile.txt';
108.                            $remoteFile = 'httpdocs/remoteTestPutFile.txt';
109.
110.                            $result = $ftp->get($localFile, $remoteFile);
111.                            $this->assertTrue($result, 'Wystąpił błąd w trakcie pobierania pliku z
    serwera');
112.                    }
113.
114.                public function testRenameAFile()
115.                {
116.                    $ftp = Applcation Models Ftp Ftp::factory('ftp', $this-> options);
117.
118.                    $currentFilename = 'httpdocs/remoteTestPutFile.txt';
119.                    $newFilename = $currentFilename.'.bak';
120.
121.                    $result = $ftp->rename($currentFilename, $newFilename);
122.                    $this->assertTrue($result, 'Wystąpił błąd w trakcie zmiany nazwy pliku na
    serwera');
123.                }
124.
125.                public function testRenameBFile()
126.                {
127.                    $ftp = Applcation Models Ftp Ftp::factory('ftp', $this-> options);
128.
129.                    $currentFilename = 'httpdocs/remoteTestPutFile.txt.bak';
130.                    $newFilename = 'httpdocs/remoteTestPutFile.txt';
131.
132.                    $result = $ftp->rename($currentFilename, $newFilename);
133.                    $this->assertTrue($result, 'Wystąpił błąd w trakcie zmiany nazwy pliku na
    serwera');
134.                }
135.
136.                    public function testEqualPutAndGetFile()
137.                    {
138.                            $putLocalFile = 'testPutFile.txt';
139.                            $getLocalFile = 'testGetFile.txt';
140.                            $result = (file get contents($putLocalFile) ==
    file get contents($getLocalFile));
141.                            $this->assertTrue($result, 'Plik przesłany na serwer i ponowie pobrany z
    serwera są sobie różne!');
142.
143.                            unlink($getLocalFile);
144.                    }
145.
146.                public function testDeleteFile()
147.                {
148.                    $ftp = Applcation_Models_Ftp_Ftp::factory('ftp', $this->_options);
149.
150.                    $path = 'httpdocs/remoteTestPutFile.txt';
```

```
151.                    $result = $ftp->delete($path);
152.                    $this->assertTrue($result, 'Wystąpił błąd w trakcie usuwania pliku na serwera');
153.                }
154.
155.                    public function testListRawParse()
156.                    {
157.                            $rawList = array(
158.                                    "drwxr-xr-x 3 gabriel gabriel 4096 2010-06-12 16:22 Adapter",
159.                                    "-rw-r--r-- 1 gabriel gabriel 4338 2010-06-21 22:11 FtpTest.php",
160.                                    "-rw-r--r-- 1 gabriel gabriel   11 2010-06-12 16:22
    testPutFile.txt",
161.                                    "drwxr-x--- 5 widmogrod 504   4096 Oct        1     2009 anon_ftp"
162.                            );
163.
164.                            $result = array();
165.
166.                            foreach($rawList as $list) {
167.                                    // to nie jest najlepsze rozwiązanie ale zawsze ogranicza pole
    błędu
168.                                    if (preg_match('#(\d{4}-\d{2}-\d{2})#', $list, $matched)) {
169.                                            $info = sscanf($list, "%s %d %s %s %d %s %s %s");
170.                                            list($permisions, $size, $user, $group, $filesize, $date,
    $hour, $filename) = $info;
171.                                    } else {
172.                                            $info = sscanf($list, "%s %d %s %s %d %s %d %s %s");
173.                                            list($permisions, $size, $user, $group, $filesize, $month,
    $day, $year, $filename) = $info;
174.                                            $date = $month . ' ' . $day . ' ' . $year;
175.                                            $hour = '';
176.                                    }
177.
178.                                    $result[] = array(
179.                                            'type' => ((substr($permisions,0,1) == 'd') ? 'DIR' :
    'FILE'),
180.                                            'filesize' => $filesize,
181.                                            'permisions' => $permisions,
182.                                            'user' => $user,
183.                                            'group' => $group,
184.                                            'time' => strtotime("$date, $hour")
185.                                    );
186.                            }
187.
188.                            var_dump($result);
189.
190.                            $success = array(
191.                              array (
192.                                    "type"=> "DIR",
193.                                    "filesize"=> 4096,
194.                                    "permisions"=> "drwxr-xr-x",
195.                                    "user"=> "server",
196.                                    "group"=>"server",
197.                                    "time"=>1276352520
198.                              ),
```

```
199.                            array (
200.                                  "type"=> "FILE",
201.                                  "filesize"=> 4338,
202.                                  "permisions"=> "-rw-r--r--",
203.                                  "user"=> "server",
204.                                  "group"=>"server",
205.                                  "time"=>1277151060
206.                            ),
207.                            array (
208.                                  "type"=> "FILE",
209.                                  "filesize"=> 11,
210.                                  "permisions"=> "-rw-r--r--",
211.                                  "user"=> "server",
212.                                  "group"=>"server",
213.                                  "time"=>1276352520
214.                            ),
215.                            array (
216.                                  "type"=> "DIR",
217.                                  "filesize"=> 4096,
218.                                  "permisions"=> "drwxr-x---",
219.                                  "user"=> "server",
220.                                  "group"=>"504",
221.                                  "time"=>1254348000
222.                            )
223.                          );
224.
225.                          $this->assertIdentical($result, $success, 'Wartości sparsowane "rawList"
    nie są identyczne!');
226.                  }
227.          }
228.
229.          $r = new Applcation_Models_Ftp_FtpTest();
230.          $r->run(new TextReporter());
```