Aplikacje WWW

Ćwiczenia laboratoryjne

Projekt – aplikacja internetowa systemu ogłoszeń, rezerwacji usług lub przedmiotów

Wydział Inżynierii i Ekonomii Kierunek Informatyka

Celem projektu jest stworzenie aplikacji webowej o dowolnej treści. Projekt ma na celu zastosowanie wiedzy z zakresu programowania, inżynierii oprogramowania, grafiki, baz danych oraz tworzenia interfejsów użytkownika. Projekt nie jest pełnym procesem projektowym, mimo iż zawiera pewne jego elementy. Obejmuje aspekt techniczny skupiający się na narzędziach i językach programowania. Efektem końcowym ma być działająca aplikacja.

Aplikacja internetowa może być wykonana na wzór poniższego przykładu - systemu ogłoszeń, rezerwacji usług lub przedmiotów, który jednocześnie przedstawia podstawowe wymagania dla projektu przy zastosowaniu HTML, CSS, JS, PHP, mySQL (Xampp). Technologia ta nie jest narzucona, można skorzystać z innej - dodatkowe przykłady dla React i Django w osobym pliku.

Ocena na zaliczenie ćwiczeń będzie zależna od złożoności projektu, jego struktury i przejrzystości kodu. Ważnym kryterium jest odpowiedni i atrakcyjny wygląd aplikacji (zastosowanie stylów oraz grafiki).

Zakres projektu

1. **Planowanie projektu i analiza wymagań :**
    - Zdefiniowanie wymagań funkcjonalnych aplikacji.
    - Określenie technologii używanych w projekcie (w przykładzie HTML/JS dla frontendu, PHP dla backendu, MySQL dla bazy danych, GIMP do grafiki rastrowej).
2. **Projektowanie systemu :**
    - Wykorzystanie wzorca projektowego (np. MVC), architektury systemu SPA lub MPA lub tradycyjnej architektury trójwarstwowej
    - Projekt interfejsu użytkownika (makiety, layout w narzędziach takich jak Figma lub Adobe XD).
    - Stworzenie diagramów ERD (Entity Relationship Diagram) dla bazy danych.
3. **Implementacja:**
    - **Front-end**:
        - Stworzenie responsywnego interfejsu użytkownika
        - Użycie bibliotek do stylizacji (np. CSS, Bootstrap lub inne)

        - Zastosowanie grafiki i/lub animacji (np. Gimp)
        - Implementacja komponentów interfejsu użytkownika
    - **Back-end**:
        - Implementacja serwera aplikacji z wykorzystaniem PHP, Node.js/Express.js, Laravel, Django lub inne)
        - Obsługa autoryzacji i uwierzytelniania użytkowników (np. JWT, OAuth2).
        - Tworzenie logiki biznesowej (np. zarządzanie zadaniami, przypisywanie zadań do członków zespołu).
    - **Baza danych**:
        - Tworzenie schematu bazy danych (np. MySQL, MongoDB, PostgreSQL) i modeli danych.
        - Implementacja zapytań CRUD (Create, Read, Update, Delete) w aplikacji.
4. **Testowanie:**
    - Testowanie funkcji aplikacji (np. dodawanie zadań, przypisywanie zadań, komunikacja zespołowa).
5. **Dokumentacja:**
    - Dokumentacja techniczna (np. opis architektury systemu)
6. **Wdrożenie:**
    - Hostowanie aplikacji (np. Heroku, Vercel, AWS).
7. **Prezentacja projektu:**
    - Przygotowanie demonstracji działania aplikacji.

Proponowane technologie i narzędzia:

- **Front-end:** React.js, HTML, CSS (lub biblioteki np. Bootstrap, TailwindCSS), Webpack
- **Back-end:** Node.js, Express.js, JWT/OAuth2, Django
- **Baza danych:** XAMPP - MySQL, MongoDB lub PostgreSQL.
- **Grafika:** Gimp, Photoshop inne
- **Kontrola wersji:** Git (np. GitHub lub GitLab).
- **Hostowanie:** Heroku, Vercel lub AWS.
- **Inne narzędzia:** Figma, Adobe XD, Google Docs, Trello, Jira, GitHub

Efekt końcowy i warunki zaliczenia:

- **Działająca aplikacja webowa**.
- **Sprawozdanie zawierające krótkie opisy etapów tworzenia projektu wraz ze zrzutami**
- **Prezentacja projektu przedstawiająca proces realizacji oraz kluczowe rozwiązania**.

Podstawowe określenie funkcjonalności aplikacji www:

- Tworzenie ogłoszeń z opisem, kategorią, zdjęciami itp.
- Rezerwacja ogłoszeń przez użytkowników
- Logowanie i rejestracja użytkowników
- Opcjonalnie Zarządzanie rezerwacjami
- Opcjonalnie Panel administracyjny

Zastosowanie HTML, CSS, JS, PHP, mySQL (Xampp)

Wprowadzenie do technologii

Zapoznanie studentów z podstawowymi technologiami używanymi w projekcie.

1. **HTML & CSS** – Struktura i styl strony internetowej.
2. **JavaScript (JS)** – Interaktywność (np. dynamiczne ładowanie danych).
3. **PHP** – Obsługa logiki aplikacji i interakcji z bazą danych.
4. **MySQL** – Przechowywanie danych (np. ogłoszeń i rezerwacji).
5. **XAMPP** – Środowisko do uruchamiania PHP i MySQL lokalnie.

https://how2html.pl/

https://www.kurshtml.edu.pl/html/html5.html

https://www.kurshtml.edu.pl/temat.php?q=CSS3

https://phpkurs.pl/

https://podstawyjs.pl/spis-tresci/

Instalacja i konfiguracja środowiska

Przygotowanie narzędzi do pracy.

1. Pobranie i instalacja **XAMPP**: https://www.apachefriends.org/index.html
2. Uruchomienie **Apache i MySQL** w panelu XAMPP.
3. Wejście na stronę *http://localhost/* w przeglądarce – sprawdzenie, czy serwer działa.
4. Tworzenie folderu projektu *w htdocs* (np. C:\xampp\htdocs\rezerwacje).
5. Przygotowanie edytora kodu (Visual Studio Code, Sublime, Notepad+, Brackets inny)

Tworzenie bazy danych

Stworzenie struktury do przechowywania danych.

1. Wejście do **phpMyAdmin** (http://localhost/phpmyadmin/).
2. Stworzenie nowej bazy danych, np. rezerwacje_db.
3. Stworzenie tabel:
    - uzytkownicy (id, imie, email, haslo)
    - ogloszenia (id, tytul, opis, cena, data, status)
    - rezerwacje (id, id_ogloszenia, id_uzytkownika, data_rezerwacji)

Struktura plików projektu

Organizacja plików w projekcie.

```
rezerwacje/
├── index.php           # Strona główna
├── rejestracja.php     # Rejestracja użytkownika
├── logowanie.php       # Logowanie użytkownika
├── dashboard.php       # Panel użytkownika
├── dodaj_ogloszenie.php # Formularz dodawania ogłoszeń
├── rezerwacje.php      # Lista rezerwacji
├── styl.css            # Plik stylów
├── skrypty.js          # Plik JavaScript
├── config.php          # Połączenie z bazą danych
└── img/                # Folder na obrazy
```

Tworzenie interfejsu użytkownika (HTML + CSS)

Zbudowanie podstawowego wyglądu aplikacji.

1. Stworzenie *index.html* z podstawowym układem strony.
2. Dodanie pliku *styl.css* do stylizacji strony.
3. Zaprojektowanie prostego formularza rejestracji/logowania.

Przykład *index.html*:

```html
<!DOCTYPE html>
<html lang="pl">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Portal Rezerwacyjny</title>
    <link rel="stylesheet" href="styl.css">
```

```
</head>
<body>
    <h1>Witaj w Portalu Rezerwacyjnym</h1>
    <a href="rejestracja.php">Rejestracja</a> |
    <a href="logowanie.php">Logowanie</a> |
    <a href="lista.php">Ogłoszenia</a><br><br><hr><br>
    <img src="obr7.png" alt="">
</body>
</html>
```

Przykład *styl.css*:

```
body {
    font-family: Arial, sans-serif;
    background-color: #f4f4f4;
    text-align: center;
}

form {
    background: white;
    padding: 20px;
    width: 300px;
    margin: auto;
    border-radius: 5px;
}
```

Obsługa rejestracji i logowania (PHP + MySQL)

Umożliwienie użytkownikom rejestracji i logowania.

1. Tworzenie pliku *config.php* do połączenia z bazą danych.
2. Formularz rejestracji (*rejestracja.php*).
3. Przechowywanie danych użytkowników w bazie (uzytkownicy).
4. Logowanie użytkownika (*logowanie.php*) i obsługa sesji.

Przykład *config.php*:

```
<?php
$host = "localhost";
$user = "root";
$pass = "";
$db = "rezerwacje_db";

$conn = new mysqli($host, $user, $pass, $db);
if ($conn->connect_error) {
    die("Błąd połączenia: " . $conn->connect_error);
}
?>
```

Przykład *rejestracja.php*:

```
<form method="POST">
    Imię: <input type="text" name="imie" required><br>
    Email: <input type="email" name="email" required><br>
    Hasło: <input type="password" name="haslo" required><br>
    <button type="submit">Zarejestruj się</button>
</form>

<?php
include "config.php";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $imie = $_POST['imie'];
    $email = $_POST['email'];
    $haslo = password_hash($_POST['haslo'], PASSWORD_DEFAULT);

    $sql = "INSERT INTO uzytkownicy (imie, email, haslo) VALUES ('$imie',
'$email', '$haslo')";

    if ($conn->query($sql) === TRUE) {
        echo "Rejestracja udana! <a href='logowanie.php'>Zaloguj się</a>";
    } else {
        echo "Błąd: " . $conn->error;
    }
}
?>
```

Przykład *logowanie.php*:

```
<form method="POST">
    Email: <input type="email" name="email" required><br>
    Hasło: <input type="password" name="haslo" required><br>
    <button type="submit">Zaloguj się</button>
</form>

<?php
include "config.php";
session_start();

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $email = $_POST['email'];
    $haslo = $_POST['haslo'];

    $sql = "SELECT * FROM uzytkownicy WHERE email='$email'";
    $result = $conn->query($sql);

    if ($result->num_rows == 1) {
        $row = $result->fetch_assoc();
        if (password_verify($haslo, $row['haslo'])) {
```

```
            $_SESSION['user_id'] = $row['id'];
            header("Location: lista.php");
            exit();
        } else {
            echo "Nieprawidłowe hasło!";
        }
    } else {
        echo "Nie znaleziono użytkownika!";
    }
}
?>
```

Tworzenie systemu ogłoszeń

Umożliwienie użytkownikom dodawania i przeglądania ogłoszeń.

1. Formularz dodawania ogłoszenia (dodaj_ogloszenie.php).
2. Obsługa dodawania ogłoszeń do bazy (ogloszenia).
3. Strona z listą dostępnych ogłoszeń (dashboard.php).

Przykład *dodaj_ogloszenie.php*:

```
<form method="POST">
    Tytuł: <input type="text" name="tytul" required><br>
    Opis: <textarea name="opis" required></textarea><br>
    Cena: <input type="number" name="cena" required><br>
    <button type="submit">Dodaj</button>
</form>

<?php
include "config.php";
session_start();

if (!isset($_SESSION["user_id"])) {
    die("Błąd: Musisz być zalogowany, aby dodać ogłoszenie.");
}

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $tytul = $_POST['tytul'];
    $opis = $_POST['opis'];
    $cena = $_POST['cena'];
    $id_uzytkownika = $_SESSION["user_id"];

    $stmt = $conn->prepare("INSERT INTO ogloszenia (id_uzytkownika, tytul,
opis, cena, status) VALUES (?, ?, ?, ?, 'dostępne')");
    $stmt->bind_param("issd", $id_uzytkownika, $tytul, $opis, $cena);

    if ($stmt->execute()) {
        echo "Ogłoszenie dodane!";
```

```
    } else {
        echo "Błąd: " . $stmt->error;
    }

    $stmt->close();
    $conn->close();
}
?>
```

Przykład *dashboard.php*:

```
<?php
session_start();

include "config.php";

if (!isset($_SESSION['user_id'])) {
    header("Location: logowanie.php");
    exit();
}

$sql = "SELECT * FROM ogloszenia";
$result = $conn->query($sql);
?>

<!DOCTYPE html>
<html lang="pl">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Lista ogłoszeń</title>
</head>
<body>
    <h2>Lista ogłoszeń</h2>
    <a href="dodaj_ogloszenie.php">Dodaj ogłoszenie</a>
    <ul>
    <?php while ($row = $result->fetch_assoc()): ?>
        <li>
            <?php echo htmlspecialchars($row['tytul']); ?> - <?php echo
htmlspecialchars($row['cena']); ?> PLN
            <a href="rezerwacje.php?id=<?php echo $row['id'];
?>">Zarezerwuj</a>
        </li>
    <?php endwhile; ?>
    </ul>
</body>
</html>
```

Implementacja systemu rezerwacji

Umożliwienie użytkownikom rezerwacji ogłoszeń.

1. Dodanie opcji rezerwacji (rezerwacje.php).
2. Zapis rezerwacji w bazie (rezerwacje).
3. Obsługa statusu ogłoszenia (np. "zarezerwowane").

Przykład *rezerwacje.php*:

```php
<?php
include "config.php";
session_start();

if (!isset($_SESSION['user_id'])) {
    header("Location: logowanie.php");
    exit();
}

$id_ogloszenia = $_GET['id'];
$id_uzytkownika = $_SESSION['user_id'];

$sql = "INSERT INTO rezerwacje (id_ogloszenia, id_uzytkownika,
data_rezerwacji) VALUES ('$id_ogloszenia', '$id_uzytkownika', NOW())";
$conn->query($sql);

$sql_update = "UPDATE ogloszenia SET status='zarezerwowane' WHERE
id='$id_ogloszenia'";
$conn->query($sql_update);

echo "Rezerwacja dokonana!";
?>
```

Modyfikacja i uatrakcyjnienie layoutu strony

Dodanie sekcji, modyfikacje CSS, dodanie grafiki

1. Zmiana układu strony w index.html i podstronach przez wprowadzenie głównych sekcji layoutu
2. Stylowanie strony w pliku styl.css (można też zastosować Bootstrap).
3. Dodanie grafiki w celu uatrakcyjnienia aplikacji.

Wprowadzenie JavaScript (opcjonalnie)

Dodanie dynamicznych elementów i funkcji do strony w skrypty.js

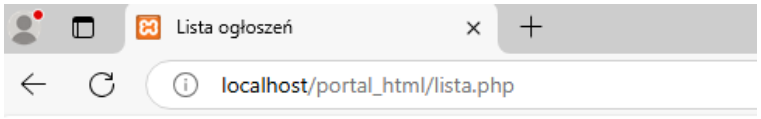
1. Walidacja formularzy rejestracji i logowania.
2. Dynamiczne filtrowanie ogłoszeń.
3. Obsługa AJAX do aktualizacji rezerwacji bez przeładowania strony.

Testowanie i wdrożenie

Sprawdzenie działania aplikacji i przygotowanie do publikacji.

1. Testowanie aplikacji (rejestracja, logowanie, dodawanie ogłoszeń, rezerwacje).
2. Opcjonalnie: Eksport bazy danych i tworzenie dokumentacji.
3. Opcjonalnie: Wdrożenie na serwer (np. hosting z obsługą PHP i MySQL).

Wygląd przykładowej aplikacji



Lista ogłoszeń

Dodaj ogłoszenie

- asaws - 2.00 PLN Zarezerwuj