Politechnika Warszawska

Ośrodek Kształcenia na Odległość

Wydział EiTI

**Dokumentacja – projekt nr 1**

Zaawansowane aplikacje internetowe

Dominik Kanthak

Studia magisterskie

Kierunek Informatyka

Indeks: 309478

[1. Treść zadania projektu nr 1 (PHP) 3](#_Toc56242195)

[2. Opis konfiguracji serwera i uruchomienia witryny 5](#_Toc56242195)

[2.1. Uruchomienie lokalne 5](#_Toc56242198)

[2.2. Konfiguracja serwera 7](#_Toc56242198)

[2.2. Dane aplikacji 9](#_Toc56242198)

[3. Opis struktury bazy danych 10](#_Toc56242195)

[4. Opis kodu stworzonej aplikacji 12](#_Toc56242195)

[5. Zrzuty ekranu demonstrujące zaimplementowane funkcjonalności 18](#_Toc56242195)

1. **Treść zadania projektu nr 1 (PHP)**

Zaimplementuj prosty system „pamiętnikowy” pozwalający odnotowywać na osi czasu wydarzenia mające postać odcinków czasu oraz prezentować je w estetyczny sposób. Potencjalne zastosowanie możesz wyobrazić sobie jako np. prezentacja dla klientów firmy szczegółowych informacji z historii jej rozwoju. Może to być także narzędzie o charakterze intranetowym, w którym odnotowywane są wydarzenia z życia danej społeczności. Potencjalnych zastosowań takiej aplikacji może być wiele – można zaproponować własne zastosowanie.

Specyfikacja funkcjonalna aplikacji (minimalne wymagania):

1. Wydarzenie rozumiemy jako odcinek na osi czasu charakteryzujący się następującymi cechami:
2. nazwa wydarzenia,
3. daty rozpoczęcia i zakończenia wydarzenia (dzień, miesiąc, rok),
4. opis tekstowy,
5. ilustracja graficzna,
6. kategoria wydarzenia.
7. Kategorie pozwalają powiązać ze sobą podobne wydarzenia i np. wyświetlić je w sposób charakterystyczny. Typ powinien mieć edytowalną nazwę oraz konfigurowalną jakąś cechę graficzną (np. kolor lub/i ikonę).
8. Są dwie grupy użytkowników: czytelnicy (niezalogowani) i administratorzy (zalogowany).
9. Użytkownicy niezalogowani mogą swobodnie przeglądać zawartość pamiętnika, ale bez możliwości edycji wpisów.
10. Administratorzy mogą zarówno przeglądać zawartość jak i ją edytować (operacje CRUD na wydarzeniach i kategoriach).
11. Każdy użytkownik może wydrukować pełną zawartość pamiętnika w czytelny sposób. Drukowanie powinno być zrealizowane przez odpowiednio przygotowany za pomocą CSS widok pozbawiony kontrolek sterujących aplikacji.
12. Zadbać o właściwe przechowywanie hasła dostępowego i możliwość jego zmiany przez zalogowanego użytkownika.
13. Zadbać o estetyczną formę prezentacji pamiętnika. Można skorzystać z bibliotek JS, które ułatwiają oznaczanie zdarzeń na osi czasu, można spróbować samodzielnej implementacji z użyciem odpowiednio przygotowanego arkusza CSS i elementów języka HTML5.
14. Kliknięcie w dane wydarzenie na osi czasu powinno wyświetlać pełen opis danego Elementu.

Interfejs aplikacji powinien być przejrzysty – zastosować style CSS3 do formatowania wyglądu elementów.

Wymagania techniczne:

* proszę wykorzystać technologie PHP, MariaDB / MySQL, HTML, JavaScript, CSS we współczesnych wersjach (w miarę możliwości),
* zaprojektować relacyjną bazę danych adekwatną do postawionego problemu: dobrać odpowiednie typy pól, zaplanować zależności pomiędzy tabelami, zastosować klucze obce,
* zachęcam do korzystania z frameworków, ale nie ma takiego wymogu; tym samym np. mechanizm uwierzytelniania użytkownika może zostać stworzony samodzielnie (można wykorzystywać materiały dołączone do kursu); użycie frameworka upraszcza pewne kwestie implementacyjne, ale wymaga zrozumienia i poznania samego frameworka – wybór leży w gestii Studenta,
* aplikacja powinna zapewniać choćby w minimalnym stopniu bezpieczeństwo przed włamaniami:

◦ zastosować zabezpieczenie przed SQL-Injection,

◦ hasła użytkowników muszą być przechowywane w bazie w formie hashy.

Stronę należy umieścić na dowolnym publicznie dostępnym serwerze w sieci Internet.

1. **Opis konfiguracji serwera i uruchomienia witryny**

Aplikacja została uruchomiona na platformie Railway.app. Jest to platforma hostingowa typu PaaS (Platform as a Service), która umożliwia łatwe wdrażanie, zarządzanie i skalowanie aplikacji internetowych oraz baz danych bez konieczności posiadania zaawansowanej infrastruktury serwerowej. Railway oferuje wsparcie dla różnych języków programowania, takich jak Node.js, Python, Go, czy PHP, a także frameworków takich jak Laravel, co ułatwia wdrożenie aplikacji na tej platformie.

Obraz zawierający zrzut ekranu, tekst

Opis wygenerowany automatycznie

Platforma ta została wybrana ze względu na mało skomplikowaną ciągłą procedurę wdrożenia aplikacji oraz darmową cenę. Oferuje ona konto TRAIL, które po poprawnym zarejestrowaniu i połączeniu z kontem Github (warunek konieczny do odsiania kont spamerskich) pozwala na wykorzystanie 5$ środków. Udostępnia ona wtedy zasoby, które pokazane zostały powyżej. Jednocześnie nie wymagana przy tym podpinania kart płatniczych. Dla czasowych jak i zasobowych wymagań projektu platforma ta wydaje się wystarczającym rozwiązaniem. W celu dodatkowego zaoszczędzenia środków – platforma pozwala uśpić udostępnione serwisy.

**2.1. Uruchomienie lokalne**

1. Pobranie oraz zainstalowanie PHP w domyślnej lokalizacji C:/php (to istotne - inna lokalizacja może powodować problemy).

Należy ustawić ścieżkę instalacji jako kolejny parametr zmiennej globalnej PATH.

Dodatkowo, w pliku php.ini znajdującym się w lokalizacji instalacji należy odkomentować poniższe rozszerzenia pakietu PHP:

* extension=fileinfo
* extension=mbstring
* extension=mysqli
* extension=openssl
* extension=pdo\_mysql

1. Należy pobrać oraz zainstalować w domyślnej lokalizacji program Composer – system zarządzania pakiętami dla języka PHP
2. Należy pobrać i zainstalować pakiet Node.js oraz npm
3. Należy pobrać oraz zainstalować wybraną bazę danych – MySQL lub MariaDB

W momencie instalacji możemy wybrać domyślne wartości nazwy użytkownika root, hasła użytkownika root oraz port na której będzie działała baza danych lub podać też swoje wartości. Należy zachować te wartości, gdyż będą one potrzebne podczas dalszej konfiguracji.

1. W kolejnym korku należy pobrać kod źródłowy z repozytorium Github oraz przejść do folderu z aplikacją:

git clone <https://github.com/kanthakdominik/timeline_php.git>

1. Wykonać poniższą komendę, aby umożliwić uruchomienie skryptu:

Set-ExecutionPolicy -ExecutionPolicy Unrestricted -Scope CurrentUser

1. Uruchomić skrypt z poziomu folderu z kodem źrodłowym:

.\run.ps1

Skrypt ten służy do:

* automatycznej konfiguracji aplikacji
* zbudowania aplikacji
* pobrania zależności aplikacji
* zbudowania bazy danych
* uzupełnienia bazy danych przykładowymi danymi
* ewentualnego wyczyszczenia bazy danych
* uruchomienia lokalnie aplikacji webowej

Podczas działania w skrypcie pojawi się możliwość uzupełnienia danych bazy danych własnymi parametrami, które można było podać podczas instalacji bazy. Jest także opcja wyczyszczenia bazy danych do wyboru.

**Aplikacja lokalnie działa pod adresem:**

<http://localhost:8000>

**2.2. Konfiguracja serwera**

1. Utworzenie nowego konto, najlepiej powiązać je z kontem Github
2. Utworzyć nowy projekt na platformie Railway
3. Stworzenie serwisu bazy danych – do wyboru jest PostgreSQL, MongoDB, Redis oraz MySQL – które został zastosowany
4. Stworzenie serwisu aplikacji – poprzez wybranie określonego repozytorium z podpiętego konta Github. Po tym kroku Railway sam wykryje, zbuduje i wdroży aplikację (kontener z aplikacją)
5. Konfiguracja – należy uzupełnić zmienne w serwisie z aplikacją, najlepiej poprzez skopiowanie ich pliku .env z lokalnego środowiska. Następnie należy uzupełnić dane dostępowe bazy danych (które można podejrzeć w serwisie bazy danych), nazwę aplikacji, udostępniany port.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

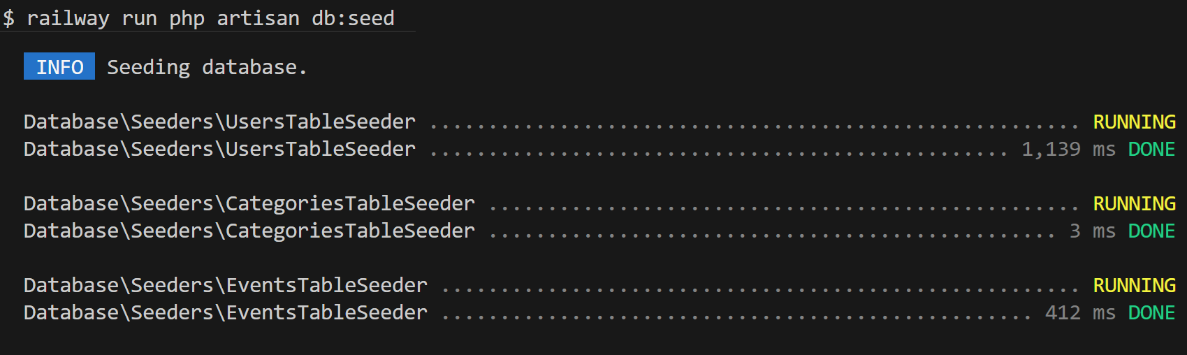
Widok ze zmiennymi w serwisie z wdrażaną aplikacją.

1. Ewentualna konfiguracja innych ustawień – jak automatyczną budowę i wdrażanie po każdej zmianie wykrytej w repozytorium kodu, czy możliwość usypiania serwisów w momentach bezczynności.
2. Połączenie się poprzez dowolny terminal do Railway Command Line Interface:

* Zainstalowanie narzedzi Railway CLI poprzez npm: ‘npm i -g @railway/cli’
* Zalogowanie się na konto platforym: ‘railway login’
* Link – wybranie projektu: ‘railway link’
* Wykonanie komend inicjujących projekt poprzez Railway CLI:
  + ‘railway run composer install‘ - instacja zależności
  + ‘railway run php artisan optimize:clear’
  + ‘railway run php artisan migrate’ – migracja I budowa schemata bazy danych



* + ‘railway run php artisan db:seed’ – generowanie przykładowych danych i kont użytkowników



**2.3. Dane aplikacji**

**Aplikacja działa pod adresem:**

<https://timeline-app.up.railway.app/>

W ramach testowych oraz pokazania treści aplikacja jest uzupełniona danymi przedstawiającymi przykładowe wydarzenia oraz kategorie. Są także stworzone 2 konta administratorów:

|  |  |
| --- | --- |
| **Email** | **Hasło** |
| admin@timeline.com | Password123! |
| admin2@timeline.com | Password123! |

**Repozytorium kodu:**

<https://github.com/kanthakdominik/timeline_php>

1. **Opis struktury bazy danych**

W projekcie zdecydowano się na wykorzystanie relacyjnej bazy danych, opartej na silniku MySQL (środowisko produkcyjne) oraz MariaDb (środowisko developerskie).

Do tworzenia schematu bazy danych oraz do uzupełniania przykładowymi danymi stworzonej bazy danych są używane gotowe funkcjonalności frameworka Laravel.

Podgląd na strukturę bazy danych w programie Dbeaver:

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

Przedstawione tabele można podzielić na dwie grupy: zaprojektowane ręcznie oraz tabele wygenerowane przez framework Laravel. Skupmy się więc na tabelach zaprojektowanych ręcznie: **events, categories** oraz **users.**

Tabela **events** – tabela ta przechowuje informacje o wydarzeniach, składa się z pól:

* id – typ bigint(20) unsigned auto increment - klucz główny
* name - typ varchar(50) NOT NULL
* start\_date – typ date NOT NULL
* end\_date – typ date NOT NULL
* description – typ text
* image – typ mediumblob
* create\_at – typ timestamp
* updated\_at – typ timestamp
* category\_id – typ bigin(20) unsigned NOT NULL - klucz obcy do tabeli categories

Tabela **categories** – tabela ta przechowuje informacje o kategoriach wydarzeń, składa się z pól:

* id – typ bigint(20) unsigned auto increment - klucz główny
* name - typ varchar(50) NOT NULL
* color– typ varchar(50)
* create\_at – typ timestamp
* updated\_at – typ timestamp

Tabela **users** – tabela ta przechowuje informacje o użytkownikach – administratorach aplikacji, składa się z pól:

* id – typ bigint(20) unsigned auto increment - klucz główny
* name - typ varchar(255) NOT NULL
* email - typ varchar(255) NOT NULL
* email\_verified\_at – typ timestamp
* password – varchar(255) NO NULL
* remember\_token – varchar(100)
* create\_at – typ timestamp
* updated\_at – typ timestamp

1. **Opisu kodu stworzonej aplikacji**

Aplikacja została napisana w języku **PHP**. Wykorzystany został także framework **Laravel** – jeden z najpopularniejszych narzędzi do tworzenia aplikacji webowych w języki PHP. Opiera się on na architekturze MVC (Model-View-Controller) oraz oferuje wygodny sposób na obsługę routingu pomiędzy żądaniami webowymi i obsługę zapytań do bazy danych.

Po stronie front-end’owej wykorzystywany jest **Blade** Engine – silnik szablonów -również zawarty we frameworku Laravel. Umożliwia tworzenie dynamicznych widoków HTML oraz wiąże on zmienne, dane kodu PHP z kodem HTML.

Oprócz wyżej wymienionych technologii, w projekcie wykorzystano również bibliotekę **Bootstrap**. Pozwala on na wykorzystanie gotowych, wystylizowanych komponentów (wykorzystujących HTML, CSS, Javascript). Z jego pomocą tworzony jest też wizualny komponent linii czasu w projekcie.

**Kluczowe elementy projektu:**

* **Migrations** – znajdują się tutaj pliki odpowiedzialne za migrację oraz tworzenie schematu bazy danych. Opisują one poprzez kod PHP nazwy tabel, nazwy kolumn, ich typy oraz powiązania między nimi.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Obraz zawierający zrzut ekranu, tekst

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

* **Seeders** – znajdują się tutaj pliki odpowiedzialne za generowanie przykładowych danych i kont użytkowników znajdujących się w bazie danych.

Obraz zawierający zrzut ekranu, tekst, Czcionka, linia

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

* **Models** – znajdują się tutaj pliki odpowiedzialne za klasy modelu aplikacji – klasy będące abstrakcją tabel, danych z bazy danych
* **Routes/web.php** – plik odpowiedzialny za definiowanie tras dla aplikacji webowej, - ścieżki URL oraz odpowiadające im kontrolery. Umożliwiają także dodanie warstwy pośredniej (middleware) do dodatkowe obsługi zapytań.

**Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie**

* **Controllers** – znajdują się tutaj klasy odpowiedzialne za odbiór i przetwarzanie żądań. Obsługują logikę aplikacji, komunikują się z modelem a także przekazują dane do widoków. W nich także zaimplementowano walidację danych wejściowych po stronie backendu oraz autentykację użytkownika.

**Obraz zawierający tekst, zrzut ekranu, oprogramowanie

Opis wygenerowany automatycznie**

Przykładowy endpoint który obsługuję usuwanie kategorii – w projekcie założono, że można usunąć tylko taką kategorię, dla której nie istnieje żadne wydarzenie.

* **Middleware/HttpRedirect.php** oraz **Providers/AppServiceProvider.php** – klasy służące do konfiguracji bezpiecznego protokołu przesyłu żądań w aplikacji – protokołu HTTPS. Wymuszają one użycie HTTPS zamiast domyślnego HTTP.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie,

* **Views** – folder zawierający wszystkie widoki generowane w aplikacji:
  + **welcome.blade.php** – główny plik strony aplikacji. Zawiera w sobie wstrzykiwane pozostałe widoki oraz konfiguruje niezbędne importy.
  + **events-cards.blade.php** – komponent osi czasu wraz kartami, w których wyświetlane są w kolejności wydarzenia. Każda taka karta posiada także możliwość edycji czy usunięcia dla zalogowanego użytkownika.
  + **auth/register.blade.php** - strona rejestracji, zawiera walidację wprowadzanych danych po stronie frontendu, a także pozwala wyświetlać zwrócone błędy walidacji ze strony backendu
  + **auth/login.blade.php**  - strona logowania, zawiera walidację wprowadzanych danych po stronie frontendu, a także pozwala wyświetlać zwrócone błędy walidacji ze strony backendu
  + **auth/change-password.blade.php**  - strona zmiany hasła zalogowanego użytkownika, zawiera walidację wprowadzanych danych po stronie frontendu, a także pozwala wyświetlać zwrócone błędy walidacji ze strony backendu
  + **bar/main-bar.blade.php** – komponent widoku głównego paska strony. Zawiera logo strony, przyciski służące do rejestracji, logowania, zmiany hasła oraz wylogowania. Wyświetla także nazwę obecnie zalogowanego użytkownika.
  + **bar/categories-bar.blade.php** – komponent widoku paska z wyborem kategorii. Zawiera także przycisk służący do podglądu wydruku strony oraz przycisk rozwijający / zwijający karty z wydarzeniami. Dodatkowo dla zalogowanego użytkownika wyświetla także przyciski służące do edycji koloru, edycji nazwy oraz usunięcia kategorii.
  + **bar/setting-bar.blade.php** – komponent widoku paska widocznego jedynie dla zalogowanego użytkownika. Wyświetla przyciski służące do dodawania nowego wydarzenia lub dodania nowej kategorii.
  + **modal/add-category-modal.blade.php** – modal pozwalający wprowadzić dane podczas tworzenia nowej kategorii.
  + **modal/add-event-modal.blade.php** – modal pozwalający wprowadzić dane podczas tworzenia nowego wydarzenia.
  + **modal/edit-event-modal.blade.php** – modal pozwalający wprowadzić dane podczas edycji wydarzenia.
  + **modal/chage-category-name-modal.blade.php** – modal pozwalający wprowadzić dane podczas edycji nazwy kategorii.
  + **modal/chage-category-color-modal.blade.php** – modal pozwalający wprowadzić dane podczas edycji koloru kategorii.
* **custom.js** – plik zawierający funkcje napisane w języku Javascript obsługujące logikę strony frontednowej aplikacji. Odpowiada za zachowanie na dane akcje użytkownika na stronie.
* **custom.css** – plik opisujący kaskadowe arkusze stylów – definiujące wygląd aplikacji. Oprócz wykorzystywania wbudowanych stylów z biblioteki Boostrap, aplikacja korzysta także z własnych stylów lub nadpisuje te z biblioteki Boostrap**.**

**Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie**

Przykładowa klasa ccs, w tym przypadku odpowiada ona za widok wydruku strony w trybie podglądu wydruku.

* **Validation.php** – plik zawierający treści komunikatów aplikacji wyświetlanych użytkownikowi w rożnych sytuacjach (np. podczas wyświetlania błędów walidacji danych wejściowych) w języku polskim.

1. **Zrzuty ekranu demonstrujące zaimplementowane funkcjonalności**

**Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Strona internetowa

Opis wygenerowany automatycznie**

Strona główna

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Strona internetowa

Opis wygenerowany automatycznie

Strona główna (zalogowany użytkownik)

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Ikona komputerowa

Opis wygenerowany automatycznieStrona rejestracji

Obraz zawierający tekst, zrzut ekranu, oprogramowanie

Opis wygenerowany automatycznieStrona logowania

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Ikona komputerowa

Opis wygenerowany automatycznieStrona zmiany hasła (zalogowany użytkownik)

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Ikona komputerowa

Opis wygenerowany automatycznie

Strona wydruku

Obraz zawierający tekst, logo, zrzut ekranu, Marka

Opis wygenerowany automatycznie

Rozwinięte karty wydarzeń

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Strona internetowa

Opis wygenerowany automatycznie

Przefiltrowane karty wydarzeń po kategorii muzycznej

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Ikona komputerowa

Opis wygenerowany automatycznie

Edycja wydarzenia

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Ikona komputerowa

Opis wygenerowany automatycznie

Dodawanie nowego wydarzenia

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

Dodawanie nowej kategorii

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

Zmiana koloru kategorii

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

Zmiana nazwy kategorii