

Politechnika Warszawska  
Ośrodek Kształcenia na Odległość  
Wydział EiTl

# **Dokumentacja – projekt nr 1**

Zaawansowane aplikacje internetowe

Dominik Kanthak  
Studia magisterskie  
Kierunek Informatyka  
Indeks: 309478

1.	Treść zadania projektu nr 1 (PHP) .....	3
2.	Opis konfiguracji serwera i uruchomienia witryny. ....	<b>Błąd! Nie zdefiniowano zakładki.</b>
2.1.	konfiguracja serwera.....	<b>Błąd! Nie zdefiniowano zakładki.</b>
2.2.	Dane aplikacji .....	7
3.	Opis struktury bazy danych.....	8
4.	Opis kodu stworzonej aplikacji .....	10
5.	Zrzuty ekranu demonstrujące zaimplementowane funkcjonalności .....	<b>Błąd! Nie zdefiniowano zakładki.</b>

## 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:
  1. nazwa wydarzenia,
  2. daty rozpoczęcia i zakończenia wydarzenia (dzień, miesiąc, rok),
  3. opis tekstowy,
  4. ilustracja graficzna,
  5. kategoria wydarzenia.
2. 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).
3. Są dwie grupy użytkowników: czytelnicy (niezalogowani) i administratorzy (zalogowany).
4. Użytkownicy niezalogowani mogą swobodnie przeglądać zawartość pamiętnika, ale bez możliwości edycji wpisów.
5. Administratorzy mogą zarówno przeglądać zawartość jak i ją edytować (operacje CRUD na wydarzeniach i kategoriach).
6. Każdy użytkownik może wydrukować pełną zawartość pamiętnika w czytelny sposób. Drukowanie powinno być zrealizowane przez odpowiednio przygotowany za pomocą CSS widok pozbawiony kontrolek sterujących aplikacją.
7. Zadbaj o właściwe przechowywanie hasła dostępowego i możliwość jego zmiany przez zalogowanego użytkownika.
8. Zadbaj 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.
9. 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.

## 2. 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.

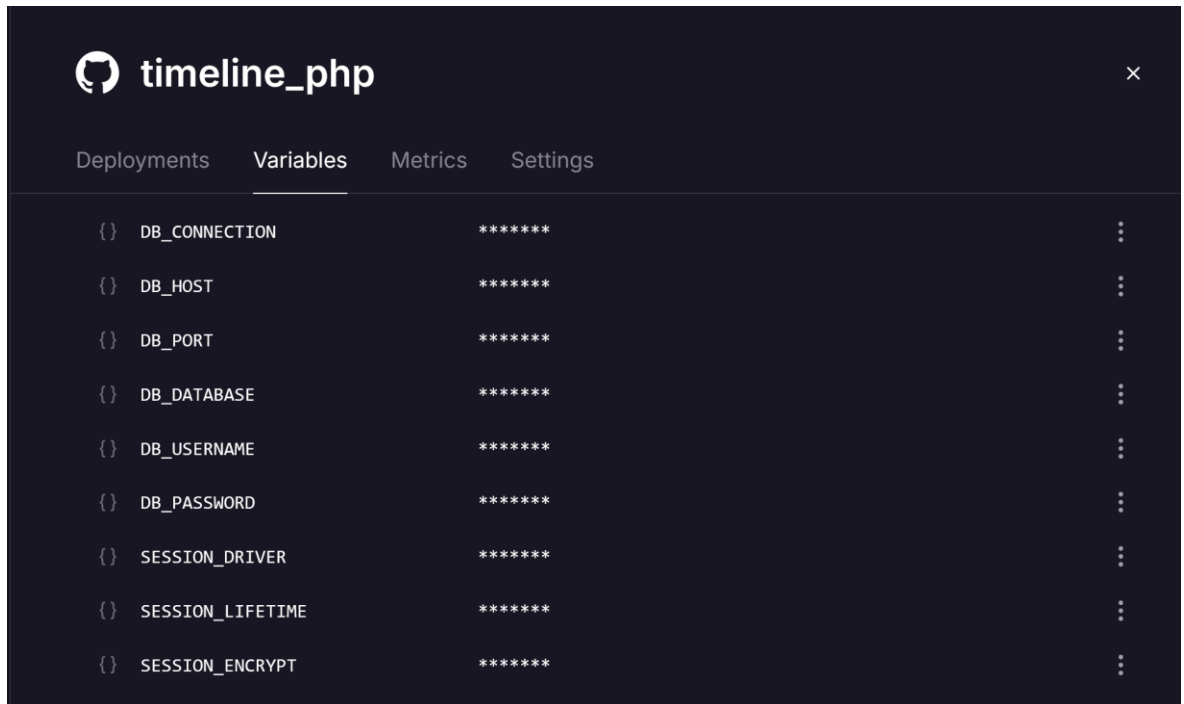


Platforma ta została wybrana ze względu na mało skomplikowaną ciągłą procedurę wdrożenia aplikacji oraz darmową cenę. Oferuje ona konto TRAIL, które po poprawnym zarejestrowaniu i połączeniu z kontem Github (warunek konieczny do odsiania kont spamerskich) pozwala na wykorzystanie 5\$ środków. Udostępnia 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. Konfiguracja serwera

1. Utworzenie nowego konta, 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 podpętego 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.



Widok ze zmiennymi w serwisie z wdrażaną aplikacją.

6. Ewentualna konfiguracja innych ustawień – jak automatyczną budowę i wdrażanie po każdej zmianie wykrytej w repozytorium kodu, czy możliwość usypiania serwisów w momentach bezczynności.
7. Połączenie się poprzez dowolny terminal do Railway Command Line Interface:
  - Zainstalowanie narzędzi Railway CLI poprzez npm: 'npm i -g @railway/cli'
  - Zalogowanie się na konto platformy: 'railway login'
  - Link – wybranie projektu: 'railway link'
  - Wykonanie komend inicjujących projekt poprzez Railway CLI:
    - 'railway run composer install' - instalacja zależności
    - 'railway run php artisan optimize:clear'
    - 'railway run php artisan migrate' – migracja i budowa schemata bazy danych

```
$ railway run php artisan migrate
INFO Preparing database.
Creating migration table ..... 372.17ms DONE
INFO Running migrations.
0001_01_01_000000_create_users_table ..... 274.93ms DONE
0001_01_01_000001_create_cache_table ..... 22.38ms DONE
0001_01_01_000002_create_jobs_table ..... 146.69ms DONE
2024_10_10_180741_create_categories_table ..... 33.08ms DONE
2024_10_10_180758_create_events_table ..... 119.74ms DONE
2024_10_16_191818_create_personal_access_tokens_table ..... 52.99ms DONE
```

- 'railway run php artisan db:seed' – generowanie przykładowych danych i kont użytkowników

```
$ railway run php artisan db:seed

INFO Seeding database.

Database\Seeders\UsersTableSeeder ..... RUNNING
Database\Seeders\UsersTableSeeder ..... 1,139 ms DONE

Database\Seeders\CategoriesTableSeeder ..... RUNNING
Database\Seeders\CategoriesTableSeeder ..... 3 ms DONE

Database\Seeders\EventsTableSeeder ..... RUNNING
Database\Seeders\EventsTableSeeder ..... 412 ms DONE
```

## 2.2. 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:

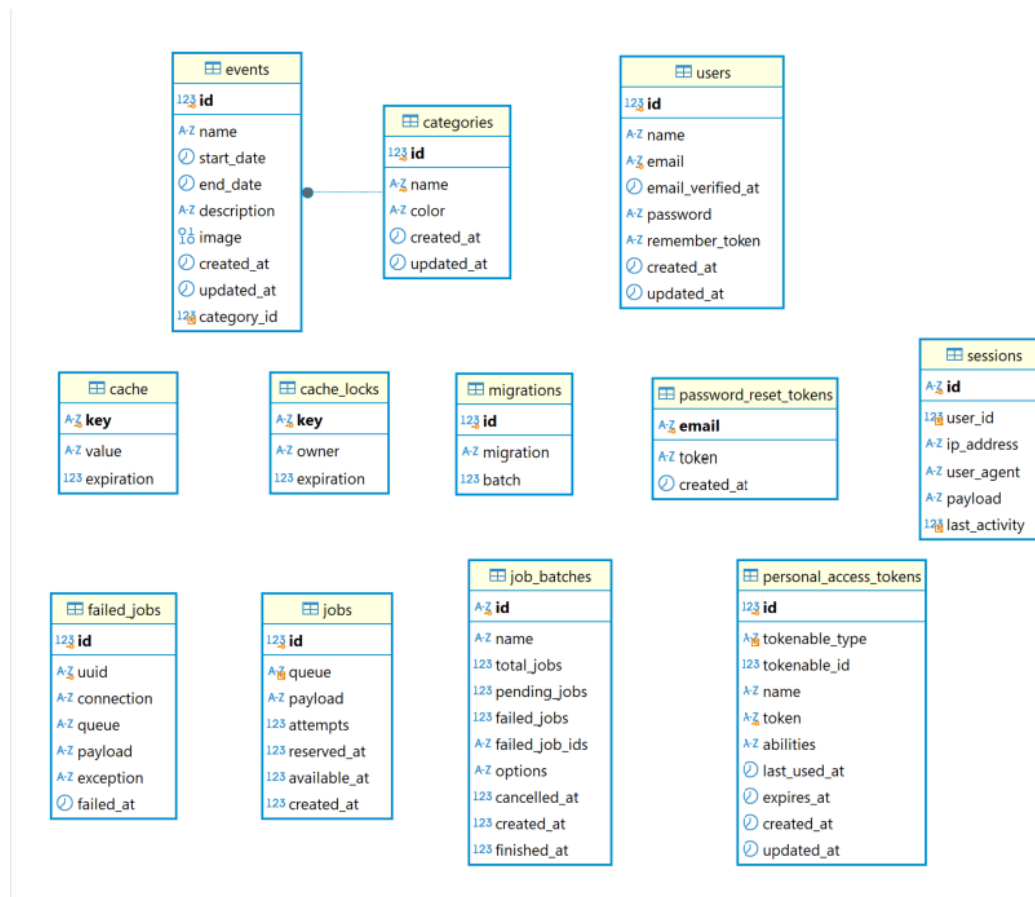
Email	Hasło
admin@timeline.com	Password123!
admin2@timeline.com	Password123!

### 3. 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:



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 begin(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

## 4. Opisu kodu stworzonej aplikacji

Aplikacja została napisana w języku **PHP**. Wykorzystany został także framework **Laravel** – jeden z najpopularniejszych narzędzi do tworzenia aplikacji webowych w języku 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.

```
public function up(): void
{
    Schema::create(table: 'categories', callback: function (Blueprint $table):
    {
        $table->id();
        $table->string(column: 'name', length: 50)->unique();
        $table->string(column: 'color', length: 50)->nullable();
        $table->timestamps();
    });
}
```

```
Schema::create(table: 'categories', callback: function (Blueprint $table): void {
    $table->id();
    $table->string(column: 'name', length: 50)->unique();
    $table->string(column: 'color', length: 50)->nullable();
    $table->timestamps();
});
```

```
Schema::create(table: 'users', callback: function (Blueprint $table): void {
    $table->id();
    $table->string(column: 'name');
    $table->string(column: 'email')->unique();
    $table->timestamp(column: 'email_verified_at')->nullable();
    $table->string(column: 'password');
    $table->rememberToken();
    $table->timestamps();
});
```

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

```
DB::table(table: 'categories')->insert(values: [
    ['name' => 'sportowa', 'color' => '#47b39d', 'created_at' => now(), 'updated_at' => now()],
    ['name' => 'muzyczna', 'color' => '#ffc153', 'created_at' => now(), 'updated_at' => now()],
    ['name' => 'kulturalna', 'color' => '#eb6b56', 'created_at' => now(), 'updated_at' => now()],
    ['name' => 'edukacyjna', 'color' => '#b05f6d', 'created_at' => now(), 'updated_at' => now()],
    ['name' => 'rekreacyjna', 'color' => '#9969c7', 'created_at' => now(), 'updated_at' => now()]
]);
```

```
DB::table(table: 'events')->insert(values: [
    [
        'name' => 'Mistrzostwa Polski w Piłce Nożnej',
        'start_date' => '2024-03-10',
        'end_date' => '2024-03-17',
        'description' => 'Najlepsze drużyny piłkarskie z całego kraju rywalizują o tytuł mistrza Polski',
        'image' => file_get_contents(filename: storage_path(path: 'app/public/logos/1.png')),
        'created_at' => now(),
        'updated_at' => now(),
        'category_id' => $sportCategoryId
    ],
]);
```

```
DB::table(table: 'users')->insert(values: [
    [
        'name' => 'Administrator',
        'email' => 'admin@timeline.com',
        'password' => Hash::make(value: 'Password123!'),
        'created_at' => now(),
        'updated_at' => now(),
    ],
]);
```

- **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.

```
Route::get(uri: '/', action: [EventController::class, 'index'])->name(name: 'home');

Route::resource(name: '/events', controller: EventController::class)->middleware(middleware: 'auth');

Route::put(uri: '/categories/update-name', action: [CategoryController::class, 'updateName'])->name(name: 'cate');
Route::put(uri: '/categories/update-color', action: [CategoryController::class, 'updateColor'])->name(name: 'ca');
Route::delete(uri: '/categories/{category}', action: [CategoryController::class, 'destroy'])->name(name: 'categ');
Route::post(uri: '/categories', action: [CategoryController::class, 'store'])->name(name: 'categories.store');

Route::get(uri: '/register', action: [UserController::class, 'showRegistrationForm']);
Route::post(uri: '/register', action: [UserController::class, 'register'])->name(name: 'register');

Route::get(uri: '/login', action: [UserController::class, 'showLoginForm']);
Route::post(uri: '/login', action: [UserController::class, 'login'])->name(name: 'login');

Route::get(uri: '/password/change', action: [UserController::class, 'showChangePasswordForm'])->name(name: 'pas');
Route::post(uri: '/password/change', action: [UserController::class, 'changePassword'])->name(name: 'password.c');

Route::post(uri: '/logout', action: [UserController::class, 'logout'])->name(name: 'logout');
```

- **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.

```
4 references | 0 overrides
public function destroy(Category $category): RedirectResponse
{
    // Check if any events are associated with this category
    $eventsCount = Event::where(column: 'category_id', operator: $category->id)->count();

    if ($eventsCount > 0) {
        return redirect()->back()->withErrors(provider: ['category' => 'Nie można usunąć']);
    }

    $category->delete();
    return redirect()->back()->with(key: 'success', value: 'Kategoria usunięta pomyślnie.');
```

Przykładowy endpoint który obsługuje 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.

```

0 references | 0 implementations
class AppServiceProvider extends ServiceProvider
{
    0 references | 0 overrides
    public function boot(): void
    {
        if (env(key: 'APP_ENV') === 'production') {
            URL::forceScheme(scheme: 'https');
        }
    }
}

```

```

class HttpRedirect
{
    public function handle(Request $request, Closure $next)
    {
        if (!$request->secure() && App::environment('production')) {
            return redirect()->secure($request->getRequestUri());
        }

        return $next($request);
    }
}

```

- **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/change-category-name-modal.blade.php** – modal pozwalający wprowadzić dane podczas edycji nazwy kategorii.
  - **modal/change-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 Bootstrap, aplikacja korzysta także z własnych stylów lub nadpisuje te z biblioteki Bootstrap.

```

@media print {
  .no-print {
    display: none !important;
  }

  .print-view {
    display: block;
  }

  .card-no-break {
    page-break-inside: avoid;
  }

  #toggleLabel::before {
    content: '' !important;
  }

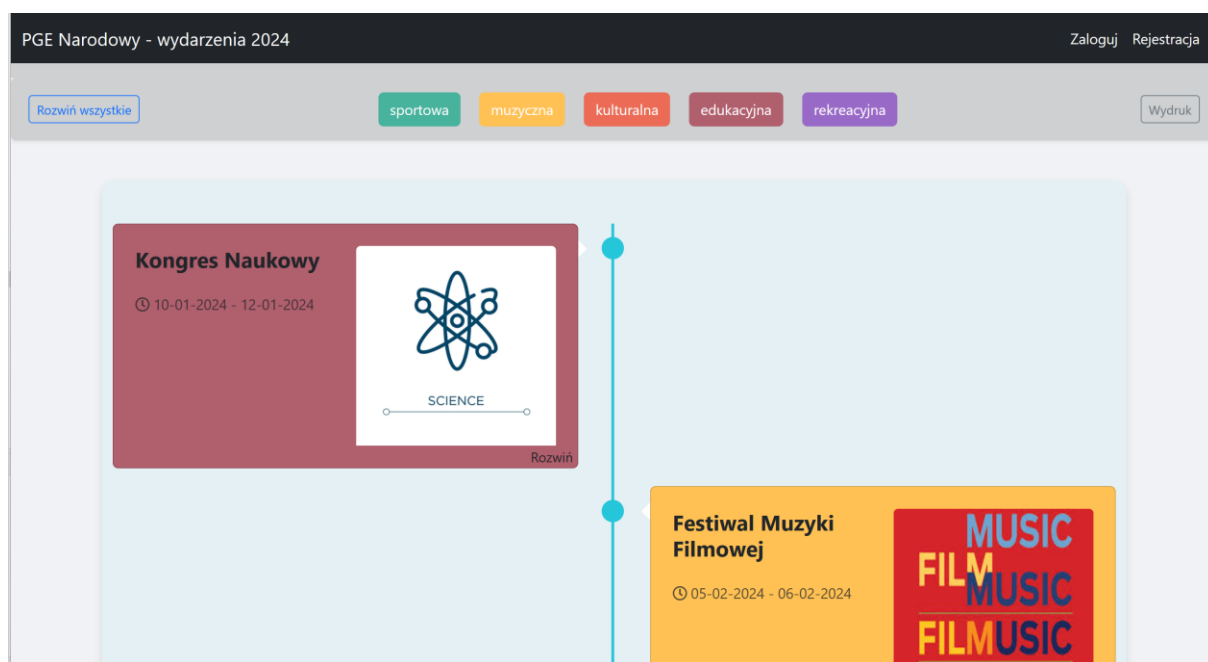
  .timeline-2 {
    width: 100% !important;
    padding: 0 !important;
    margin-bottom: 20px;
  }
}

```

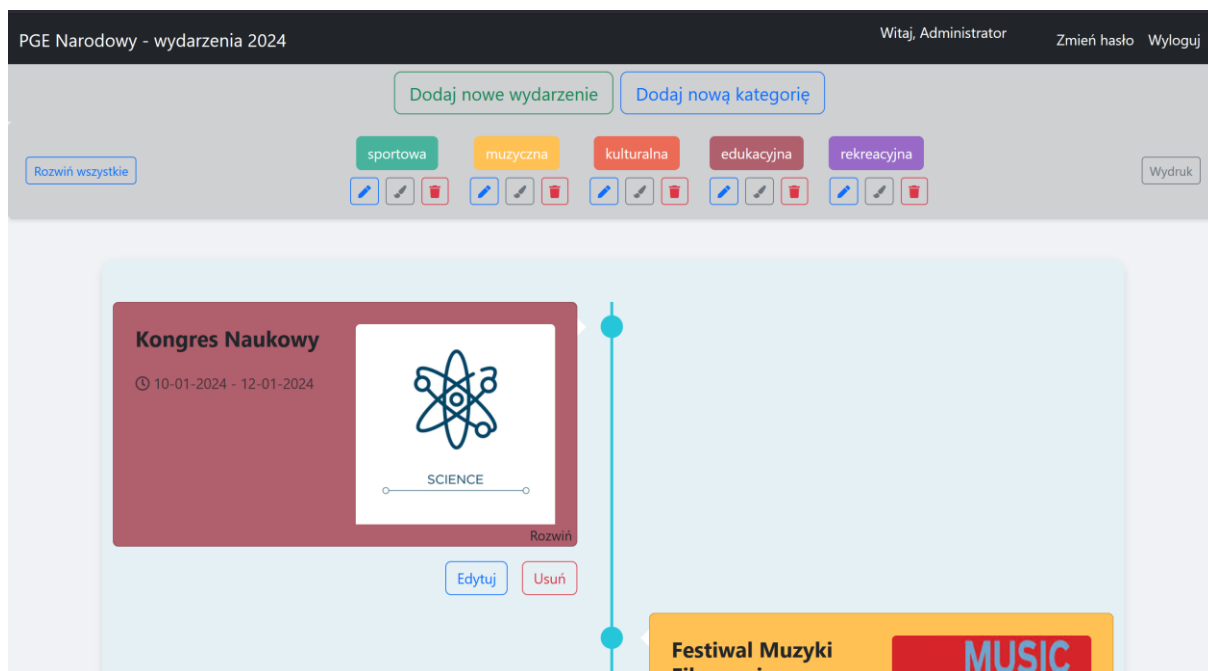
Przykładowa klasa ccs, w tym przypadku odpowiada ona za widok wydruku strony w trybie podglądu wydruku.

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

## 5. Zrzuty ekranu demonstrujące zaimplementowane funkcjonalności




Strona główna



Strona główna (zalogowany użytkownik)





Rejestracja

Nazwa użytkownika

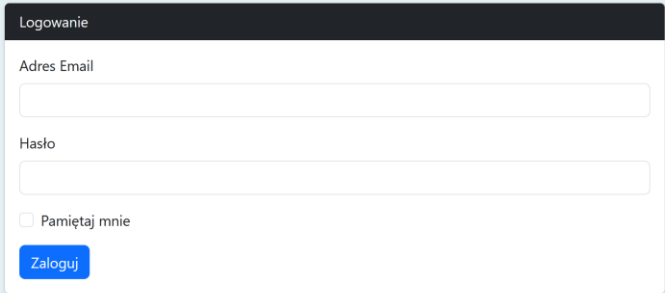
Adres Email

Hasło

Potwierdź hasło

[Zarejestruj](#)

## Strona rejestracji



Logowanie

Adres Email

Hasło

☐ Pamiętaj mnie

[Zaloguj](#)

## Strona logowania

Zmień hasło

Aktualne hasło

Nowe hasło

Potwierdź nowe hasło

Zmień hasło

Strona zmiany hasła (zalogowany użytkownik)

PGE Narodowy - wydarzenia 2024

Rozwiń wszystkie

Kongres Naukowy

10-01-2024 - 12-01-2024

Spotkanie naukowców z różnych dziedzin, prezentujące najnowsze osiągnięcia i badania. Wykłady, panele dyskusyjne i warsztaty. Idealna okazja do wymiany wiedzy i doświadczeń.

Festiwal Muzyki Filmowej

05-02-2024 - 06-02-2024

Koncerty orkiestr symfonicznych, wykonujących znane utwory filmowe. Niezapomniana podróż przez świat muzyki filmowej. Idealne wydarzenie dla miłośników kina i muzyki.

Mistrzostwa Polski w Piłce Nożnej

10-03-2024 - 17-03-2024

Najlepsze drużyny piłkarskie z całego kraju rywalizują o tytuł mistrza Polski. Niezapomniane emocje i zacięta walka na boisku. Kibice mogą liczyć na wiele emocjonujących momentów i niespodziewane zwroty akcji.

1 of 9

05/11/2024, 10:24

Print

9 sheets of paper

Destination

Samsung ML-1860 Series

Copies

1

Orientation

Portrait Landscape

Pages

All

Color mode

Black and white

Fewer settings

Paper size

A4

Scale

Fit to page width

Scale 100

Pages per sheet

Print Cancel

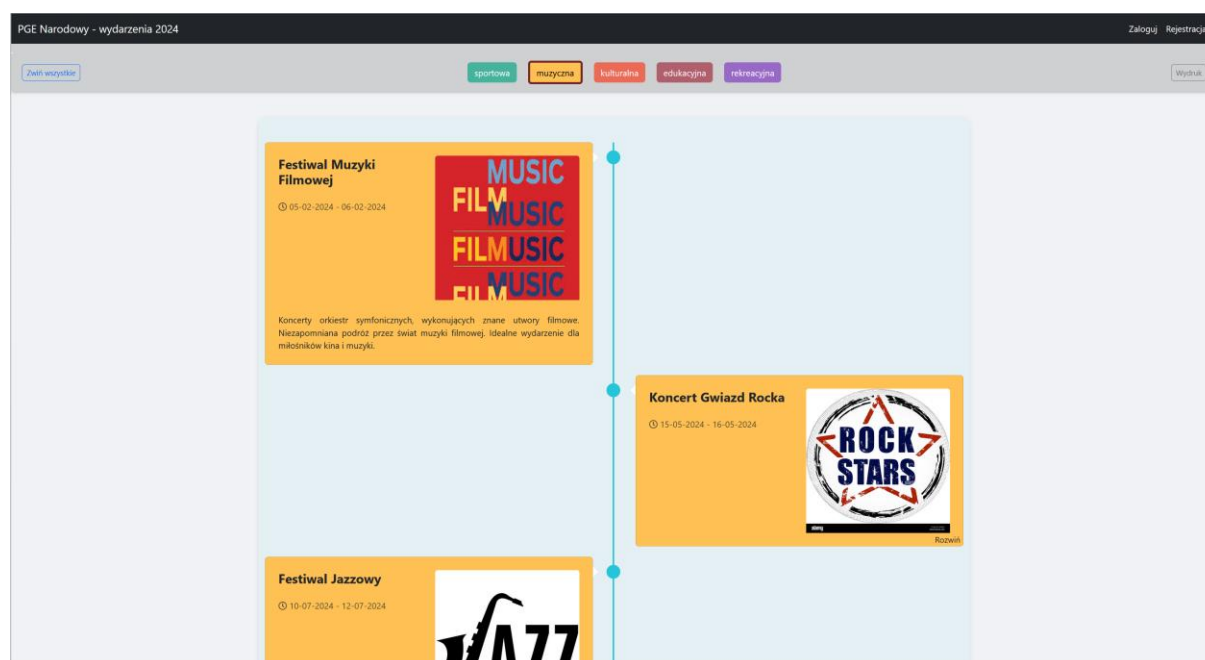
Zaloguj Rejestracja

Wydruk

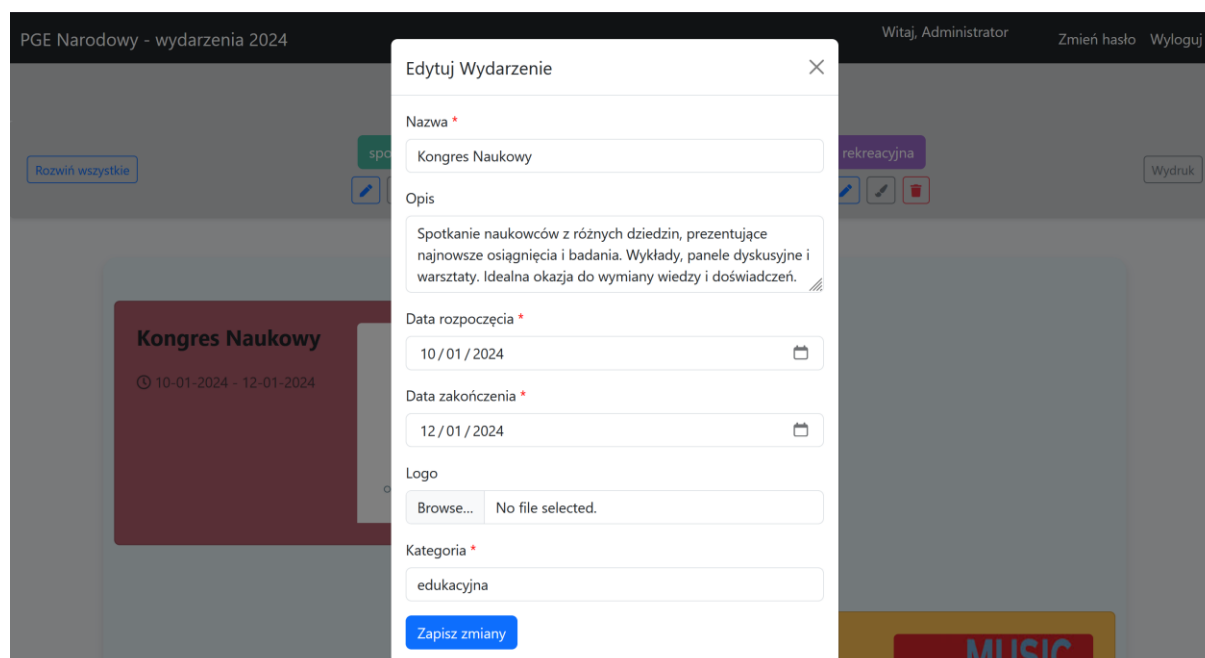
Strona wydruku



Rozwinięte karty wydarzeń



Przefiltrowane karty wydarzeń po kategorii muzycznej



Edycja wydarzenia

PGE Narodowy - wydarzenia 2024

Witaj, Administrator   Zmień hasło   Wyloguj

Rozwiń wszystkie

**Dodaj nowe wydarzenie** ✕

Nazwa \*

Opis

Data rozpoczęcia \*

Data zakończenia \*

Logo  No file selected.

Kategoria \*

**Dodaj**

**Kongres Naukowy**  
🕒 10-01-2024 - 12-01-2024

Wydruk

MUSIC

Dodawanie nowego wydarzenia

PGE Narodowy - wydarzenia 2024

Witaj, Administrator   Zmień hasło   Wyloguj

Rozwiń wszystkie

**Dodaj nową kategorię** ✕

Nazwa kategorii \*

Kolor kategorii \*

**Zapisz**

**Kongres Naukowy**  
🕒 10-01-2024 - 12-01-2024

SCIENCE

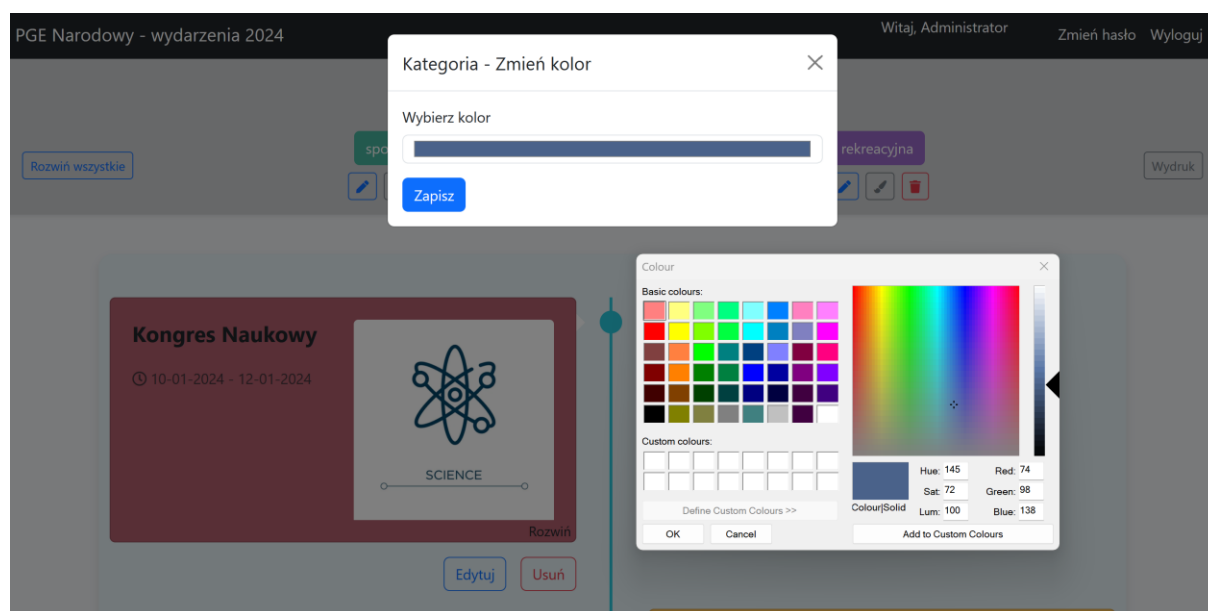
Rozwiń

Edytuj   Usuń

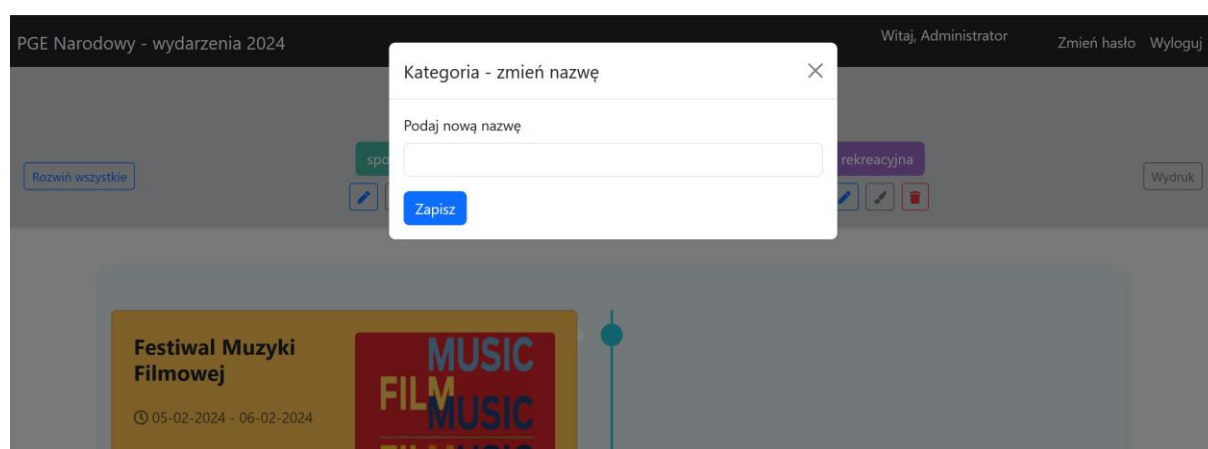
Festiwal Muzyki

MUSIC

Dodawanie nowej kategorii



Zmiana koloru kategorii



Zmiana nazwy kategorii