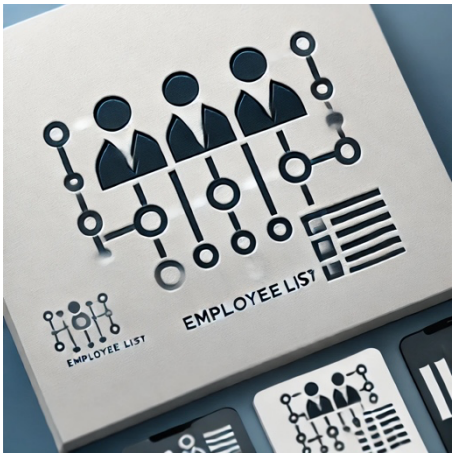
PRAKTYKI	Dokumentacja projektu
Autor	Dawid Olko
Kierunek, rok	Informatyka, III rok, st. stacjonarne (3,5-l)
Temat projektu	*Aplikacja backendowa wykorzystująca bazę MySQL*



EmployeeListApp

19.01.2025r. - 25.02.2025r.

Spis treści

1. Narzędzia i technologie

W projekcie „Lista pracowników” wykorzystano szereg narzędzi i technologii, które wspólnie zapewniają wydajność, skalowalność oraz wygodę w rozwoju aplikacji. Poniżej przedstawiono dokładniejszy opis poszczególnych elementów:

1.1. Język programowania: PHP

- **Wersja**: Projekt opiera się na PHP w wersji co najmniej 8.2.
- **Zalety**:
    - Bogaty ekosystem bibliotek i frameworków.
    - Duże wsparcie społeczności – nieustanny rozwój oraz szybkie reagowanie na zgłoszenia błędów.
    - Łatwa integracja z serwerami WWW (Apache, Nginx) i wieloma systemami zarządzania bazami danych.

1.2. Framework: Laravel

- **Wersja**: Co najmniej 11.x (na moment powstawania projektu).
- **Charakterystyka**:
    1. **Struktura MVC**: Laravel wspiera wzorzec Model-View-Controller, co ułatwia rozdzielenie warstw aplikacji:
        - **Model (Eloquent ORM)**: Obsługa bazy danych i logika biznesowa.
        - **View (Blade lub inny silnik szablonów)**: Prezentacja danych użytkownikowi.
        - **Controller**: Warstwa pośrednia, realizująca komunikację między Modelem a View.
    2. **Eloquent ORM**:
        - Zapewnia prosty i czytelny interfejs do tworzenia zapytań do bazy danych.
        - Pozwala na mapowanie tabel na klasy w projekcie, co znacznie przyspiesza prace programistyczne oraz redukuje ilość tzw. „SQL injection” i błędów związanych z zapytaniami.
    3. **Migrations & Seeders**:
        - Umożliwiają kontrolę wersji schematu bazy danych oraz łatwe dodawanie danych początkowych.
    4. **Routing**:
        - Łatwa konfiguracja tras (URLe) i wiązanie ich z kontrolerami lub akcjami w plikach routes/web.php (dla warstwy webowej) oraz routes/api.php (dla REST API).

Dlaczego Laravel?
Framework ten ceniony jest za wyraźną strukturę, dużą społeczność, a także obszerną dokumentację. Rozwiązania takie jak wbudowany kontener IoC (Inversion of Control), Eventy czy Task Scheduling ułatwiają pracę przy większych projektach, zapewniając również skalowalność.

1.3. Baza danych: MySQL

- **Wykorzystana baza**: MySQL Employees Sample Database.
- **Powód wyboru**:
    1. **Popularność MySQL** – Jest jednym z najczęściej używanych systemów zarządzania relacyjnymi bazami danych.
    2. **Przykładowa baza „Employees”** – Idealnie nadaje się do celów szkoleniowych i prezentacyjnych; zawiera realistyczne dane dotyczące pracowników, pensji, departamentów, co pozwala pokazać w praktyce obsługę bardziej złożonych relacji i zapytań w Laravel.
- **Obsługa w Laravel**:
    - Połączenia konfigurowane w pliku .env oraz w pliku config/database.php.
    - Eloquent ORM zapewnia intuicyjną warstwę abstrakcji, np. nie musimy ręcznie pisać zapytań SQL, lecz korzystamy z metod w stylu Employee::where(...).

1.4. Composer – menedżer pakietów dla PHP

- **Zadania**:
    1. **Instalacja pakietów**: Automatyczne pobieranie bibliotek (np. barryvdh/laravel-dompdf, laravel/sanctum) i ich zależności.
    2. **Aktualizacje**: Ułatwiona obsługa aktualizacji pakietów – wystarczy polecenie composer update.
    3. **Automatyczne ładowanie**: Generowanie autoloadera PSR-4, dzięki czemu nie musimy ręcznie dołączać plików klas.

W pliku composer.json zdefiniowane są wszystkie konieczne paczki wraz z ich wersjami. Po instalacji i aktualizacji pakietów plik composer.lock zabezpiecza spójną wersję oprogramowania w zespole.

1.5. Barryvdh/laravel-dompdf

- **Cel**: Generowanie plików PDF w oparciu o dane z aplikacji.
    - Pozwala łatwo wyeksportować listy pracowników, raporty czy zestawienia finansowe do formatu PDF.
    - Integruje się z silnikiem szablonów Blade, dzięki czemu tworzenie estetycznych raportów jest proste.

2. Baza danych

Aplikacja „Lista pracowników” korzysta z przykładowej bazy danych **Employees**, udostępnionej przez zespół MySQL w repozytorium datacharmer/test_db. Dzięki temu możliwe jest szybkie uruchomienie projektu z gotowymi, przykładowymi danymi pracowników i departamentów. Poniżej zaprezentowano najważniejsze informacje związane z bazą danych w kontekście niniejszego projektu.

2.1. Struktura bazy danych

Baza „Employees” zawiera kilka tabel reprezentujących różne aspekty związane z danymi o pracownikach oraz organizacją firmy:

1. **employees**
    - Gromadzi podstawowe informacje o pracownikach, takie jak:
        - emp_no: unikalny identyfikator pracownika (klucz główny).
        - birth_date: data urodzenia.
        - first_name: imię.
        - last_name: nazwisko.
        - gender: płeć (M/F).
        - hire_date: data zatrudnienia.
    - Tabela ta ma kluczowe znaczenie, ponieważ dane w pozostałych tabelach często odwołują się właśnie do emp_no.
2. **departments**
    - Zawiera informacje o działach:
        - dept_no: unikalny identyfikator działu (klucz główny).
        - dept_name: nazwa działu (np. „Sales”, „Research” itp.).
    - W relacji do pracowników jest to tabela słownikowa – w innych tabelach (np. dept_emp) znajdziemy odniesienia do dept_no.
3. **dept_emp**
    - Powiązanie pomiędzy pracownikiem (emp_no) a działem (dept_no).
    - Zawiera dodatkowe informacje o okresie zatrudnienia w danym dziale:
        - from_date: data rozpoczęcia pracy w dziale.
        - to_date: data zakończenia pracy w dziale (może być bieżąca data, jeśli pracownik nadal tam pracuje).
    - Dzięki temu aplikacja może określać bieżący dział pracownika na podstawie aktualnej daty.

4. **dept_manager**
    - Analogicznie do dept_emp, ale przechowuje informacje o menedżerach działów.
    - Każdy dział może mieć przypisanego jednego bądź więcej menedżerów w różnych okresach (np. gdy następują zmiany na tym stanowisku).
5. **salaries**
    - Lista wynagrodzeń wszystkich pracowników w czasie:
        - emp_no: pracownik, do którego należy wynagrodzenie.
        - salary: wysokość pensji.
        - from_date, to_date: przedział czasowy obowiązywania danej pensji.
    - Dzięki tej tabeli można m.in. obliczyć łączną sumę wypłat dla konkretnego pracownika, a także zweryfikować aktualne wynagrodzenie (ostatni wpis obowiązujący do „9999-01-01” wskazuje aktualną pensję).
6. **titles**
    - Przechowuje nazwy stanowisk (np. „Engineer”, „Senior Staff”, „Manager”) wraz z przedziałami czasowymi, w których pracownik pełnił daną funkcję:
        - emp_no: identyfikator pracownika.
        - title: nazwa stanowiska.
        - from_date, to_date: czas pełnienia danego stanowiska.

Tabele dept_emp, dept_manager, salaries oraz titles zawierają przeważnie przedziały czasowe, co pozwala śledzić historię zatrudnienia i stanowisk pracownika. W aplikacji potrzebujemy w szczególności wiedzieć, jaki jest **obecny** dział i **aktualna** pensja pracownika – do tego służą zapytania filtrujące rekordy aktywne (np. to_date = '9999-01-01' albo zbliżona wartość graniczna).

2.2. Import bazy danych

1. **Pobranie repozytorium**
   Z oficjalnego repozytorium datacharmer/test_db ściągamy pliki SQL i skrypty shell, które pozwalają na szybkie zaimportowanie bazy.
2. **Tworzenie bazy w MySQL**
   Na lokalnym serwerze MySQL tworzymy nową bazę, np. o nazwie employees:

   ```
   mysql -u root -p -e "CREATE DATABASE employees"
   ```
3. **Import pliku employees.sql**
   W głównym katalogu repozytorium *test_db* znajduje się plik employees.sql zawierający definicje tabel i przykładowe dane. Importujemy go:

   ```
   mysql -u root -p employees < employees.sql
   ```

Po tej operacji nasza baza danych będzie wypełniona tabelami i przykładowymi rekordami.

4. **Konfiguracja w pliku .env**
   W projekcie Laravel należy ustawić parametry połączenia z bazą w pliku .env, np.:

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=employees
DB_USERNAME=root
DB_PASSWORD=
```

2.3. Zastosowanie w aplikacji

- **Eloquent Models**:
  W celu wygodnej obsługi bazy w Laravel można utworzyć modele odpowiadające tabelom, np. Employee, Department, Salary, Title. Każdy model odwzorowuje jedną tabelę:

```
class Employee extends Model
{
    protected $table = 'employees';
    protected $primaryKey = 'emp_no';
    // ...
}
```

Pozwala to na pisanie zapytań w stylu:

```
$employee = Employee::with('salaries', 'titles')
                    ->where('emp_no', $id)
                    ->first();
```

Zamiast manualnego pisania kwerend SQL.

- **Relacje**:
  Aplikacja może definiować relacje między modelami np.:
  - Employee ma wiele Salaries (one-to-many).
  - Employee ma wiele Titles (one-to-many).
  - Employee ma wiele Departments przez dept_emp (many-to-many). Dzięki temu pobieranie danych z wielu tabel jednocześnie jest znacznie prostsze.
- **Filtrowanie**:
  Przy wyborze wyłącznie aktualnych pracowników lub aktualnej pensji w zapytaniach Eloquent można zawrzeć warunek to_date = '9999-01-01' lub podobny. W ten sposób odnajdujemy bieżące stanowiska i aktywne pensje.
- **Eksport danych**:
  Korzystając z tabel salaries, można obliczyć całkowitą sumę wynagrodzeń.

2.4. Zalety przykładowej bazy „Employees”

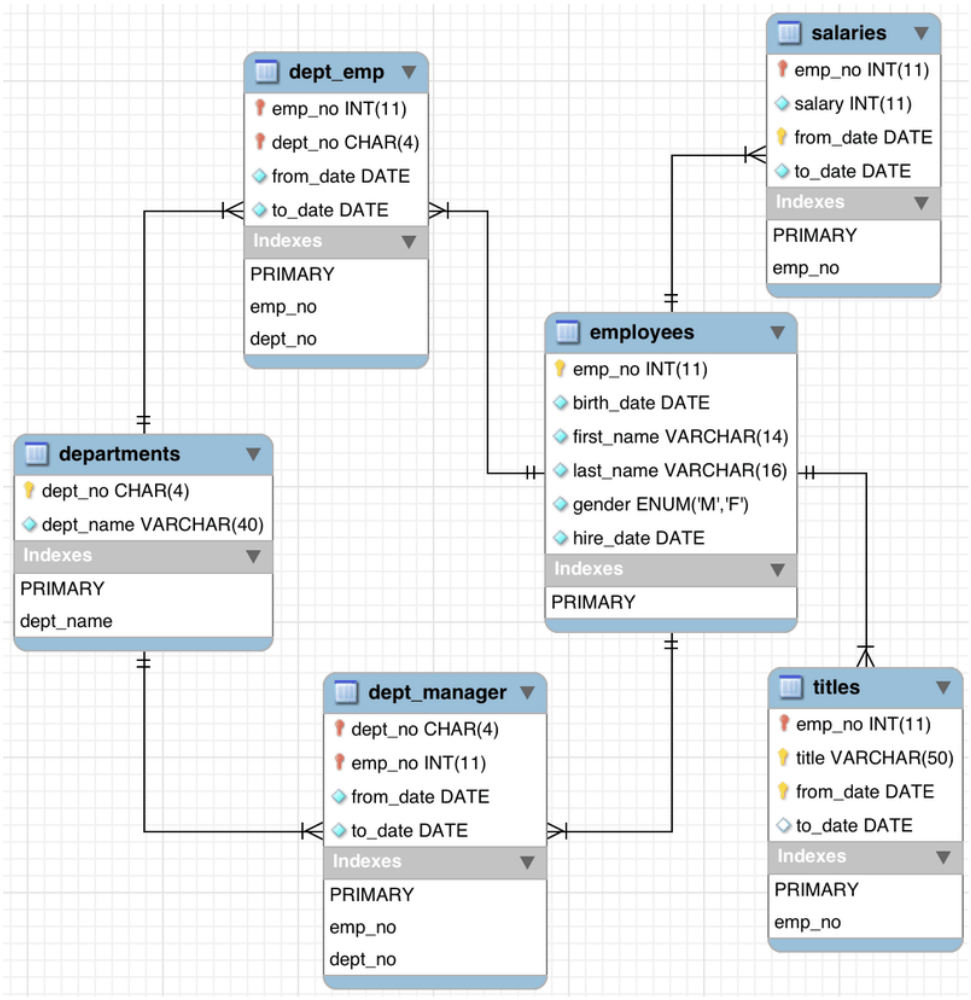
1. **Realistyczne dane** – zawiera setki tysięcy rekordów (w zależności od importu), co pozwala testować wydajność aplikacji i zapytań.
2. **Różnorodność struktur** – posiadamy tabele z relacjami typu one-to-many, many-to-many, a także dane historyczne (z datami from_date i to_date).
3. **Bieżące i historyczne rekordy** – ułatwia naukę operowania na aktualnych i przeszłych stanach zatrudnienia, co jest typowe w rzeczywistych systemach kadrowych.

2.5. Podsumowanie rozdziału

Baza danych *Employees* stanowi solidną podstawę do tworzenia i testowania aplikacji związanych z zarządzaniem pracownikami. Jej struktura pozwala na:

- Skuteczną demonstrację relacji i zapytań SQL w kontekście rzeczywistych danych.
- Ćwiczenie mechanizmów Eloquent w Laravel, takich jak relacje czy filtry.
- Prezentację funkcjonalności raportowania (np. generowania listy pracowników z bieżącymi tytułami i pensjami, a także eksportu PDF/CSV).

Figure 1 The Employees Schema



3. GUI

3.1. Wprowadzenie

W ramach oceny interfejsu użytkownika (GUI) projektu przeprowadzono testy przy pomocy narzędzia **Lighthouse**, dostępnego w **Google Chrome DevTools**. Lighthouse pozwala na zautomatyzowaną analizę strony internetowej i wygenerowanie raportu obejmującego:

- **Performance** (wydajność),
- **Accessibility** (dostępność),
- **Best Practices** (najlepsze praktyki),
- **SEO** (pozycjonowanie).

Testy wykonano zarówno w trybie **Mobile** (symulacja urządzeń mobilnych), jak i **Desktop** (komputery stacjonarne). Poniżej zaprezentowano opis poszczególnych testów oraz wnioski z nich płynące.

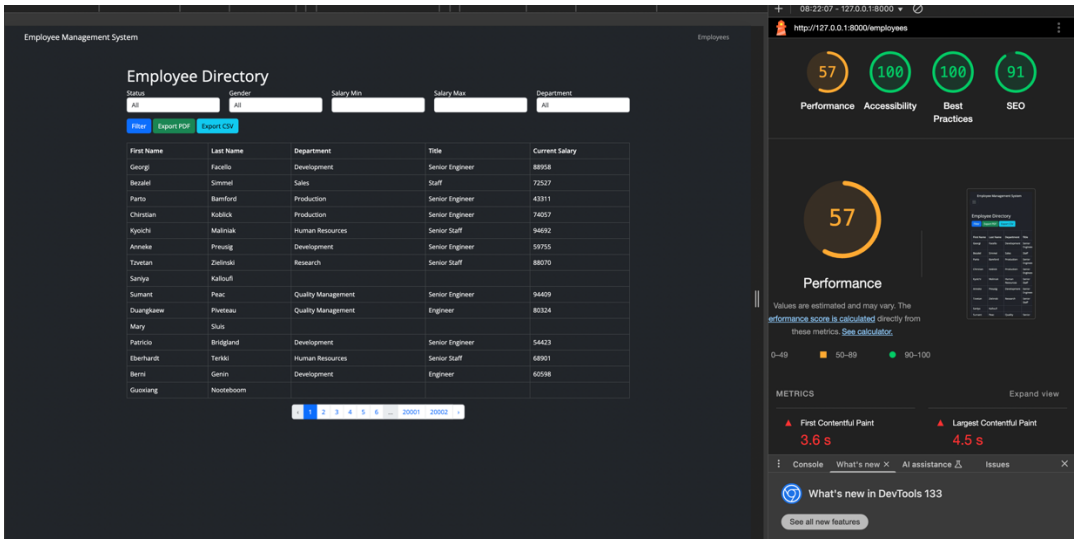
3.2. Lighthouse – Wersja Mobilna

Przeprowadzając test w trybie mobilnym, Lighthouse wykorzystuje parametry charakterystyczne dla urządzeń z mniejszymi ekranami (np. smartfonów). Symuluje także warunki sieciowe typowe dla 3G/4G, co pozwala ocenić, czy strona jest **szybka i wygodna** w użyciu w środowisku mobilnym.

Główne kryteria oceny (Mobile):

1. **Performance (Wydajność)**
    - Czas ładowania zasobów (HTML, CSS, JavaScript, obrazów).
    - Responsywność aplikacji w momencie interakcji użytkownika.
    - Wskaźniki takie jak *Time to Interactive*, *Largest Contentful Paint* czy *Cumulative Layout Shift*.
2. **Accessibility (Dostępność)**
    - Czytelność i kontrast treści na ekranach o małej rozdzielczości.
    - Prawidłowe etykiety ARIA, opisy alternatywne obrazów i elementów interfejsu.
    - Przyjazność dla użytkowników czytników ekranu.
3. **Best Practices (Najlepsze praktyki)**
    - Zgodność ze standardami sieciowymi (np. brak błędów w konsoli, poprawne użycie HTTPS, optymalizacja zasobów).
    - Użycie nowoczesnych technologii webowych z zachowaniem kompatybilności wstecznej.
4. **SEO**
    - Czy strona posiada metatagi i elementy ułatwiające indeksowanie (tytuł, opis, struktura nagłówków)?
    - Poprawna implementacja responsywności (viewport meta tag), co wpływa na ranking w wyszukiwarkach mobilnych.

Wynik testu mobilnego:



Uwagi optymalizacyjne:

- Skonfigurowanie obrazów w formatach nowej generacji (np. WebP) może obniżyć wielkość plików.
- Redukcja liczby zewnętrznych skryptów (lub ich asynchroniczne ładowanie) usprawnia pierwsze wyświetlenie treści.
- Zadbanie o odpowiedni kontrast tekstu na tle (zwłaszcza w jasnym świetle) ułatwia korzystanie osobom z wadami wzroku.

3.3. Lighthouse – Wersja Desktop

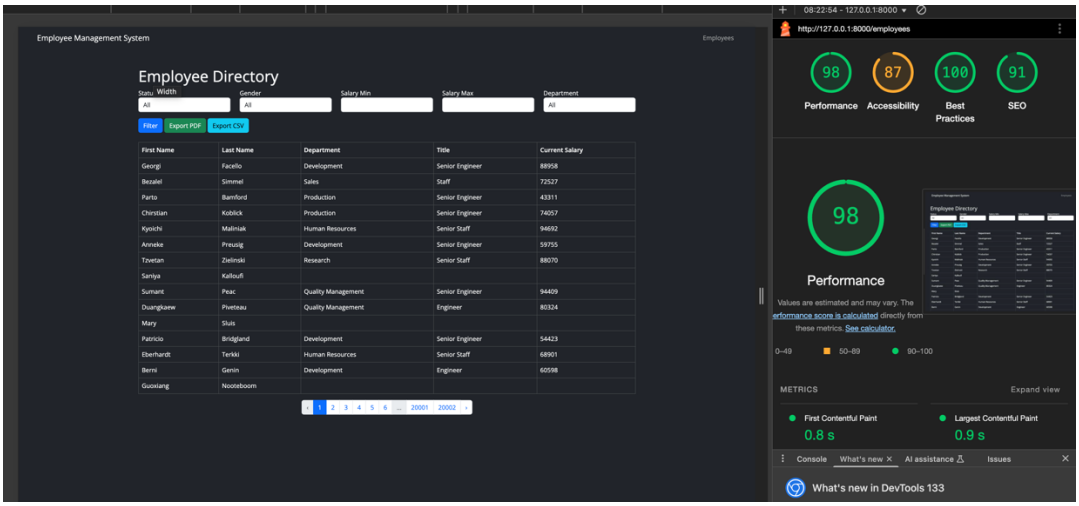
Kolejny test Lighthouse skonfigurowano dla środowiska desktopowego, co pozwala na weryfikację działania aplikacji przy większych rozdzielczościach ekranu i często szybszym łączu internetowym.

Główne kryteria oceny (Desktop):

1. **Performance (Wydajność)**
    - Pomiar szybkości ładowania strony (czas do pojawienia się treści na ekranie).
    - Ocena płynności przewijania i renderowania elementów interfejsu.
2. **Accessibility (Dostępność)**
    - Czytelne rozdzielenie sekcji, logiczna hierarchia nagłówków (H1, H2, H3...).
    - Dostosowanie rozmiaru i stylu czcionek do większych ekranów.
    - Odpowiedni kontrast dla wygody pracy w warunkach biurowych.
3. **Best Practices (Najlepsze praktyki)**
    - Sprawdzanie poprawnego działania skryptów w przeglądarkach desktopowych.
    - Bezpieczeństwo (HTTPS, brak luk w JavaScript).
    - Poprawne implementacje atrybutów w znaczeniach HTML5.

4. **SEO**
    - Kompletność elementów meta, przyjazne adresy URL, prawidłowa struktura HTML.
    - Szybkość reakcji i obciążenie serwera, co wpływa na ranking stron w wyszukiwarkach.

Wynik testu desktopowego:



Uwagi optymalizacyjne:

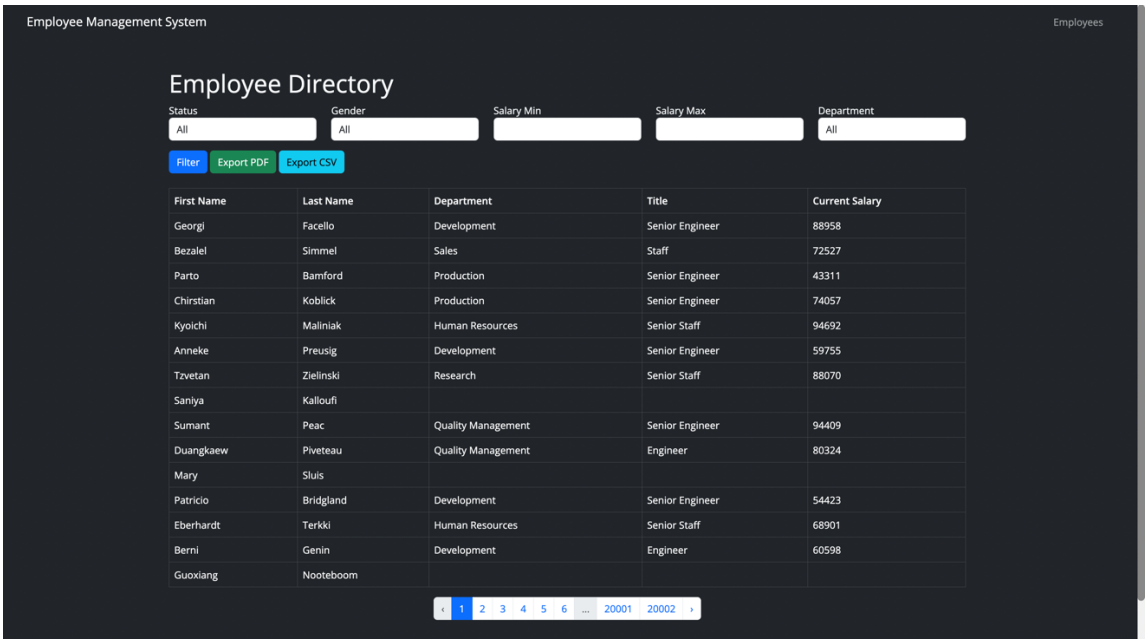
- Dla dużych ekranów warto rozważyć „leniwe ładowanie” (lazy loading) elementów, które nie pojawiają się od razu w widoku użytkownika.
- Minimalizacja i kompresja plików CSS/JS przed ich publikacją zwiększają wydajność.
- Kluczowe treści (critical CSS) można zaimplementować inline, aby jeszcze bardziej przyspieszyć wyświetlanie interfejsu.

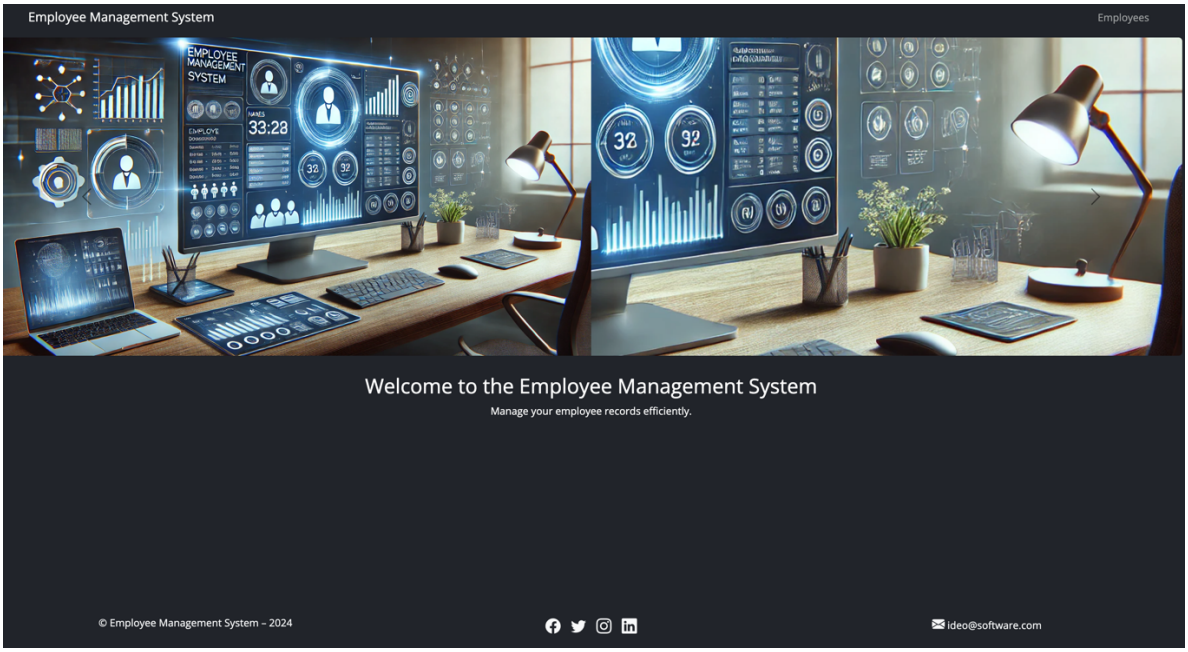
3.4. Podsumowanie

Raporty **Lighthouse** zarówno w wersji mobilnej, jak i desktopowej wskazują, że aplikacja stoi na wysokim poziomie w kontekście wydajności, dostępności, najlepszych praktyk i SEO. Pomimo ogólnie dobrych wyników, istnieje kilka obszarów, które można usprawnić – głównie pod kątem **optymalizacji grafik** oraz **minimalizacji zasobów** w celu poprawy czasu wczytywania.

Kluczowe wnioski:

- **Wersja mobilna** wymaga szczególnej uwagi w zakresie responsywności i szybkości ładowania przy wolniejszych połączeniach internetowych.
- **Wersja desktopowa** może zostać zoptymalizowana, np. przez wprowadzenie lazy loadingu oraz dalsze zmniejszenie rozmiaru plików JS/CSS.
- Dobra dostępność (Accessibility) i wysokie noty SEO świadczą o tym, że aplikacja jest intuicyjna dla szerokiego grona użytkowników i przyjazna dla wyszukiwarek internetowych.


Employee Management System
Employees
Employee Directory
Status
All
Gender
All
Salary Min
Salary Max
Department
All
Filter
Export PDF
Export CSV
First Name
Last Name
Department
Title
Current Salary
Georgi
Facello
Development
Senior Engineer
88958
Bezalel
Simmel
Sales
Staff
72527
Parto
Bamford
Production
Senior Engineer
43311
Chirstian
Koblick
Production
Senior Engineer
74057
Kyoichi
Maliniak
Human Resources
Senior Staff
94692
Anneke
Preusig
Development
Senior Engineer
59755
Tzvetan
Zielinski
Research
Senior Staff
88070
Saniya
Kalloufi
Sumant
Peac
Quality Management
Senior Engineer
94409
Duangkaew
Piveteau
Quality Management
Engineer
80324
Mary
Sluis
Patricio
Bridgland
Development
Senior Engineer
54423
Eberhardt
Terkki
Human Resources
Senior Staff
68901
Berni
Genin
Development
Engineer
60598
Guoxiang
Nooteboom
‹ 1 2 3 4 5 6 ... 20001 20002 ›


Employee Management System
Employees
EMPLOYEE MANAGEMENT SYSTEM
Welcome to the Employee Management System
Manage your employee records efficiently.
© Employee Management System – 2024
ideo@software.com

4. Uruchomienie aplikacji

Poniższy rozdział wyjaśnia, w jaki sposób uruchomić projekt **Lista pracowników** zbudowany w oparciu o **Laravel** i korzystający z przykładowej bazy danych *Employees* (MySQL).

4.1. Wymagania systemowe

1. **PHP**
    - Wersja **8.2** lub wyższa (z zainstalowanym rozszerzeniem pdo_mysql).
2. **MySQL / MariaDB**
    - Baza danych zgodna z MySQL (np. MySQL 5.7, 8.x lub MariaDB).
    - Zaimportowana przykładowa baza *Employees* (z repozytorium datacharmer/test_db).
3. **Composer**
    - Menedżer pakietów dla PHP – wymagana instalacja przed pobraniem zależności projektu.
4. **Przeglądarka**
    - Dowolna nowoczesna przeglądarka (np. Chrome, Firefox, Edge) do testowania aplikacji w trybie lokalnym.

4.2. Konfiguracja i uruchomienie projektu

Krok 1: Pobranie projektu

Jeżeli kod przechowywany jest w repozytorium GIT, należy go sklonować do lokalnego folderu:

```
git clone https://gitlab.ideo.pl/m.koszyk/pracownicy
cd pracownicy
```

Krok 2: Instalacja zależności i ustawienie środowiska

1. **Instalacja zależności Composer**
   W głównym katalogu projektu (gdzie znajduje się plik composer.json) wykonaj:

```
composer install
```

To polecenie pobierze wszystkie paczki wymagane do uruchomienia aplikacji Laravel (m.in. laravel/framework, barryvdh/laravel-dompdf, laravel/sanctum).

2. **Kopia pliku .env**
   Skopiuj plik .env.example do .env:

```
cp .env.example .env
```

Następnie w edytorze tekstu uzupełnij ustawienia bazy danych zgodnie z lokalną konfiguracją MySQL i nazwą bazy (np. employees), np.:

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=employees
DB_USERNAME=root
DB_PASSWORD=
```

3. **Generowanie klucza aplikacji**
   W celu zabezpieczenia sesji i danych należy wygenerować unikatowy klucz:

   ```
   php artisan key:generate
   ```

4. **Tworzenie łącza do katalogu storage** (opcjonalne)
   Jeżeli aplikacja zapisuje pliki (np. eksporty PDF), możesz potrzebować:

   ```
   php artisan storage:link
   ```

Umożliwi to łatwy dostęp do plików umieszczonych w storage/app/public.

Krok 3: (Opcjonalnie) Migracje i seedy

Jeśli w projekcie znajdują się dodatkowe tabele lub seedy (oprócz bazy Employees), można je uruchomić komendą:

```
php artisan migrate
php artisan db:seed
```

Jeśli baza *Employees* jest już zaimportowana z zewnętrznego pliku SQL, te kroki dotyczą tylko ewentualnych autorskich tabel i danych uzupełniających projekt.

Krok 4: Uruchomienie serwera deweloperskiego

Aby uruchomić aplikację w trybie lokalnym (z pomocą wbudowanego serwera Laravel), wystarczy:

```
php artisan serve
```

Domyślnie aplikacja będzie dostępna pod adresem: `http://127.0.0.1:8000`.

4.3. Podsumowanie rozdziału

Proces uruchamiania aplikacji **Lista pracowników** obejmuje kilka kroków: od pobrania projektu, przez instalację zależności (Composer, opcjonalnie npm), konfigurację pliku .env, aż po uruchomienie wbudowanego serwera Laravel. Po tych czynnościach aplikacja powinna bezproblemowo nawiązać połączenie z przykładową bazą *Employees*, a użytkownik zyska dostęp do funkcjonalności przeglądania i filtrowania listy pracowników, jak również generowania raportów (eksport do PDF) i innych zaplanowanych w projekcie możliwości.

5. Podsumowanie

Projekt **Lista Pracowników** stanowi przykład praktycznego wykorzystania frameworka **Laravel** oraz przykładowej bazy danych *Employees* (MySQL). W trakcie realizacji przedsięwzięcia dokonano następujących kluczowych kroków:

1. **Zastosowanie sprawdzonej architektury**:
   Dzięki Laravelowi możliwe było wprowadzenie jasnej struktury projektowej (Model-View-Controller) i wykorzystanie mechanizmu **Eloquent ORM** do zwięzłego zarządzania danymi o pracownikach, pensjach czy działach.

2. **Bieżące oraz historyczne dane**:
   Wykorzystana baza *Employees* pozwoliła na ćwiczenie pracy z danymi o charakterze historycznym (np. okresy obowiązywania konkretnych pensji i stanowisk). W aplikacji można było zaprezentować wyłącznie aktualne informacje lub też uwzględnić pełną historię zatrudnienia.

3. **Filtrowanie i generowanie raportów**:
   Kluczowe funkcjonalności – takie jak wyszukiwanie pracowników według działu, zakresu pensji czy płci – demonstrują mechanizmy zapytań w Eloquent. Natomiast eksport do pliku (np. PDF) pokazuje praktyczne zastosowanie dodatkowych bibliotek (np. **barryvdh/laravel-dompdf**).

4. **Dobre praktyki i narzędzia**:
   - Wykorzystanie **Composer** do zarządzania zależnościami oraz
   - Możliwość uruchamiania migracji i seedów, co ułatwia przenoszenie i odtwarzanie środowisk deweloperskich.
   - Potencjalna integracja z narzędziami front-endowymi (np. Vite, npm) w celu optymalizacji zasobów.

5. **Możliwości rozbudowy**:
   Projekt można rozszerzyć o moduł zarządzania uprawnieniami, zaawansowaną analitykę danych (np. raporty roczne), integrację z API czy systemem autoryzacji tokenowej **(Sanctum)**. Dzięki elastyczności Laravela wdrażanie nowych funkcji jest stosunkowo łatwe.

Realizacja niniejszego projektu potwierdza, że **Laravel** w połączeniu z przykładową bazą *Employees* stanowi doskonałe środowisko szkoleniowe. Umożliwia naukę kluczowych elementów programowania obiektowego w PHP, obsługi relacyjnej bazy danych oraz budowy funkcjonalnego interfejsu użytkownika. Aplikacja nie tylko prezentuje dane pracowników w czytelnej formie, ale również zapewnia opcje filtracji i eksportu, co odpowiada **rzeczywistym wyzwaniom** stawianym przed systemami kadrowymi i raportowymi.