|  |  |
| --- | --- |
| **AI1** | Dokumentacja projektu |
| **Autor** | Bartłomiej Florek, 125115 |
| **Kierunek, rok** | Informatyka, II rok, st. stacjonarne (3,5-l) |
| **Temat**   **projektu** | *Sklep ze słodyczami na wagę* |

Spis treści

[Wstęp 3](#_Toc167698779)

[Narzędzia i technologie 4](#_Toc167698780)

[Baza danych 5](#_Toc167698781)

[GUI 6](#_Toc167698782)

[Uruchomienie aplikacji 10](#_Toc167698783)

[Funkcjonalności aplikacji 12](#_Toc167698784)

# Wstęp

Tematem projektu jest utworzenie strony w tematyce zamawiania słodyczy na wagę. Niezalogowany użytkownik może założyć nowe konto oraz przeglądać udostępnione zasoby przez aplikacje. Zalogowany użytkownik, ma przedstawione dostępne produkty oferowane przez sklep, które może zamówić za odpowiednia opłatą. Użytkownik ma dostęp do wielu funkcjonalności oferowanych przez aplikacje takich jak galeria zdjęć, formularz kontaktowy, cennik, odnośniki do social mediów, sprawdzania swoich poprzednich zamówień oraz wiele innych. Admin posiada możliwość dodawanie nowych produktów, ich edycji jak i ich usuwania.

Projekt znajduje się pod linkiem: <https://github.com/flor3kk/ApkiInternetowe/tree/main/projekt>

# Narzędzia i technologie

Wersja PHP (w CMD wpisać php -v): PHP 8.2.12 (cli) (built: Oct 24 2023 21:15:15) (ZTS Visual C++ 2019 x64)

Wersja Laravela (w terminalu w command prompt wpisać polecenie php artisan --version): Laravel Framework 11.4.0

Wersja XAMPPA: 3.3.0

Wersja Visual Studio Code: 1.89.1 (user setup)

Dokumentacje oraz technologie:

Laravel: <https://laravel.com/docs/11.x>

PHP: <https://www.php.net/manual/en/>

Baza danych: <https://docs.phpmyadmin.net/en/latest/>

Bootstrap: <https://getbootstrap.com/docs/5.3/getting-started/introduction/>

JavaScript: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>

HTML5: <https://developer.mozilla.org/en-US/docs/Glossary/HTML5>

Narzędzia:

Visual Studio Code: <https://code.visualstudio.com/>

XAMPP: <https://www.apachefriends.org/pl/index.html>

Composer Setup: <https://getcomposer.org/Composer-Setup.exe>

# Baza danych

Diagram ERD wygenerowany przez phpMyAdmin

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

Rysunek 1 - diagram

Baza danych zawiera takie tabele jak products, która zawiera informacje na temat produktów możliwych do zamówienia, categories zawierających informacje o kategoriach danego produktu,

Orders, która trzyma informacje o zamówieniach poszczególnych użytkowników oraz tabele users zawierającą dane użytkowników. Tabele sessions, password\_reset\_tokens, migrations oraz users są automatycznie tworzone przez framework laravela.

Tabela users do tabeli orders, relacja jeden do wielu,

Tabela orders do tabeli products, relacja jeden do jeden

Tabela categories do tabeli products, relacja jeden do wielu

# GUI

Strona główna aplikacji:

Obraz zawierający tekst, Strona internetowa, Reklama internetowa, zrzut ekranu

Opis wygenerowany automatycznie

Rysunek 2 - główna strona

Strona przedstawiająca ofertę sklepu:

Obraz zawierający tekst, zrzut ekranu, jedzenie, Strona internetowa

Opis wygenerowany automatycznie

Rysunek 3 - asortyment

Formularz kontaktowy:

Obraz zawierający tekst, zrzut ekranu, mapa, diagram

Opis wygenerowany automatycznie

Rysunek 4 - formularz kontaktowy

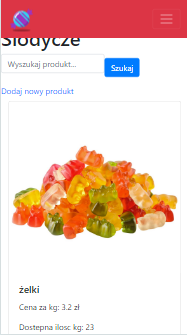
Strona przedstawiająca koszyk i przykładowo dodane do niego produkty:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Ikona komputerowa

Opis wygenerowany automatycznie

Rysunek 5 - koszyk użytkownika

Na mniejszych ekranach strona przedstawiająca artykuły:



Rysunek 6 - na telefonie IPhone

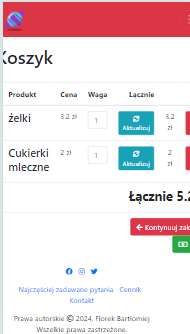
Strona główna:

Obraz zawierający jedzenie, słodycz, słodycze, Przekąska

Opis wygenerowany automatycznie

Rysunek 7 - na telefonie

Koszyk:



Rysunek 8 - koszyk na telefonie

Menu:

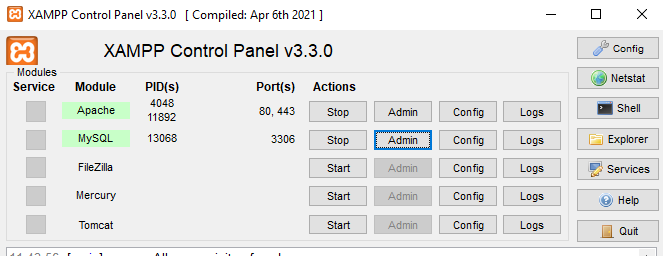
Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Strona internetowa

Opis wygenerowany automatycznie

Rysunek 9 - menu na telefonie

# Uruchomienie aplikacji

Aby mieć możliwość uruchomienia aplikacji należy mieć pobrane takie aplikacje jak XAMPP, Composer Setup oraz Visual Studio Code. Należy podążać z etapami instalacji. Uruchomić XAMPP Control Panel i wystartować serwer Apache oraz MySQL. Przejść przez przycisk admin przy MySQL do panelu admina. Utworzyć nową bazę danych o nazwie projekt.



Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Następie uruchomić aplikacje Visual Studio Code, otworzyć folder zawierający pliki projektowe. Przejść do zakładki terminal i otworzyć nową kartę terminala.

Obraz zawierający tekst, oprogramowanie, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Zmienić terminal z powershella na command prompt.

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

Wpisać komendę ‘composer install’, to polecenie utworzy folder vendor, teraz trzeba skopiowac zawartość pliku .env.example do pliku .env i uzupełnić go tak jak poniżej

Obraz zawierający tekst, Czcionka, design

Opis wygenerowany automatycznie

Rysunek 10 - fragment pliku .env

Po skonfigurowaniu pliku .env należy wpisać w konsoli polecenie ‘php artisan key:generate’, które wygeneruje klucz do naszego pliku. Bez tego klucza aplikacja nie będzie działała poprawnie. Po wykonaniu tych czynności należy wpisać komendę ‘php artisan serve’, uruchomi to serwer. Otworzyć drugą kartę terminala i wpisać polecenie ‘php artisan migrate’, to polecenie utworzy nowe tabele w naszej bazie danych. Aby uzupełnić te tabele przykładowymi danymi musimy wpisać kolejną komendę ‘php artisan db:seed’. Teraz po wykonaniu tych poleceń możemy uruchomić przeglądarkę i przejść pod adres <http://127.0.0.1:8000/>.

# Funkcjonalności aplikacji

Gość:

* Utworzenie konta
* Logowanie
* Przeglądanie ogólnodostępnych zasobów
* Wyszukiwanie produktów

Użytkownik:

* Zamawianie produktów
* Dodawanie do koszyka
* Usuwanie swoich zamówień

Administrator:

* Edycja produktów
* Dodawanie nowych produktów
* Usuwanie bieżących produktów

Etap logowania na użytkownika o loginie [jan@email.com](mailto:jan@email.com) o raz haśle 1234, lub na konto admina o loginie [admin@admin.com](mailto:admin@admin.com) o tym samym haśle

Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie

Tworzenie konta oraz walidacja danych:

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

Funkcja zajmująca się rejestracja nowego użytkownika

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Rysunek 11 - fragment kodu do rejestracji

Metoda registration(), jest to metoda GET, która odpowiada za wyświetlanie formularza rejestracji użytkownika, zwracając widok auth.register

Metoda registrationPost(), jest to metoda POST, która obsługuje dane przesłane z formularza rejestracji. Przyjmuje obiekt Request jako argument, który zawiera dane z formularza. Metoda ta waliduje dane wejściowe, sprawdzając czy wszystkie pola są wymagane, czy pole email jest poprawnym adresem email, niepowtarzającym się w tabeli users, czy hasło zawiera co najmniej 8 znaków, i czy pole repass jest identyczne.

Jeśli wszystkie dane zostały poprawnie wprowadzone, zostają one przygotowywane do zapisania w bazie danych. Pole name oraz email pozostaje bez zmian, natomiast hasło jest hashowane za pomocą funkcji Hash::make().

Tworzony jest nowy rekord w tabeli users, za pomocą metody User::create($data), w którym przekazujemy podane dane.

Jeśli dodanie przejdzie pomyślnie, użytkownik przekierowywany jest do strony logowania, w przeciwnym wypadku wraca on na stronę rejestracji z odpowiednimi informacjami.

CRUD przeprowadzany przez administratora:

Dodawanie nowych produktów

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, numer, Czcionka

Opis wygenerowany automatycznie

Realizacja walidacji znajduje się w pliku StoreProductRequest, w funkcji rules, która zawiera tablice sprawdzającą wpisane pola

Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie

Rysunek 12 - fragment kodu odpowiedzialny za walidacje

Metoda rules(), zwraca tablice reguł walidacji dodawania nowego produktu. Określa ona wymagania dotyczące pól, które muszą zostać spełnione aby dodać nowy produkt do bazy

Reguły walidacji:

- pole ‘name’ – jest wymagane, typu string, unikalna wartość w tabeli products, oraz max 50 znaków

- pole ‘price’ – wymagane, musi być liczbą, wartość nie może przekroczyć 10 ale musi być co najmniej 1

- pole ‘available’ – wymagane, musi być liczba, max 30, co najmniej 1

- pole ‘calories’ – wymagane, typu string, min 10 znaków, max 255

- pole ‘description’ – wymagane, typu string, min 10 znaków, max 255

- pole ‘category\_id’ – wymagane, musi być liczbą, wartość pola musi istnieć w kolumnie id w tabeli ‘categories’

Funkcja w ProductController, która obsługuje dodawanie produktu do bazy

Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie

Rysunek 13 - funkcja zapisująca nowy produkt

Funkcja store() przyjmuje request zawierający dane nowego produktu, waliduje je za pomocą klasy StoreProductRequest, a następnie zapisuje te dane w tabeli products.

Pobiera wszystkie dane o nowym produkcie z requesta i zapisuje je w tablicy $input.

Product::create($input) tworzy nowy rekord w tabeli products z danymi zawartymi w tablicy $input. Teraz następuje przekierowanie do strony głównej produktów.

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

Rysunek 14 - funkcja create

Usuwanie wybranego produktu

Obraz zawierający tekst, zrzut ekranu, jedzenie

Opis wygenerowany automatycznie

Usuwanie produktu znajduje się w funkcji destroy(), przyjmującej jako argument obiekt Product, który jest automatycznie zainicjowany na podstawie ID przekazanego w żądaniu. Następuje sprawdzenie czy aktualnie zalogowany użytkownik jest adminem, a po tym sprawdzenie czy aktualny produkt ma jakieś powiązania z innymi tabelami, np. z zamówieniami, wtedy otrzymamy komunikat, że takie działanie jest niedozwolone.

Jeśli natomiast produkt nie ma żadnych powiązań, tzn. nikt go nie zamówił, zostaje on usunięty, i wracamy do strony głównej z produktami. Jeśli zalogowany użytkownik nie posiada praw do usuwania, aplikacja zabroni mu usunięcia produktu, oraz zwróci odpowiedni błąd i przekieruje go na stronę do logowania.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Rysunek 15 - funkcja usuwająca produkt

Edycja wybranego przez admina produktu, zawiera się w funkcji edit(), przyjmującej ID klikniętego produktu. Metoda próbuje znaleźć produkt w tabeli products oraz pobiera wszystkie rekordy z tabeli categories i przypisuje je do zmiennej $categories.

Tak samo jak w przypadku usuwania, następuje sprawdzanie aktualnie zalogowanego użytkownika i przekierowanie go bądź nie, do widoku edycji produktu wraz ze zmiennymi.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Rysunek 16 - funkcja edytująca produkt

Obraz zawierający tekst, zrzut ekranu, numer, Czcionka

Opis wygenerowany automatycznie

Po kliknięciu przycisku wyślij, uruchamia się funkcja update(), aktualizującej podany produkt. Funkcja przyjmuje request, zawierający dane do edycji oraz produkt do zaktualizowania. Funkcja realizuje walidacje na takiej samej zasadzie jak w przypadku rejestracji użytkownika. Do tablicy $input, pobiera wszystkie dane przesłane w requescie. Następnie aktualizuje podany produkt o nowe dane zawarte w tablicy $input. I wraca do głównego widoku produktów

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

Rysunek 17 - funkcja walidująca edytowane dane

Przeglądanie ogólnodostępnych zasobów

Obraz zawierający tekst, zrzut ekranu, Strona internetowa, oprogramowanie

Opis wygenerowany automatycznie

Funkcja index() służy do wyświetlania listy produktów wraz z ich kategoriami. Do zmiennej $products przypisywane są produkty wraz z ich kategoriami with(‘category’),

Funkcja zwraca widok products.home, przekazując tablice produktów.

Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie

Rysunek 18 - funkcja zwracająca widok produktów

Poprzez wyszukiwarkę możemy wyszukać interesujące nas produkty, wyszukuje to po nazwie oraz po opisie.

Metoda obsługująca wyszukiwarkę:

Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie

Rysunek 19 - funkcja odpowiedzialna za wyszukiwarke

Do zmiennej $query przypisujemy wartość wpisaną przez użytkownika.

Metoda wykonuje zapytanie do bazy, szukając produktów, których nazwa ‘name’ zawiera wpisany przez użytkownika ciąg znaków, albo produktów, których opis ‘description’ zawiera wpisany przez użytkownika ciąg. Get() pobiera wszystkie produkty spełniające którykolwiek warunek.

Następnie funkcja zwraca widok search i przekazuje do niego tablice danych produktów oraz oryginalne zapytanie użytkownika.

Obraz zawierający tekst, jedzenie

Opis wygenerowany automatycznie

Użytkownik może dodawać produkty do koszyka, które potem w karcie koszyka może edytować w sensie zwiększać ilość, usunąć z koszyka itp. Po złożeniu zamówienia użytkownik może przeglądać swoje zamówienia oraz je usuwać.

Metoda pozwalająca na dodawanie produktów do koszyka:

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Rysunek 20 - funkcja odpowiedzialna za dodawanie do koszyka

Koszyk przechowywany jest w sesji. Metoda przyjmuje ID jako parametr, następnie próbuje znaleźć produkt o podanym ID w tabeli ‘products’ i przypisać go do zmiennej $product.

Do zmiennej $cart metoda pobiera aktualny stan koszyka, jeśli nie istnieje tworzy pusta tablice. Dodanie lub aktualizowanie produktu w koszyku następuje poprzez sprawdzenie czy produkt o podanym ID znajduje się już w koszyku czy nie. Jeśli tak to zwiększamy jego ilość, jeśli nie to dodajemy go do koszyka z informacjami oraz ustawiamy jego ilość na 1. Na końcu stan koszyka zostaje zapisany w sesji i użytkownik zostaje przekierowany na poprzednia stronę wraz z komunikatem o pomyślnym dodaniu do koszyka.

Obraz zawierający tekst, zrzut ekranu, jedzenie

Opis wygenerowany automatycznie

To jest fragment kodu odpowiedzialny za wyświetlanie produktów w koszyku, na zdjeciu powyżej.

Obraz zawierający tekst, zrzut ekranu, Czcionka, wyświetlacz

Opis wygenerowany automatycznie

Rysunek 21 - fragment kodu odpowiedzialny za wyswietlanie produktów w koszyku

Po kliknieciu w przycisk ‘zamow’, aplikacja przekieruje nas do okna koszyka. Ta akcja realizowana jest za pomoca fukncji cart(), która zwraca widok ‘cart’

Obraz zawierający tekst, Czcionka, linia, biały

Opis wygenerowany automatycznie

Rysunek 22 - funkcja zwracająca widok koszyka

Widok koszyka zawiera aktualne produkty w koszyku, ich cene, wage/ilość, którą możemy aktualizowac, ich łączną cene oraz przycisk usuń:

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

Po kliknieciu w przycisk kontynuuj zakupy, aplikacja wróci nas do strony głównej z produktami, a przycisk kup zresetuje koszyk i wyśle do bazy danych, do tabeli ‘orders’, obecne zamówienie.

Fragment kody odpowiedzialny za usunięcie z koszyka:

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

Rysunek 23 - funkcja odpowiedzialna za usuwanie produktu z koszyka

Metoda przyjmuje obiekt request, przechowujący dane przesłane w formularzu. W metodzie nastepuje sprawdzenie czy w żądaniu znajduje się ID produktu, jeśli nie metoda zakończy się i przekieruje nas do strony koszyka. Jeśli jednak ID znajduje się w żądaniu, wtedy następuje pobranie do zmiennej $cart aktualnego stanu koszyka. Teraz metoda sprawdza czy czy produkt znajduje się w koszyku, jeśli tak to unset($cart[$request->id]) usunie go z koszyka i zapisze stan koszyka w sesji. Zostaje dodany komunikat o pomyślnym usunieciu z koszyka i przekierowanie do koszyka.

Fragment kodu odpowiedzialny za aktualizowanie ilości konkretnego produktu:

Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie

Rysunek 24 - funkcja aktaulizująca ilość produktów w koszyku

Tak samo jak w przypadku usunięcia z koszyka sprawdzane jest czy w żądaniu znajduje się ID produktu jak i jego ilość. Jeśli któryś z tych warunków nie jest spełniony następuje zakończenie metody i zwrócenie widoku koszyka. Jeśli natomiast oba waunki są spełnione pobieramy aktualny stan koszyka z sesji, i następuje aktualizacja ilości produktu o podanym ID na nową wartość podaną w requeście. Potem zostaje zapisany nowy stan koszyka w sesji i poinformowanie użytkownika o pomyślnej aktualizacji koszyka.

Fragment kodu odpowiedzialny zapisanie zamowienia do tabeli po kliknieciu w przycisk ‘kup’, znajduje się w pliku OrderController:

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Rysunek 25 - funkcja wysyłająca zamówienie do bazy

Metoda przyjmuje jako parametr obiekt request, zawierające dane o nowym zamówieniu. Do zmiennej $cart zostaje przypisany aktualny stan koszyka z sesji. Teraz następuje iteracja w pętli po każdym produkcie zawartym w koszyku z aktualnej sesji. W pętli dla każdego produktu tworzone jest nowe zamówienie z jego obecnymi informacjami takimi jak: data zamówienia, ilośc sztuk/kg, cena. Po zapisaniu danego zamówienia, koszyk w sesji zostaje opróżniony i uzytkownik jest przekierowywany do głównej strony z produktami z informacja, że jego zamówienie zostało pomyślnie złożone.

Strona przedstawiająca aktualne zamówienia dla obecnie zalogowanego użytkownika:

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

Fragment kodu odpowiedzialny za wyświetlanie zamówień, zawarty w pliku OrderController:

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

Rysunek 26 - funkcja wyświetlająca zamówienia użytkownika

Metoda sprawdza czy użytkownik jest zalogowany, jeśli nie to przekierowuje go do strony logowania, a jeśli jest zalogowany, to przypisuje do zmiennej $userId, ID aktualnie zalogowanego użytkownika. Do zmiennej $orders zostają przypisane wszystkie zamówienia z tabeli ‘orders’ dla użytkownika o podanym ID (w naszym przypadku aktualnie zalogowanego). Na koniec zostaje zwrócony widok ‘orders.show’ wraz z tablica orders, zawierającą wszystkie zamówienia.

Strona z zamówieniami umożliwia użytkownikowi usuwanie zamówień. To fragment kodu za to odpowiedzialny:

Obraz zawierający tekst, Czcionka, linia, zrzut ekranu

Opis wygenerowany automatycznie

Rysunek 27 - funkcja usuwająca zamówienia

Metoda destroy(), przyjmuje jako parametr obiekt typu Order. Następnie pobrany obiekt $order zostaje usunięte z bazy i aplikacja przekieruje nas na stronę zamówień.

Obraz zawierający tekst, Czcionka, linia, numer

Opis wygenerowany automatycznie

Po kliknięciu ‘więcej szczegółów’ w karcie produktu, aplikacja otworzy szczegółową kartę z danymi o produkcie. Realizuje to metoda show() w ProductController:

Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie

Rysunek 28 - funkcja zwracająca szczegółowy widok produktu

Metoda ta przyjmuje jako parametr obiekt typu Product, i zwraca widok ‘products.show’ z przekazaniem do niego parametru.

Aby móc zapisywać własne logi do pliku, musimy wykonać polecenie: ‘php artisan make:middleware LogActivity’, utworzy to nowy middleware w aplikacji o podanej nazwie. Middleware jest warstwą pośrednia, która może być wykonywana przed lub po przetworzeniu żądania http przez aplikacje. W miejscu ‘Log::channel(…)’ następuje podmiana nazwy kanału na który chcemy wysyłać logi na poziomie ‘notice’, które informuje, że ktoś wykonał metodę http na określonym URI.

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

Rysunek 29 - funkcja ustawiająca nowy kanał dla logów

Zapisywanie logów do pliku o nazwie ’projekt\_errors.log’. W pliku ‘config/logging.php’ utworzony jest nowy kanał o nazwie ‘projekt\_errors’, który będzie rejestrował tylko błędy o poziomie ‘error’. Określenie sterownika jako ‘single’, oznacza, że kanał będzie używał pojedynczego sterownika logowania. Path określa ścieżkę do pliku, w którym będą rejestrowane błędy.

Obraz zawierający tekst, Czcionka, zrzut ekranu, linia

Opis wygenerowany automatycznie

Rysunek 30 - dodawanie nowego kanału

W pliku ‘bootstrap/app.php’ aby poprawnie działało zapisywanie logów do pliku ‘projekt\_errors’, trzeba ustawić obsługę raportowania wyjątków.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Rysunek 31 - przekierowanie błedów do kanału

$exceptions->report… raportuje każdy wyjątek przechwycony w aplikacji i wysyła go do kanału ‘projekt\_error’, który ustawiliśmy wcześniej.

Aby móc zmienić język wyświetlanych komunikatów o błędach walidacji musimy wykonać następujące komendy: ‘composer require laravel-lang/common - - dev’, ‘php artisan lang:add pl’ i na koniec ‘php artisan lang:update’.

Te polecenia wykonają po kolei: dodanie pakietu laravel-lang/common jako zależność. Pakiet ten zawiera zestaw plików tłumaczeń dla Laravela w wielu językach. Kolejna komenda dodaje nowy język tłumaczeń do aplikacji, w tym przypadku polski. Ostatnie polecenie aktualizuje tłumaczenia aplikacji na podstawie plików znajdujących się w pakiecie zainstalowanym pierwszą komendą. Oznacza to, że zostaną dodane brakujące tłumaczenia, a także zaktualizowane istniejące tłumaczenia.

W ‘config/app.php’ została wprowadzona zmiana aby błędy walidacji były wyświetlane w języku polskim. Dzięki poprzednio wykonanym poleceniom.

Obraz zawierający tekst, Czcionka, linia, zrzut ekranu

Opis wygenerowany automatycznie

Rysunek 32 - zmiana języka komunikatów