

Projekt końcowy cz. 2 — aplikacja PHP, kopia i prezentacja

Czas: 2 × 45 min

praca w zespołach 2–3-osobowych (te same co w zadaniu 33)

Cel: Zbudowanie w zespole prostej aplikacji PHP (PDO, zapytania przygotowane) korzystającej z bazy z części 1 przez konto z minimalnymi uprawnieniami, wykonanie i sprawdzenie kopii zapasowej, przygotowanie dokumentacji i 5-minutowej prezentacji projektu.

Po zajęciach potrafisz:

- łączy się z bazą przez PDO z trybem wyjątków, korzystając z pliku konfiguracyjnego poza katalogiem publicznym
- wyświetla i zapisuje dane zapytaniami przygotowanymi oraz wywołuje procedurę składowaną z obsługą błędów SIGNAL
- zabezpiecza wypisywane dane (htmlspecialchars) i parametry (biała lista, walidacja)
- wykonuje kopię bazy z procedurami i wyzwalaczami i sprawdza jej odtworzenie
- dokumentuje projekt (ERD, opis tabel, instrukcja) i prezentuje go w 5 minut

Sprzęt i materiały

- komputer z XAMPP (Apache, MariaDB, PHP 8) dla każdego członka zespołu
- baza z zadania 33 (projekt.sql) albo — gdy nie działa — wzorcowa baza `r2_34_warsztat_wzor.sql` z folderu Pliki_do_zadan
- rzutnik do prezentacji

Oprogramowanie

- **XAMPP** — aplikacja w `C:\xampp\htdocs\<projekt>`, konfiguracja poza `htdocs` apacheFriends.org
- **VS Code** — kod PHP code.visualstudio.com
- **draw.io (diagrams.net)** — ERD do dokumentacji app.diagrams.net
- **Edytor prezentacji (PowerPoint, Impress lub Google Slides)** — maks. 6 slajdów

Przygotowanie przed zajęciami

- Zaimportujcie aktualny projekt.sql z zadania 33 i sprawdźcie, że konto aplikacji działa (logowanie w konsoli `mysql -u <konto> -p`).
- Przypomnijcie sobie z zadań 2.19–2.22 połączenie PDO, prepare/execute, htmlspecialchars i obsługę wyjątków.

Bezpieczeństwo

- Aplikacja łączy się z bazą wyłącznie kontem `<baza>_app` z części 1, nigdy kontem root.
- Plik z hasłem (config.php) trzymajcie poza katalogiem htdocs i nie umieszczajcie go w repozytorium Git (dodajcie do .gitignore); w dokumentacji opiszcie tylko jego budowę.
- Każdą wartość od użytkownika przekazujcie jako parametr zapytania przygotowanego, a nazwy kolumn do sortowania lub filtry — przez białą listę. Każdą wartość wypisywaną na stronie przepuszczajcie przez htmlspecialchars.
- Użytkownikowi pokazujcie tylko komunikaty z waszych SIGNAL; inne błędy zapisujcie w dzienniku (error_log) i wyświetlajcie ogólny komunikat.
- Kopia bazy zawiera całe dane — przechowujcie ją w folderze zespołu, nie w htdocs.

Przebieg zadania

1 Struktura aplikacji i połączenie

Wymagania minimalne: lista danych z widoku z filtrem, formularz dodawania rekordu, wywołanie procedury z części 1 przez formularz, czytelne komunikaty błędów.

```
C:\xampp\<projekt>\_config\config.php  <- dsn, użytkownik, hasło (return [...])
C:\xampp\htdocs\<projekt>\db.php      <- funkcja połącz(): PDO i funkcja e() =
htmlspecialchars
C:\xampp\htdocs\<projekt>\index.php   <- lista z widoku + filtr
C:\xampp\htdocs\<projekt>\dodaj.php  <- formularz INSERT
C:\xampp\htdocs\<projekt>\operacja.php <- formularz wywołujący procedurę (CALL)
```

1. Utwórzcie pliki według struktury z ramki. W db.php połączenie PDO z charset=utf8mb4, PDO::ERRMODE_EXCEPTION i domyślnym trybem FETCH_ASSOC; plik konfiguracyjny wczytujcie przez require ze ścieżki poza htdocs.
2. Sprawdźcie, że adres `http://localhost/<projekt>/config.php` nie istnieje (plik jest poza katalogiem publicznym).

2 Funkcje aplikacji

1. Strona index.php: tabela danych z widoku z części 1 i filtr (lista select) przekazywany metodą GET. Dozwolone wartości filtra sprawdźcie białą listą (in_array z true), a wartość przekażcie do zapytania jako parametr.
2. Strona dodaj.php: formularz POST dodający rekord do głównej tabeli (zapytanie przygotowane INSERT), walidacja pól (filter_input, wymagane pola), po zapisie przekierowanie (header Location) i komunikat.
3. Strona operacja.php: formularz wywołujący procedurę z części 1 (`CALL nazwa(?, ?, ?)` przez prepare/execute). Błąd z SIGNAL (SQLSTATE 45000 w `$e->errorInfo[0]`) pokażcie użytkownikowi, pozostałe zapiszcie przez error_log i pokażcie ogólny komunikat.
4. Przetestujcie: poprawne dodanie, błędne dane w formularzu, wywołanie procedury kończące się każdym z waszych komunikatów SIGNAL, wartość filtra spoza listy, tekst z apostrofem i znakami < > w polu formularza. Sprawdźcie w tabeli dziennika, że zmiany z aplikacji zapisują nazwę konta aplikacji.

3 Kopia zapasowa i odtworzenie

mysqldump domyślnie zapisuje wyzwalacze, ale NIE zapisuje procedur i funkcji — potrzebna jest opcja --routines.

```
mysqldump -u root --single-transaction --routines --triggers --databases <baza> >
kopia_<baza>.sql
```

1. Wykonajcie kopię poleceniem z ramki (w Shell XAMPP) i sprawdźcie w pliku, że zawiera CREATE PROCEDURE i CREATE TRIGGER. Porównajcie z kopią bez --routines.
2. Odtwórzcie kopię pod inną nazwą bazy albo na komputerze innego członka zespołu i sprawdźcie, że aplikacja działa na odtworzonej bazie (po utworzeniu konta aplikacji).

4 Dokumentacja i prezentacja

Prezentacja: 5 minut, maks. 6 slajdów, każdy członek zespołu mówi; po niej 2–3 pytania nauczyciela lub klasy.

1. Dokumentacja (plik PDF lub DOCX): opis dziedziny, ERD, opis każdej tabeli (kolumny, klucze, ograniczenia), lista widoków, procedur i wyzwalaczy z celem, uprawnienia konta aplikacji, instrukcja instalacji (import bazy, utworzenie konta, plik config.php, adres strony) i procedura kopii.
2. Prezentacja: problem i dziedzina, ERD, najciekawszy element logiki w bazie (procedura lub wyzwalacz), pokaz działania aplikacji z jednym błędem SIGNAL, zabezpieczenia, wnioski i podział pracy.
3. Oddajcie folder zespołu: kod aplikacji (bez config.php z hasłem — tylko config.przyklad.php), projekt.sql, kopia bazy, dokumentacja, prezentacja.

W sprawozdaniu umieść

- ☐ dokumentacja projektu (ERD, opis tabel i obiektów, uprawnienia, instrukcja instalacji, procedura kopii)
- ☐ tabela testów aplikacji (dane wejściowe, oczekiwany i otrzymany wynik)
- ☐ rzuty: lista z filtrem, poprawne dodanie, komunikat błędu z SIGNAL, wpis w dzienniku z kontem aplikacji
- ☐ plik kopii bazy i opis testu odtworzenia
- ☐ prezentacja i krótka samoocena pracy zespołu

Pytania kontrolne

1. Dlaczego plik config.php leży poza katalogiem htdocs i co mogłoby się stać, gdyby serwer przestał interpretować PHP?
2. Dlaczego filtra statusu nie wystarczy przekazać jako parametr, tylko sprawdza się go też białą listą?
3. Po co przekierowanie po zapisie formularza (POST → przekierowanie → GET)?
4. Co by się stało przy odtwarzaniu kopii wykonanej bez --routines i jak szybko to wykryć?

Kryteria oceny

Kryterium	Punkty
Połączenie PDO, konfiguracja poza httdocs, konto aplikacji (nie root)	3
Lista z filtrem (biała lista + parametr) i formularz INSERT z walidacją i przekierowaniem	4
Wywołanie procedury z obsługą komunikatów SIGNAL i ukrywaniem innych błędów; htmlspecialchars	4
Kopia z procedurami i wyzwalaczami oraz test odtworzenia	2
Dokumentacja (ERD, opis tabel i obiektów, instrukcja)	3
Prezentacja (czas, pokaz działania, udział całego zespołu, odpowiedzi na pytania)	4
Razem	20

Sprawozdanie przygotuj na wzorze dołączonym do zadania (plik .docx o tym samym numerze).