Cele pracy

- Zaprojektowanie i implementacja aplikacji webowej wspomagającej zarządzanie projektami w przedsiębiorstwach
- Umożliwienie zarządzania projektami, zadaniami, zespołami projektowymi oraz zasobami firmy, wspólna przestrzeń dyskowa, zespołem firmy
- Zapewnienie bezpiecznego dostępu do systemu poprzez mechanizmy uwierzytelniania i autoryzacji
- Rozdzielenie warstwy prezentacji od logiki biznesowej w architekturze klient–serwer
- Weryfikacja poprawności działania systemu poprzez testy integracyjne i jednostkowe

Celem pracy było zaprojektowanie i wykonanie zintegrowanego systemu webowego wspierającego zarządzanie projektami w organizacji. System umożliwia ewidencję projektów i zadań, zarządzanie zespołem projektowym, kontrolę statusów, alokację zasobów oraz generowanie raportów. Istotnym założeniem było zastosowanie architektury klient–serwer z wykorzystaniem REST API oraz zapewnienie bezpieczeństwa i spójności danych po stronie serwera.

Zakres prac

• Analiza problemu i istniejących rozwiązań

Przeprowadzono analizę problematyki zarządzania projektami w organizacjach oraz przegląd dostępnych narzędzi klasy Project Management, takich jak Jira, Azure DevOps i Trello. Na tej podstawie określono braki funkcjonalne oraz potrzeby użytkowników. Wyniki analizy posłużyły do sformułowania wymagań dla projektowanego systemu.

• Projekt systemu i modelu danych

Opracowano koncepcję systemu w architekturze klient–serwer z wykorzystaniem REST API. Zaprojektowano strukturę aplikacji, role użytkowników oraz mechanizmy autoryzacji. Integralną częścią było zaprojektowanie relacyjnej bazy danych, uwzględniającej zależności pomiędzy użytkownikami, projektami, zadaniami i zasobami.

• Uwierzytelnianie oparte o sesje użytkownika

System wykorzystuje uwierzytelnianie sesyjne po stronie serwera. Po poprawnym zalogowaniu dane użytkownika (m.in. identyfikator, rola oraz identyfikator firmy) są zapisywane w sesji. Każde kolejne żądanie do aplikacji jest weryfikowane na podstawie aktywnej sesji, bez konieczności ponownego logowania.

• Filtry autoryzacji dla widoków i REST API

Zastosowano filtry autoryzacyjne po stronie serwera, które chronią zarówno widoki aplikacji, jak i endpointy REST API. Filtry blokują dostęp do zasobów w przypadku braku aktywnej sesji lub niewystarczających uprawnień, zwracając odpowiednie kody HTTP (np. 401). Dzięki temu logika bezpieczeństwa nie jest zależna od warstwy frontendowej.

• Ograniczanie dostępu do danych (scoping danych)

Dostęp do danych jest dodatkowo zawężany na podstawie kontekstu organizacyjnego oraz przypisań użytkownika. Użytkownik widzi tylko projekty, zadania i zasoby, do których został przypisany lub które należą do jego firmy. Zapewnia to izolację danych pomiędzy organizacjami oraz zwiększa bezpieczeństwo systemu.

Przykłady rozdzielenia warstwy prezentacji od logiki biznesowej

- **Autoryzacja i kontrola dostępu**

Warstwa prezentacji nie decyduje o tym, do jakich danych użytkownik ma dostęp. Sprawdzenie uprawnień, roli użytkownika oraz przypisań do projektów odbywa się po stronie serwera, w filtrach autoryzacyjnych i logice API. Frontend jedynie wysyła żądania i reaguje na odpowiedzi serwera.

- **Obsługa operacji biznesowych przez REST API**

Tworzenie i edycja projektów, zadań, przypisywanie użytkowników oraz alokacja zasobów są realizowane poprzez endpointy REST API. Backend odpowiada za wykonanie operacji, zapis do bazy danych oraz egzekwowanie reguł biznesowych, natomiast frontend wyłącznie prezentuje wynik operacji użytkownikowi.

• Implementacja aplikacji webowej

Zaimplementowano warstwę backendową w technologii PHP z wykorzystaniem frameworka CodeIgniter 4 oraz relacyjnej bazy danych MySQL. Warstwa frontendowa została wykonana jako aplikacja kliencka w technologii Vue.js 3. System realizuje kluczowe funkcje zarządzania projektami, zadaniami, zespołami, zasobami oraz raportami.

- Implementacja aplikacji webowej

Zaimplementowano aplikację webową w architekturze klient–serwer z wyraźnym rozdzieleniem warstwy prezentacji od logiki biznesowej. Warstwa backendowa została wykonana w języku PHP z wykorzystaniem frameworka CodeIgniter 4 i udostępnia funkcjonalność systemu w postaci REST API. Odpowiada ona za obsługę żądań HTTP, realizację logiki biznesowej, walidację danych, autoryzację użytkowników oraz komunikację z relacyjną bazą danych MySQL.

Warstwa frontendowa została zrealizowana jako aplikacja kliencka w technologii Vue.js 3, działająca w przeglądarce internetowej. Interfejs użytkownika komunikuje się z backendem wyłącznie poprzez REST API, pobierając i modyfikując dane systemowe. Frontend odpowiada za prezentację danych, obsługę formularzy oraz interakcję użytkownika z systemem.

W ramach implementacji wykonano kluczowe moduły systemu, w tym zarządzanie użytkownikami, projektami, zadaniami, zespołami projektowymi oraz zasobami przedsiębiorstwa. Zaimplementowano również obsługę plików projektowych oraz generowanie raportów w formacie CSV, których metadane są przechowywane w bazie danych, a treść w systemie plików serwera.

Istotnym elementem implementacji było zastosowanie mechanizmów bezpieczeństwa, takich jak sesje użytkownika, filtry autoryzacji oraz walidacja danych po stronie serwera. Dzięki temu zapewniono spójność danych, kontrolę dostępu do zasobów systemu oraz odporność na nieautoryzowane operacje.

• Testowanie i weryfikacja działania systemu

Przeprowadzono testy integracyjne typu health-check oraz testy jednostkowe i manualne REST API z wykorzystaniem narzędzia Postman. Zweryfikowano poprawność działania kluczowych funkcji systemu oraz spełnienie założonych wymagań funkcjonalnych i niefunkcjonalnych. Na podstawie wyników testów potwierdzono poprawność i stabilność rozwiązania.

Kontrolery (app/Controllers/) — zarządzają logiką żądań i odpowiedzi; wywołują modele, ładują widoki. Przykłady: Admin.php, Auth.php,

Modele (app/Models/) — warstwa dostępu do danych; mapują tabele, walidacje i zapytania. Przykłady: ProjectModel.php, ResourceModel.php, UserModel.php, TaskModel.php.

Widoki (app/Views/) — szablony HTML/JS/CSS wyświetlane użytkownikowi. Przykłady: frontend.php, project.php, companies.php, admin.php. Widoki są renderowane przez kontrolery.

Plik Routes.php zawiera instrukcje, które mapują żądania URL (URI) na odpowiednie kontrolery i metody. Oto co zazwyczaj zawiera ten plik:

Filtry (app/Filters/) — middleware przed/po żądaniu (np. AuthFilter.php sprawdza autoryzację).

Konfiguracje (app/Config/) — ustawienia aplikacji i frameworka: App.php, Routes.php, Database.php, Services it

Baza danych (database/, app/Database/) — migracje, seedy i schematy (database/schema.sql, install.sql, app/Database/Migrations/).