

# TO 2024/25 Projekt - Backend Service

## Opis projektu

BoardGameHub to pełnoprawny backend systemu do przeglądania, oceniania i recenzowania gier planszowych. Umożliwia on użytkownikom przeglądanie gier, filtrowanie ich według kategorii i lokalizacji, wystawianie ocen i recenzji, a także zarządzanie kontem. Aplikacja wspiera rejestrację i logowanie, zarządzanie sesjami, role użytkowników oraz panel administracyjny z metrykami i możliwością moderowania treści.

Projekt został zrealizowany w języku Java 21, całkowicie bez użycia zewnętrznych bibliotek czy frameworków (takich jak Spring, Jackson, Hibernate).

## Architektura i zastosowane wzorce

System został zbudowany zgodnie z podejściem warstwowym (Layered Architecture), z separacją logiki HTTP, middleware, kontrolerów, serwisów i warstwy persystencji. Jednocześnie projekt stosuje elementy Domain-Driven Design (DDD) oraz MVC.

- Model: DTO i encje (*model/*)
- Controller: obsługa endpointów (np, *AuthController*, *GameController*)
- View: generowana opowiedź JSON (*ResponseContext*, *JsonUtil*)

## Zastosowane wzorce projektowe (GoF)

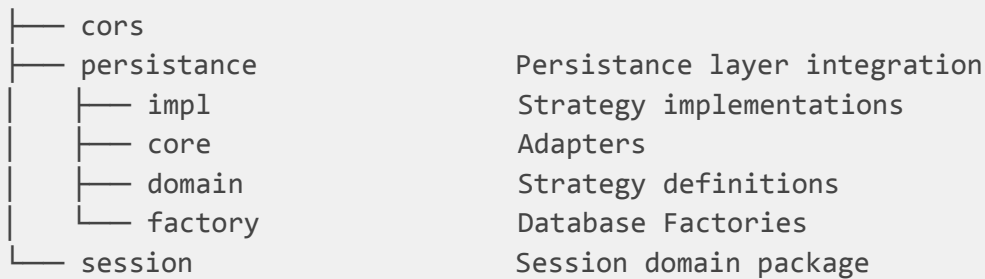
Wzorzec	Przykład użycia
Singleton	AppConfig, HttpServerSingleton, ComponentRegistry
Builder	RouteBuilder, ResponseContext (builder)
Strategy	PasswordHasher, Repository (all repos)
Factory	RepositoryFactory, PasswordHasherFactory
Adapter	JdbcPostgresAdapter, RedisAdapter
Decorator	Middlewares (composable middleware pattern)

## Dependency Injection

DI odbywa się poprzez centralną klasę ComponentRegistry utrzymującą własny rejestr komponentów.

## Struktura projektu

project	
├── middleware	Reusable middlewares
├── category	Category domain package
│   ├── dto	
├── util	Shared utilities
│   ├── json	
│   └── passwordHasher	Hasher strategies, impl and factory
│       └── impl	
├── config	Application configuration
├── auth	Auth domain package
│   └── dto	
├── health	Health domain package (healthcheck)
├── user	User domain package
│   └── dto	
├── shared	Shared functionalities, DTOs
│   ├── dto	
│   └── error	
├── game	Game domain package
│   └── dto	
├── testEndpoint	Test functionality
├── preference	Preference domain package
│   └── dto	
├── loader	Application loader and component registry
├── http	HTTP integration layer
│   ├── core	
│   │   ├── context	
│   │   └── routing	
│   ├── utils	
│   └── handlers	
├── review	Review domain package
│   └── dto	



## Szczegóły techniczne

Baza danych: PostgreSQL (obsługa transakcji przez `JdbcPostgresAdapter.transaction(...)`)

Sesje: zarządzanie przez pamięć lub Redis (wzorzec strategii)

Serializacja JSON: refleksyjny parser (`JsonUtil` + `JsonMapper`), obsługuje DTO, listy, UUID, `Instant`, `Date`, enumy

Routing: własny system zbudowany na `HttpServer` i DSL-owym API (`RouteRegistry`, `RouteBuilder`)

Middleware: każdy request przechodzi przez łańcuch middleware (np. logowanie, autoryzacja, obsługa błędów)

Obsługa CORS i cookies (`CorsMiddleware`)

Logika RBAC (`AuthMiddleware.requireRole(...)`)

Testy jednostkowe z JUnit 5

Konfiguracja z pliku `.properties` (`AppConfig`)

Obsługa błędów i automatyczna serializacja wyjątków