Quikk Framework v0.2 - Dokumentasi Lengkap

Versi: 0.2

Tanggal Rilis: 27 Juni 2025

Bahasa: PHP 8.x, MySQL 8.x

Struktur: Native PHP, OOP, Modular

Lisensi: MIT (opsional)

# 📁 Struktur Direktori

quikk\_v0.2/

├── app/

│ ├── Config/

│ │ └── config.php ✅ Konfigurasi global

│ ├── Database/

│ │ ├── DB.php ✅ Koneksi database (PDO singleton)

│ │ ├── schema.sql ✅ Struktur tabel utama

│ │ ├── seed\_users.sql ✅ Seeder user default (admin, user)

│ │ ├── seed\_menus.sql ✅ Seeder menu default

│ │ ├── seed\_access.sql ✅ Seeder hak akses default

│ │ └── SeederSentinel.php ✅ Sentinel agar seeder tidak dijalankan ulang

│ └── Services/

│ ├── AuthService.php ✅ Login aman, session, CSRF, logging

│ ├── MenuManager.php ✅ Ambil menu sesuai level user

│ ├── ModuleRegistrar.php ✅ Daftarkan modul + SQL otomatis

│ └── Router.php ✅ Routing terenkripsi + validasi hak akses

├── pages/ ✅ Folder halaman aplikasi (kosong)

├── public/ ✅ Aset statis (kosong)

├── logs/ ✅ (Opsional) untuk log aktivitas

├── README.md 📘 Dokumentasi developer

├── SRS\_Quikk\_v0.2.docx 📄 Dokumen kebutuhan sistem (SRS)

└── Dokumentasi\_Lengkap\_Quikk\_v0.2.docx 📄 Dokumentasi lengkap + source code🚀

Cara Memulai

1. Clone atau ekstrak file

2. Import schema.sql ke database Anda

3. Jalankan seeder SQL jika perlu:

- seed\_users.sql, seed\_menus.sql, seed\_access.sql

4. Edit file app/Config/config.php sesuai setting lokal

# 🔐 Sistem Keamanan

- Password disimpan dengan password\_hash()

- Login menggunakan AuthService (CSRF, rate limiter, logging)

- Routing terenkripsi berbasis token

- Hak akses dan menu dinamis berdasarkan level

# 📦 Class Utama

DB - Singleton koneksi PDO

AuthService - Login, logout, session, CSRF, logging

MenuManager - Ambil menu berdasarkan hak akses user

Router - Resolusi halaman berdasarkan token

ModuleRegistrar - Registrasi modul baru

SeederSentinel - Cegah seeder dijalankan ulang

# ⚙️ Konfigurasi config.php

Contoh konfigurasi utama yang digunakan:

<?php  
return [  
  
 // DATABASE  
 'db\_host' => 'localhost',  
 'db\_name' => 'quikk',  
 'db\_user' => 'root',  
 'db\_pass' => '',  
  
 // APLIKASI  
 'base\_url' => 'http://localhost/quikk/',  
 'base\_path' => realpath(\_\_DIR\_\_ . '/../../'),  
 'environment' => 'development',  
 'debug' => true,  
  
 // AUTH & SECURITY  
 'rate\_limit\_attempts' => 5,  
 'rate\_limit\_window' => 300,  
 'enable\_csrf' => true,  
 'session\_timeout' => 3600,  
  
 // LOGGING  
 'log\_login\_to\_file' => false,  
 'log\_path' => \_\_DIR\_\_ . '/../../logs/login.log',  
  
 // ROUTING  
 'token\_length' => 10,  
 'pages\_path' => \_\_DIR\_\_ . '/../../pages/',  
 'fallback\_404' => 'pages/error/404.php',  
  
 // MODUL & SEEDER  
 'module\_sql\_path' => \_\_DIR\_\_ . '/../Database/modules/',  
 'modules\_table' => 'modules',  
 'seeder\_lock\_file' => \_\_DIR\_\_ . '/../../.seeder.lock',  
];

# 🧪 Testing Default Login

Admin - admin / admin

User - user / user

# 📌 Catatan

- Halaman aplikasi ada di folder /pages/

- URL akan diresolusi oleh Router.php

- Class modular dan terpisah

- Modul baru daftarkan via ModuleRegistrar

# 🧩 Cara Registrasi Modul Baru

1. Buat file halaman modul di /pages/

2. Letakkan file SQL di /app/Database/modules/

3. Jalankan kode berikut:

use App\Database\DB;  
use App\Services\ModuleRegistrar;  
  
require\_once '../app/Database/DB.php';  
require\_once '../app/Services/ModuleRegistrar.php';  
  
$db = DB::getConnection();  
$config = include '../app/Config/config.php';  
  
$registrar = new ModuleRegistrar($db, 'pegawai', $config);  
$registrar->register();

4. Tambahkan entry menus, access\_rights, route\_tokens

5. Sistem akan deteksi otomatis apakah modul sudah terdaftar

✅ Keunggulan:

- Tidak perlu ubah core class

- Modul terisolasi dan reusable

- Cukup 1 baris via ModuleRegistrar

# 📄 Lampiran: Source Code Class dan SQL

## DB.php

<?php  
namespace App\Database;  
  
use PDO;  
use PDOException;  
  
class DB  
{  
 private static ?PDO $connection = null;  
  
 public static function getConnection(): PDO  
 {  
 if (self::$connection === null) {  
 $config = include(\_\_DIR\_\_ . '/../Config/config.php');  
 $dsn = "mysql:host={$config['db\_host']};dbname={$config['db\_name']};charset=utf8mb4";  
 try {  
 self::$connection = new PDO($dsn, $config['db\_user'], $config['db\_pass']);  
 self::$connection->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);  
 } catch (PDOException $e) {  
 die("Database error: " . $e->getMessage());  
 }  
 }  
 return self::$connection;  
 }  
}

## AuthService.php

<?php  
namespace App\Services;  
  
use PDO;  
  
class AuthService  
{  
 private PDO $db;  
 private array $config;  
  
 public function \_\_construct(PDO $db, array $config)  
 {  
 $this->db = $db;  
 $this->config = $config;  
 session\_start();  
 $this->initCsrfToken();  
 }  
  
 public function login(string $username, string $password, string $csrfToken): bool  
 {  
 $ip = $\_SERVER['REMOTE\_ADDR'];  
  
 if ($this->config['enable\_csrf'] && !$this->validateCsrfToken($csrfToken)) {  
 $this->logAttempt($username, 'fail', $ip);  
 return false;  
 }  
  
 if ($this->isRateLimited($ip)) {  
 $this->logAttempt($username, 'blocked', $ip);  
 return false;  
 }  
  
 if (!$this->validateInput($username, $password)) {  
 $this->logAttempt($username, 'fail', $ip);  
 return false;  
 }  
  
 $stmt = $this->db->prepare("SELECT \* FROM users WHERE username = :username");  
 $stmt->bindParam(':username', $username);  
 $stmt->execute();  
 $user = $stmt->fetch(PDO::FETCH\_ASSOC);  
  
 if ($user && password\_verify($password, $user['password\_hash'])) {  
 session\_regenerate\_id(true);  
 $\_SESSION['user'] = [  
 'username' => $user['username'],  
 'level' => $user['level']  
 ];  
 $this->logAttempt($username, 'success', $ip);  
 return true;  
 } else {  
 $this->logAttempt($username, 'fail', $ip);  
 return false;  
 }  
 }  
  
 public function logout(): void  
 {  
 session\_unset();  
 session\_destroy();  
 }  
  
 public function getUser(): ?array  
 {  
 return $\_SESSION['user'] ?? null;  
 }  
  
 public function isLoggedIn(): bool  
 {  
 return isset($\_SESSION['user']);  
 }  
  
 private function initCsrfToken(): void  
 {  
 if (empty($\_SESSION['csrf\_token'])) {  
 $\_SESSION['csrf\_token'] = bin2hex(random\_bytes(32));  
 }  
 }  
  
 public function getCsrfToken(): string  
 {  
 return $\_SESSION['csrf\_token'];  
 }  
  
 private function validateCsrfToken(string $token): bool  
 {  
 return hash\_equals($\_SESSION['csrf\_token'], $token);  
 }  
  
 private function validateInput(string $username, string $password): bool  
 {  
 return preg\_match('/^[a-zA-Z0-9\_]{4,30}$/', $username) &&  
 strlen($password) >= 6 && strlen($password) <= 255;  
 }  
  
 private function logAttempt(string $username, string $status, string $ip): void  
 {  
 $stmt = $this->db->prepare("INSERT INTO login\_logs (username, status, ip\_address) VALUES (:username, :status, :ip)");  
 $stmt->execute([  
 ':username' => $username,  
 ':status' => $status,  
 ':ip' => $ip  
 ]);  
 }  
  
 private function isRateLimited(string $ip): bool  
 {  
 $stmt = $this->db->prepare("SELECT COUNT(\*) FROM login\_logs WHERE ip\_address = :ip AND status = 'fail' AND timestamp > (NOW() - INTERVAL :window SECOND)");  
 $stmt->bindValue(':ip', $ip);  
 $stmt->bindValue(':window', $this->config['rate\_limit\_window'], PDO::PARAM\_INT);  
 $stmt->execute();  
 $failCount = $stmt->fetchColumn();  
  
 return $failCount >= $this->config['rate\_limit\_attempts'];  
 }  
}

## MenuManager.php

<?php  
namespace App\Services;  
  
use PDO;  
  
class MenuManager  
{  
 private PDO $db;  
 private string $userLevel;  
  
 public function \_\_construct(PDO $db, string $userLevel)  
 {  
 $this->db = $db;  
 $this->userLevel = $userLevel;  
 }  
  
 public function getMenu(): array  
 {  
 $stmt = $this->db->prepare("  
 SELECT m.menu\_name, t.url\_path  
 FROM menus m  
 JOIN access\_rights a ON m.menu\_code = a.menu\_code  
 JOIN route\_tokens t ON t.token = m.url\_token  
 WHERE a.level = :level AND m.is\_active = 1  
 ORDER BY m.order\_num ASC  
 ");  
 $stmt->execute([':level' => $this->userLevel]);  
 return $stmt->fetchAll(PDO::FETCH\_ASSOC);  
 }  
}

## Router.php

<?php  
namespace App\Services;  
  
use PDO;  
  
class Router  
{  
 private PDO $db;  
 private string $token;  
 private array $config;  
  
 public function \_\_construct(PDO $db, string $token, array $config)  
 {  
 $this->db = $db;  
 $this->token = $token;  
 $this->config = $config;  
 }  
  
 public function resolve(): string  
 {  
 $stmt = $this->db->prepare("SELECT url\_path FROM route\_tokens WHERE token = :token");  
 $stmt->execute([':token' => $this->token]);  
 $path = $stmt->fetchColumn();  
  
 if ($path && file\_exists($this->config['pages\_path'] . "$path.php")) {  
 return $this->config['pages\_path'] . "$path.php";  
 }  
  
 return $this->config['fallback\_404'];  
 }  
  
 public function hasAccess(string $userLevel): bool  
 {  
 $stmt = $this->db->prepare("  
 SELECT COUNT(\*) FROM access\_rights ar  
 JOIN menus m ON ar.menu\_code = m.menu\_code  
 WHERE ar.level = :level AND m.url\_token = :token  
 ");  
 $stmt->execute([  
 ':level' => $userLevel,  
 ':token' => $this->token  
 ]);  
 return $stmt->fetchColumn() > 0;  
 }  
}

## ModuleRegistrar.php

<?php  
namespace App\Services;  
  
use PDO;  
  
class ModuleRegistrar  
{  
 private PDO $db;  
 private string $moduleName;  
 private array $config;  
  
 public function \_\_construct(PDO $db, string $moduleName, array $config)  
 {  
 $this->db = $db;  
 $this->moduleName = $moduleName;  
 $this->config = $config;  
 }  
  
 public function register(): void  
 {  
 if (!$this->isRegistered()) {  
 $this->importSQL();  
 $this->insertMenus();  
 $this->insertAccessRights();  
 $this->markRegistered();  
 }  
 }  
  
 private function importSQL(): void  
 {  
 $sqlFile = $this->config['module\_sql\_path'] . $this->moduleName . '.sql';  
 if (file\_exists($sqlFile)) {  
 $sql = file\_get\_contents($sqlFile);  
 $this->db->exec($sql);  
 }  
 }  
  
 private function insertMenus(): void {}  
 private function insertAccessRights(): void {}  
  
 private function isRegistered(): bool  
 {  
 $stmt = $this->db->prepare("SELECT COUNT(\*) FROM " . $this->config['modules\_table'] . " WHERE name = :name");  
 $stmt->execute([':name' => $this->moduleName]);  
 return $stmt->fetchColumn() > 0;  
 }  
  
 private function markRegistered(): void  
 {  
 $stmt = $this->db->prepare("INSERT INTO " . $this->config['modules\_table'] . " (name, installed\_at) VALUES (:name, NOW())");  
 $stmt->execute([':name' => $this->moduleName]);  
 }  
}

## SeederSentinel.php

<?php  
namespace App\Database;  
  
class SeederSentinel  
{  
 private string $lockFile;  
  
 public function \_\_construct(array $config)  
 {  
 $this->lockFile = $config['seeder\_lock\_file'];  
 }  
  
 public function hasRun(): bool  
 {  
 return file\_exists($this->lockFile);  
 }  
  
 public function markAsRun(): void  
 {  
 file\_put\_contents($this->lockFile, date('Y-m-d H:i:s'));  
 }  
}

## config.php

<?php  
return [  
  
 // DATABASE  
 'db\_host' => 'localhost',  
 'db\_name' => 'quikk',  
 'db\_user' => 'root',  
 'db\_pass' => '',  
  
 // APLIKASI  
 'base\_url' => 'http://localhost/quikk/',  
 'base\_path' => realpath(\_\_DIR\_\_ . '/../../'),  
 'environment' => 'development',  
 'debug' => true,  
  
 // AUTH & SECURITY  
 'rate\_limit\_attempts' => 5,  
 'rate\_limit\_window' => 300,  
 'enable\_csrf' => true,  
 'session\_timeout' => 3600,  
  
 // LOGGING  
 'log\_login\_to\_file' => false,  
 'log\_path' => \_\_DIR\_\_ . '/../../logs/login.log',  
  
 // ROUTING  
 'token\_length' => 10,  
 'pages\_path' => \_\_DIR\_\_ . '/../../pages/',  
 'fallback\_404' => 'pages/error/404.php',  
  
 // MODUL & SEEDER  
 'module\_sql\_path' => \_\_DIR\_\_ . '/../Database/modules/',  
 'modules\_table' => 'modules',  
 'seeder\_lock\_file' => \_\_DIR\_\_ . '/../../.seeder.lock',  
];

## schema.sql

-- File: /app/Database/schema.sql  
-- Tabel users  
CREATE TABLE IF NOT EXISTS users (  
 id INT AUTO\_INCREMENT PRIMARY KEY,  
 username VARCHAR(50) NOT NULL UNIQUE,  
 password\_hash VARCHAR(255) NOT NULL,  
 level VARCHAR(20) NOT NULL,  
 created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP  
);  
  
-- Tabel levels (opsional: untuk referensi level user)  
CREATE TABLE IF NOT EXISTS levels (  
 level VARCHAR(20) PRIMARY KEY,  
 description VARCHAR(100) NOT NULL  
);  
  
-- Tabel login\_logs (opsional: jika log ingin disimpan di DB bukan file)  
CREATE TABLE IF NOT EXISTS login\_logs (  
 id INT AUTO\_INCREMENT PRIMARY KEY,  
 username VARCHAR(50) NOT NULL,  
 status ENUM('success', 'fail', 'blocked') NOT NULL,  
 ip\_address VARCHAR(45),  
 timestamp TIMESTAMP DEFAULT CURRENT\_TIMESTAMP  
);

## seed\_users.sql

-- Seeder default user  
INSERT INTO users (username, password\_hash, level)  
VALUES  
 ('admin', '$2y$10$eImiTXuWVxfM37uY4JANj.QzZcXaBd5yyWphGLkFad9Xuq5lqWdTy', 'admin'),  
 ('user', '$2y$10$uCjI9L7jDTNHoYHnLfE9feJkYv7aZQQaHJTCiZk8id0mSGIoN9uVS', 'user');

## seed\_menus.sql

-- Seeder default menu  
INSERT INTO menus (menu\_code, menu\_name, url\_token, parent\_code, order\_num, is\_active)  
VALUES  
 ('DASH', 'Dashboard', 'a1b2c3d4e5', NULL, 1, 1),  
 ('USR', 'Manajemen User', 'f6g7h8i9j0', NULL, 2, 1),  
 ('SET', 'Pengaturan', 'k1l2m3n4o5', NULL, 3, 1);

## seed\_access.sql

-- Seeder hak akses default  
INSERT INTO access\_rights (level, menu\_code)  
VALUES  
 ('admin', 'DASH'),  
 ('admin', 'USR'),  
 ('admin', 'SET'),  
 ('user', 'DASH')