0)Introducción

Este proyecto proporciona una base completa para una **API REST en PHP sin frameworks**, con una arquitectura modular y escalable que incluye enrutamiento, controladores, autenticación JWT, migraciones y gestión del flujo HTTP mediante middlewares. Utiliza **Composer y autoload PSR-4**, e incorpora una **interfaz web minimalista** para probar el flujo completo de autenticación, CRUD de tareas e integración frontend-backend para la API. Esta guía explica su instalación, configuración y puesta en funcionamiento.

Las 3 primeras páginas contienen lo imprescindible, y luego el resto es información adicional.

1) Requisitos

- PHP 8.1+ (ideal 8.2) con PDO/pdo-mysql (y opcionalmente sqlite3/pdo-sqlite para tests).
- Composer 2.x
- MySQL 8.x (o MariaDB) y opcionalmente phpMyAdmin.
- Un programa de compresión similar a winrar.
- Navegador web + Postman/cURL (opcional).
- Xamp/Wamp (opcional).
- Guía adaptada para Windows.
- Desactivar antivirus si da algún problema.

2) Instalación y migraciones (Backend + Frontend)

2.1) Configurar variables de entorno (.env)

Ajusta los valores de los dos **.env,** de ser necesario, como el puerto o el usuario y password de bd. El bootstrap debe cargarlo al inicio.

```
APP_ENV=local
JWT_SECRET=pon_un_secreto_largo_seguro
DB_HOST=localhost
DB_PORT=3306
DB_NAME=gestor_tareas
DB_USER=root
DB_PASSWORD=
```

2.2) Creacion DB

Puedes hacerlo de 2 formas:

Ejecutalo en la consola de comandos de myql:

```
mysql -u root -p -e "CREATE DATABASE IF NOT EXISTS gestor_tareas
CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;"
```

o ejecútalo directamente en tu gestor de bbdd (como workbench o phpmyadmin):

```
CREATE DATABASE IF NOT EXISTS gestor_tareas CHARACTER SET utf8mb4
COLLATE utf8mb4_unicode_ci;
```

2.3) Ejecución rápida de migraciones

Dentro de php-puro-api

`composer install`

Crea bd+ tablas + usuario admin demo

`php scripts/migrate.php`

```
C:\xampp\htdocs\0PLANTILLAS\php-puro-api>php scripts/migrate.php

> Applying 001_create_users.sql

> Applying 002_create_tasks.sql

All migrations applied.
```

3) Lanzar la API + main page + integración API - Dos caminos

Primero desempaqueta el archivo “prueba-tecnica-antonio-gonzalez.rar” es el escritorio, y luego elegimos un camino a seguir:

3.A) XAMPP/Apache + MySQL

1) Inicia Apache y MySQL desde XAMPP/WAMP (activa mod-rewrite y AllowOverride All).
2) Coloca el proyecto en “/htdocs” o “/wwww” (las carpetas “integracion-api”, “main-page” y “php-puro-api” directamente en la raíz “/htdocs” o “/www”)
3) Instalar dependencias de composer ejecutando “`composer install`” en la ruta del proyecto de la API “php-puro-api” (si no lo has hecho ya).
5) Abrir cmd, moverte a la ruta de “php-puro-api“ y ejecutar migraciones usando “`php scripts/migrate.php`” (si no lo has hecho ya).
6) Crear un virtual host en xamp/wamp y añadirlo al archivo hosts de Windows (no extrictamente necesario, pero recomendable para debug) y reinicia apache.

- Si usas xamp, la ruta es “C:\xampp\apache\conf\extra\httpd-vhosts.conf”
- Si usas wamp, la ruta es “C:\wamp64\bin\apache\apache2.4.62.1\conf\extra\httpd-vhosts.conf”
- La ruta del archivo hosts de Windows es “C:\Windows\System32\drivers\etc\hosts”
- El vhosts debe ser similar a este:

```
<VirtualHost *:80>
    ServerName main-page.local
    DocumentRoot "C:/wamp64/www/main-page/public"
    <Directory "C:/wamp64/www/main-page/public">
        AllowOverride All
        Require all granted
    </Directory>
</VirtualHost>
```

6) Entra en “main-page.local” y prueba el login (email: demo@demo.com, pass: demo123).
7) Inicializar la API en un php server separado en consola, abriendo un cmd, moviéndote a la ruta de “php-puro-api”, y ejecutando “`php -S localhost:8001 -t public`” para servir el directorio /public.
8) Crear un virtual-host similar al anterior, pero para la integración de la API:

```
<VirtualHost *:80>
    ServerName api-integration.local
    DocumentRoot "C:/wamp64/www/integracion-api"
    <Directory "C:/wamp64/www/integracion-api">
        AllowOverride All
        Require all granted
    </Directory>
</VirtualHost>
```

8) prueba la integración de la API, accediendo a “api-integration.local” y logueandose con “demo@demo.com” y “demo123”.

3.B) Servidor embebido de PHP

1) Si configuras las rutas “MYSQL_DIR”, “FRONTEND_DIR”, “BACKEND_DIR” y “API_DIR” en “start-dev.bat”, este lanza todos los servicios necesarios automáticamente con WT (si existe) o con cmd individuales (si WT no existe). Tambien incluye la ejecución de un gestor visual de bbdd phpmyadmin web (si existe), por si lo necesitas.

2) Arranca MySQL service (o si usas él .bat, más adelante lo puedes lanzar en consola también).
3) Debes tener php en el path de Windows para este método, sino debes usar la ruta entera de php al ejecutar los comandos o añadirlo al path.
4) Abre un cmd, muévete a la ruta de la API “php-puro-api” y usa “`composer install`” para actualizar las dependencias de composer.
5) Abre un cmd y muevete a la ruta del proyecto de la API “php-puro-api”
6) ejecuta la migración de bd usando “`php scripts/migrate.php`”
7) Ejecuta el server php en la consola “`php -S localhost:8001 -t public`”, estando en la ruta de la API (API desplegada en http://localhost:8001)
8) Arranca el main-page/public en otro php server, abriendo otro cmd, moviéndote a la ruta de la main-page “main-page” y usando “`php -S localhost:8000 -t public`”
9) Prueba la main-page en http://localhost:8000/ y logueate con “demo@demo.com” y “demo123”.

10) Arranca el api-integration en otro php server en consola, abriendo un cmd nuevo, moviéndote a la ruta de la integración “integracion-api” y usando “php -S localhost:5500”.
11) Prueba la integración de la api en http://localhost:5500/ y logueate con “demo@demo.com” y “demo123”.

4) Información adicional (opcional / fuera del flujo principal)

1) Base de datos: creación, migraciones y seeds

1.1) Migraciones (001 y 002):

001_create_users.sql

```
CREATE TABLE IF NOT EXISTS users (
  id INT AUTO_INCREMENT PRIMARY KEY,
  email VARCHAR(255) NOT NULL UNIQUE,
  password_hash VARCHAR(255) NOT NULL,
  first_name VARCHAR(100) NOT NULL,
  last_name VARCHAR(100) NOT NULL,
  role ENUM('admin','user') NOT NULL DEFAULT 'user',
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
-- Seed demo admin (password: demo123)
INSERT INTO users (email, password_hash, first_name, last_name, role)
VALUES (
  'demo@demo.com',
  '$2y$10$g4uD7iM3H0s0L1qf8a7I7u8y5q4F8QfHs8m0mQ4rM3G7x2cVdWq9e',
  'Demo', 'Admin', 'admin'
) ON DUPLICATE KEY UPDATE email = email;
```

002_create_tasks.sql

```
CREATE TABLE IF NOT EXISTS tasks (
  id INT AUTO_INCREMENT PRIMARY KEY,
  user_id INT NOT NULL,
  title VARCHAR(255) NOT NULL,
  description TEXT NULL,
  starts_at DATETIME NULL,
  ends_at DATETIME NULL,
  completed TINYINT(1) NOT NULL DEFAULT 0,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
  INDEX idx_user_id (user_id),
  CONSTRAINT fk_tasks_user FOREIGN KEY (user_id) REFERENCES users(id)
ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

2) Probar endpoints (Auth, Tasks, Users)

2.1) Login:

```
POST /api/v1/auth/login
Content-Type: application/json

{
  "email": "demo@demo.com",
  "password": "demo123"
}
```

Respuesta 200:

JSON

```
{
  "token": "<JWT>",
  "user": {
    "role": "admin" | "user",
    "is_admin": true | false
  }
}
```

2.2) Tasks (Bearer requerido):

GET /api/v1/tasks?per_page=20&page=1

```
Authorization: Bearer <JWT>
```

- Los **usuarios normales** solo verán sus propias tareas.
- Los **admins** pueden añadir:
  - ?all=1 → ver todas las tareas.
  - ?user_id=<id> → filtrar por un usuario específico.

POST /api/v1/tasks

```
Authorization: Bearer <JWT>
Content-Type: application/json
{
  "title": "Reunión",
  "description": "con equipo",
  "starts_at": "2025-11-08 10:00:00",
  "ends_at": "2025-11-08 11:00:00",
  "completed": 0
}
```

Los **admins** pueden además crear tareas para otros usuarios incluyendo el campo user_id

```
{
  "title": "Planificación semanal",
  "user_id": 12
}
```

PUT /api/v1/tasks/:id

```
Authorization: Bearer <JWT>
Content-Type: application/json
{
  "title": "Nueva descripción",
```

```
  "completed": 1
}
```

- Usuarios normales pueden editar solo sus propias tareas.
- Admins pueden editar cualquier tarea y reasignarla con user_id.

```
{
  "user_id": 7,
  "completed": 1
}
```

DELETE /api/v1/tasks/:id

```
Authorization: Bearer <JWT>
```

- Usuarios normales solo pueden eliminar sus tareas.
- Admins pueden eliminar cualquier tarea por ID.

2.3) Users (solo ADMIN):

GET /api/v1/users?per_page=20&page=1

```
Authorization: Bearer <JWT-ADMIN>
```

Lista todos los usuarios registrados (paginado con ?page y ?per page).

POST /api/v1/users

```
Authorization: Bearer <JWT-ADMIN>
Content-Type: application/json
{
  "email": "nuevo@demo.com",
  "first_name": "Nuevo",
  "last_name": "Usuario",
  "role": "user",
  "password": "Secreta123"
}
```

PUT /api/v1/users/:id

```
Authorization: Bearer <JWT-ADMIN>
Content-Type: application/json
{
  "first_name": "Nombre",
  "last_name": "Actualizado",
  "role": "admin"
}
```

Puedes incluir password si deseas cambiarla.

DELETE /api/v1/users/:id

```
Authorization: Bearer <JWT-ADMIN>
```

Elimina un usuario por ID (solo accesible para administradores).

Todos los endpoints están documentados en el apartado de
“Documentacion” de la main-page (en el navbar está el acceso directo).

3) Ejecutar tests unitarios API

- Usando composer: ejecutamos en la ruta del api, el comando "`composer test`"

```
C:\xampp\htdocs\0PLANTILLAS\php-puro-api>composer test
PHPUnit 10.5.58 by Sebastian Bergmann and contributors.

Runtime:       PHP 8.2.12
Configuration: C:\xampp\htdocs\0PLANTILLAS\php-puro-api\tests\phpunit.xml

.                                                                   1 / 1 (100%)

Time: 00:00.017, Memory: 8.00 MB

OK (1 test, 2 assertions)
```

- Usando phpUnit directamente: ejecutamos el archivo de phpunit en la ruta asi "`.\vendor\bin\phpunit.bat -c tests\phpunit.xml`"

```
C:\xampp\htdocs\0PLANTILLAS\php-puro-api>.\vendor\bin\phpunit.bat -c tests\phpunit.xml
PHPUnit 10.5.58 by Sebastian Bergmann and contributors.

Runtime:       PHP 8.2.12
Configuration: C:\xampp\htdocs\0PLANTILLAS\php-puro-api\tests\phpunit.xml

.                                                                   1 / 1 (100%)

Time: 00:00.018, Memory: 8.00 MB

OK (1 test, 2 assertions)
```

4) Solución de problemas (FAQ)

- 401 tras cambiar de DB: borra el token y vuelve a login; revisa JWT-SECRET.
- .env no se lee: confirma ruta en Bootstrap::loadEnv, permisos y reinicia; imprime jwtSecret().
- SQL: asegurate de aplicar migraciones en la DB configurada.
- CORS/Preflight: el middleware permite Authorization y Content-Type.

Esta guía cubre instalación desde cero para API, main-page y mini integración API. Incluye seeds (usuario demo admin) y pruebas básicas con JWT.