

MANUAL TÉCNICO Y DE INSTALACIÓN



Por: Ing. Ever Corazón

Versión V1.0.0

Índice

| | |
|---|----|
| 1. INTRODUCCIÓN | 3 |
| 1.1. Propósito del Documento | 3 |
| 1.2. Alcance..... | 3 |
| 2. ARQUITECTURA DEL SISTEMA | 3 |
| 2.1. Diagrama de Componentes | 3 |
| 2.2. Stack Tecnológico | 4 |
| 2.3. Requisitos del Sistema | 4 |
| 2.3.1. Servidor de Producción..... | 4 |
| 2.3.2. Estación de Desarrollo..... | 4 |
| 3. INSTALACIÓN DEL SISTEMA..... | 5 |
| 3.1 Descargar XAMPP | 5 |
| 3.2 Instalación del código desde GitHub | 7 |
| 3.3 Creación de la base de datos | 11 |
| 3.4 Inicialización de los servidores | 13 |
| ANEXOS..... | 15 |
| Script de la base de datos inicial | 15 |



1. INTRODUCCIÓN

1.1. Propósito del Documento

Este manual técnico proporciona instrucciones detalladas para la instalación, configuración, despliegue y mantenimiento del Sistema Inteligente de Gestión Ganadera. Está dirigido a administradores de sistemas, técnicos de TI y personal especializado responsable de la implementación de la solución.

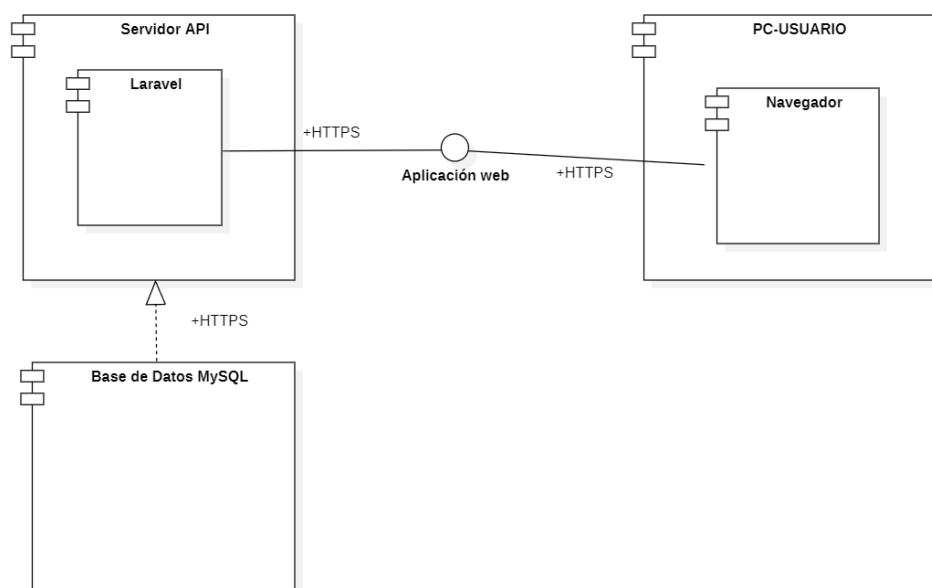
1.2. Alcance

El documento cubre todos los aspectos técnicos necesarios para:

- Despliegue en ambiente local y de producción
- Configuración de infraestructura de red
- Implementación de medidas de seguridad
- Resolución de problemas técnicos comunes

2. ARQUITECTURA DEL SISTEMA

2.1. Diagrama de Componentes



2.2. Stack Tecnológico


- **Frontend:** Laravel Blade, HTML5, CSS3, JavaScript
- **Backend:** Laravel 8.x, PHP 8.1+
- **Base de Datos:** MySQL 8.0+
- **Servidor Web:** Nginx 1.18+
- **Cache:** Redis 6.x
- **Sistema Operativo:** Ubuntu 22.04 LTS o Windows 10/11

2.3. Requisitos del Sistema

2.3.1. Servidor de Producción

- **CPU:** 2 vCPUs mínimo (4 recomendado)
- **RAM:** 4 GB mínimo (8 GB recomendado)
- **Almacenamiento:** 80 GB SSD
- **Sistema Operativo:** Ubuntu 22.04 LTS
- **Conexión de Red:** 100 Mbps dedicado

2.3.2. Estación de Desarrollo

- **CPU:** Intel Core i5 10ma generación o superior
 - **RAM:** 16 GB
 - **Almacenamiento:** 500 GB SSD
 - **Sistema Operativo:** Windows 10/11, macOS, o Linux
- 

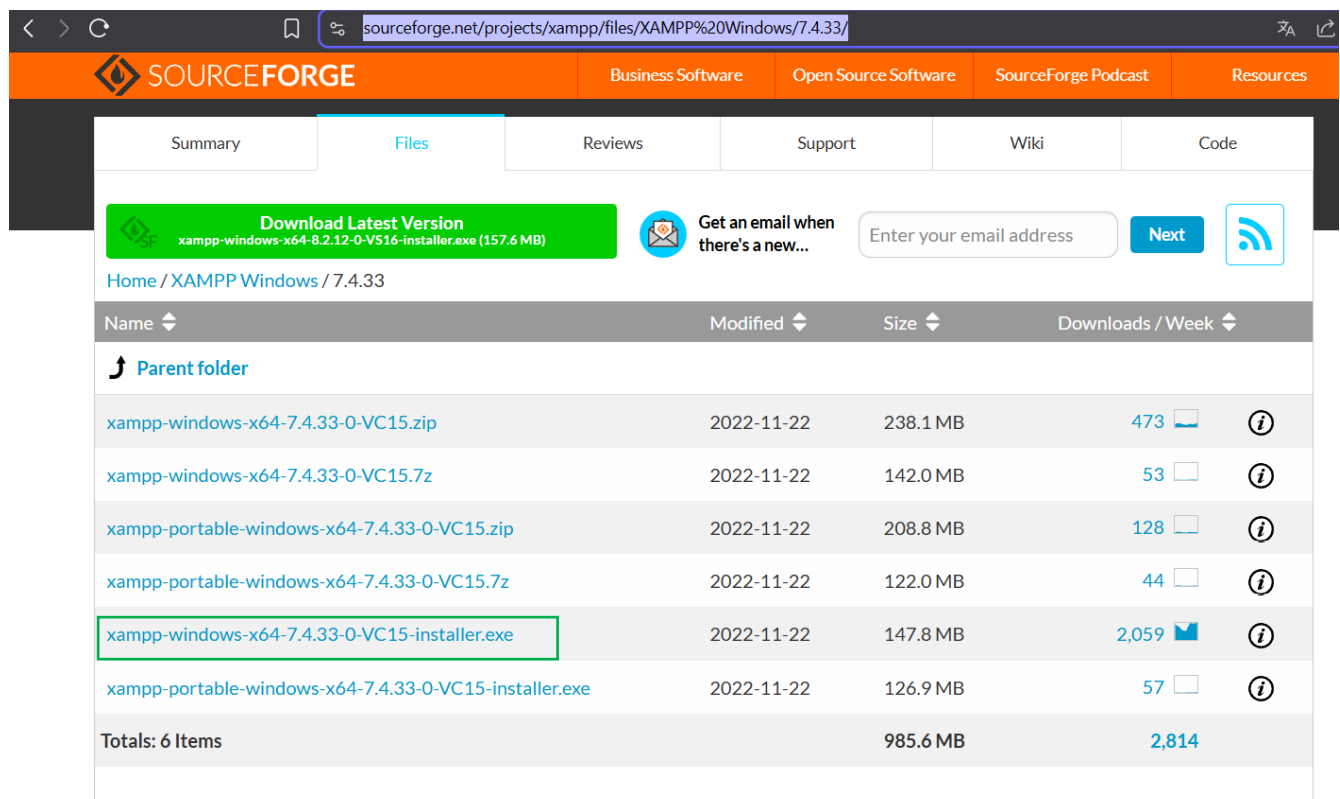
3. INSTALACIÓN DEL SISTEMA

3.1 Descargar XAMPP

XAMPP sirve para crear un entorno de desarrollo de servidores web locales en un ordenador personal, lo que permite crear, probar y desplegar sitios web y aplicaciones de manera privada antes de publicarlos en internet.

Paso 1: Dirigirse al enlace:

<https://sourceforge.net/projects/xampp/files/XAMPP%20Windows/7.4.33/>



Download Latest Version
xampp-windows-x64-8.2.12-0-VS16-installer.exe (157.6 MB)

Get an email when there's a new...
Enter your email address **Next**

Home / XAMPP Windows / 7.4.33

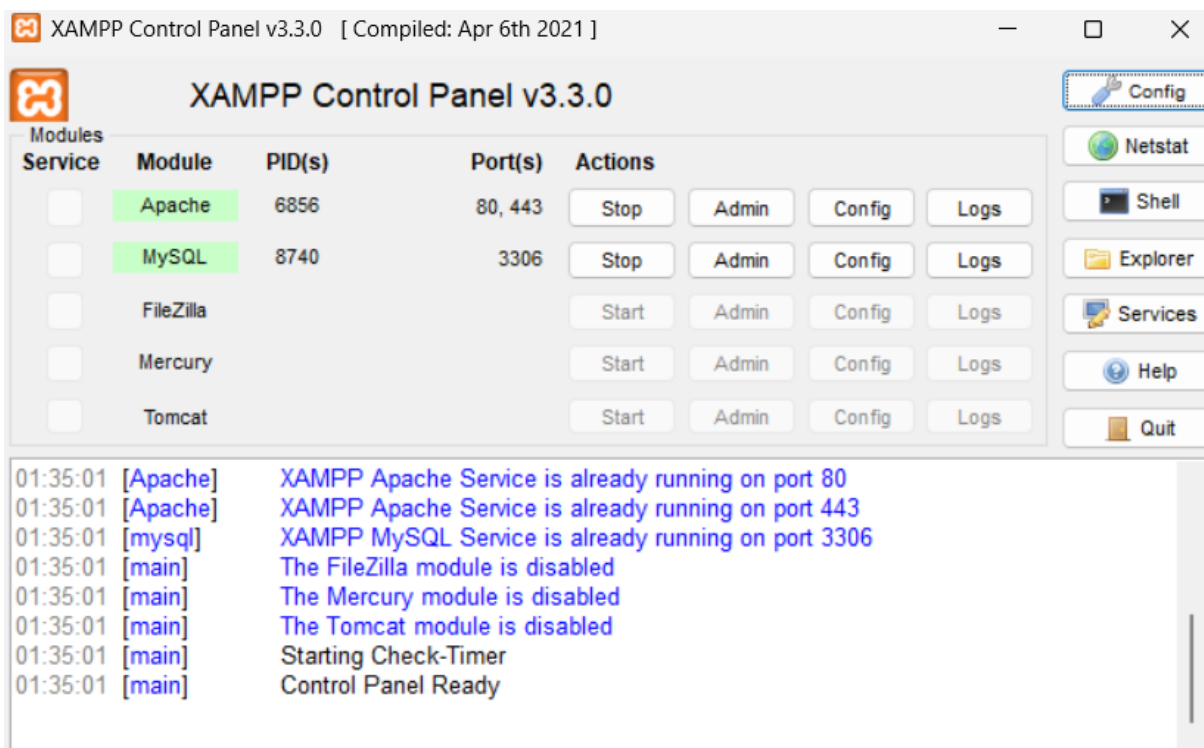
| Name | Modified | Size | Downloads / Week |
|--|------------|-----------------|------------------|
| Parent folder | | | |
| xampp-windows-x64-7.4.33-0-VC15.zip | 2022-11-22 | 238.1 MB | 473 |
| xampp-windows-x64-7.4.33-0-VC15.7z | 2022-11-22 | 142.0 MB | 53 |
| xampp-portable-windows-x64-7.4.33-0-VC15.zip | 2022-11-22 | 208.8 MB | 128 |
| xampp-portable-windows-x64-7.4.33-0-VC15.7z | 2022-11-22 | 122.0 MB | 44 |
| xampp-windows-x64-7.4.33-0-VC15-installer.exe | 2022-11-22 | 147.8 MB | 2,059 |
| xampp-portable-windows-x64-7.4.33-0-VC15-installer.exe | 2022-11-22 | 126.9 MB | 57 |
| Totals: 6 Items | | 985.6 MB | 2,814 |

Seleccionar el enmarcado, esta instalación incluye también PHP, por lo que será aún más sencillo de preparar todo.

Paso 2: Seguir las instrucciones de instalación, se recomienda instalarlo en la ruta C:\xampp

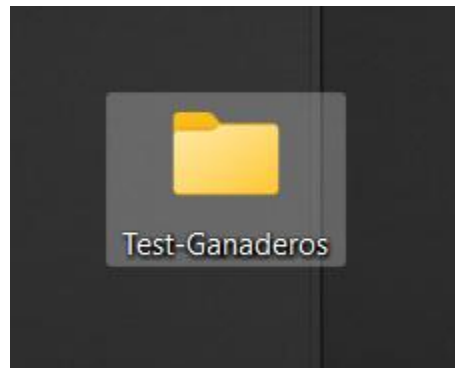


Paso 3: Ejecutar XAMPP Control Panel y ejecutar los siguientes servicios:



3.2 Instalación del código desde GitHub

Paso 1: Crear una Carpeta para alojar los códigos, ejemplo:



Paso 2: Abrir una terminal del símbolo del sistema (CMD) desde la carpeta que acabamos de crear:

```
C:\Windows\System32\cmd.e X + v
Microsoft Windows [Versión 10.0.26100.7171]
(c) Microsoft Corporation. Todos los derechos reservados.
C:\Users\Ever Corazón\Desktop\Test-Ganaderos>|
```

Paso 3: Ejecutar el siguiente comando (para el backend):

git clone <https://github.com/EverCR1/Final-Backend.git>

```
C:\Windows\System32\cmd.e X + v
Microsoft Windows [Versión 10.0.26100.7171]
(c) Microsoft Corporation. Todos los derechos reservados.
C:\Users\Ever Corazón\Desktop\Test-Ganaderos>git clone https://github.com/EverCR1/Final-Backend.git
Cloning into 'Final-Backend'...
remote: Enumerating objects: 230, done.
remote: Counting objects: 100% (230/230), done.
remote: Compressing objects: 100% (150/150), done.
remote: Total 230 (delta 79), reused 209 (delta 60), pack-reused 0 (from 0)
Receiving objects: 100% (230/230), 101.43 KiB | 607.00 KiB/s, done.
Resolving deltas: 100% (79/79), done.
C:\Users\Ever Corazón\Desktop\Test-Ganaderos>|
```

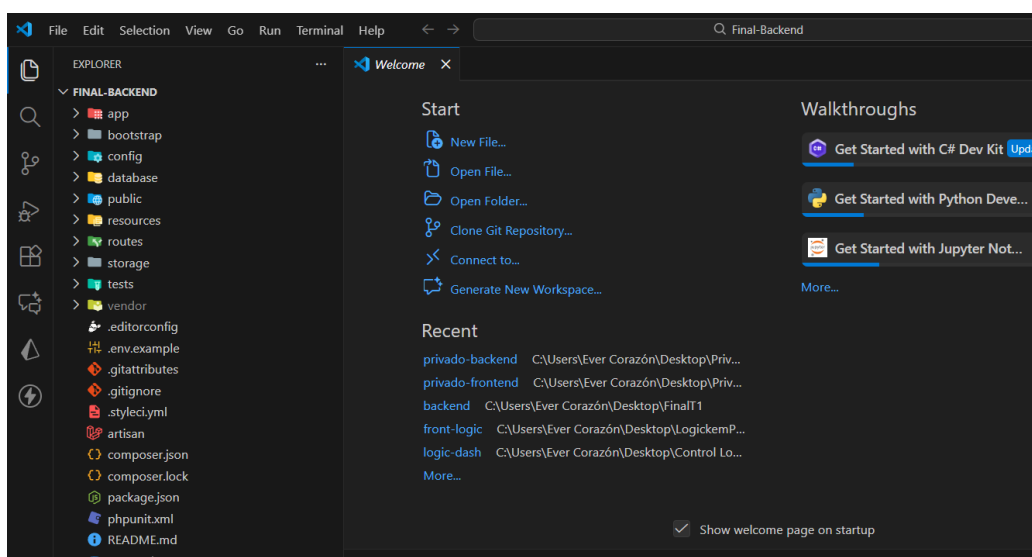
Paso 4: Ejecutar el siguiente comando (para el frontend):

git clone <https://github.com/EverCR1/Final-Frontend.git>

```
C:\Users\Ever Corazón\Desktop\Test-Ganaderos>git clone https://github.com/EverCR1/Final-Frontend.git
Cloning into 'Final-Frontend'...
remote: Enumerating objects: 330, done.
remote: Counting objects: 100% (330/330), done.
remote: Compressing objects: 100% (174/174), done.
remote: Total 330 (delta 157), reused 294 (delta 125), pack-reused 0 (from 0)
Receiving objects: 100% (330/330), 592.38 KiB | 1.48 MiB/s, done.
Resolving deltas: 100% (157/157), done.

C:\Users\Ever Corazón\Desktop\Test-Ganaderos>|
```

Paso 5: Abrir ambos proyectos con un editor de code, puede ser Visual Studio Code



Paso 6: Ejecutar el siguiente código en ambos proyectos:

Composer install #Sirve para instalar las librerías y dependencias

```
PS C:\Users\Ever Corazón\Desktop\Test-Ganaderos\Final-Backend> composer install
Installing dependencies from lock file (including require-dev)
Verifying lock file contents can be installed on current platform.
Package operations: 106 installs, 0 updates, 0 removals
- Installing doctrine/inflector (2.1.0): Extracting archive
- Installing doctrine/lexer (1.2.3): Extracting archive
- Installing dragonmantank/cron-expression (v3.5.0): Extracting archive
- Installing symfony/polyfill-php80 (v1.33.0): Extracting archive
- Installing symfony/polyfill-iconv (v1.33.0): Extracting archive
- Installing symfony/polyfill-mbstring (v1.33.0): Extracting archive
- Installing symfony/var-dumper (v5.4.48): Extracting archive
```

Paso 7: Dirigirse en un navegador a:



<http://127.0.0.1/dashboard/>


Luego presionamos PHPMyAdmin y seleccionamos Nueva (Para crear una base de datos):



Le ponemos un nombre:

Bases de datos

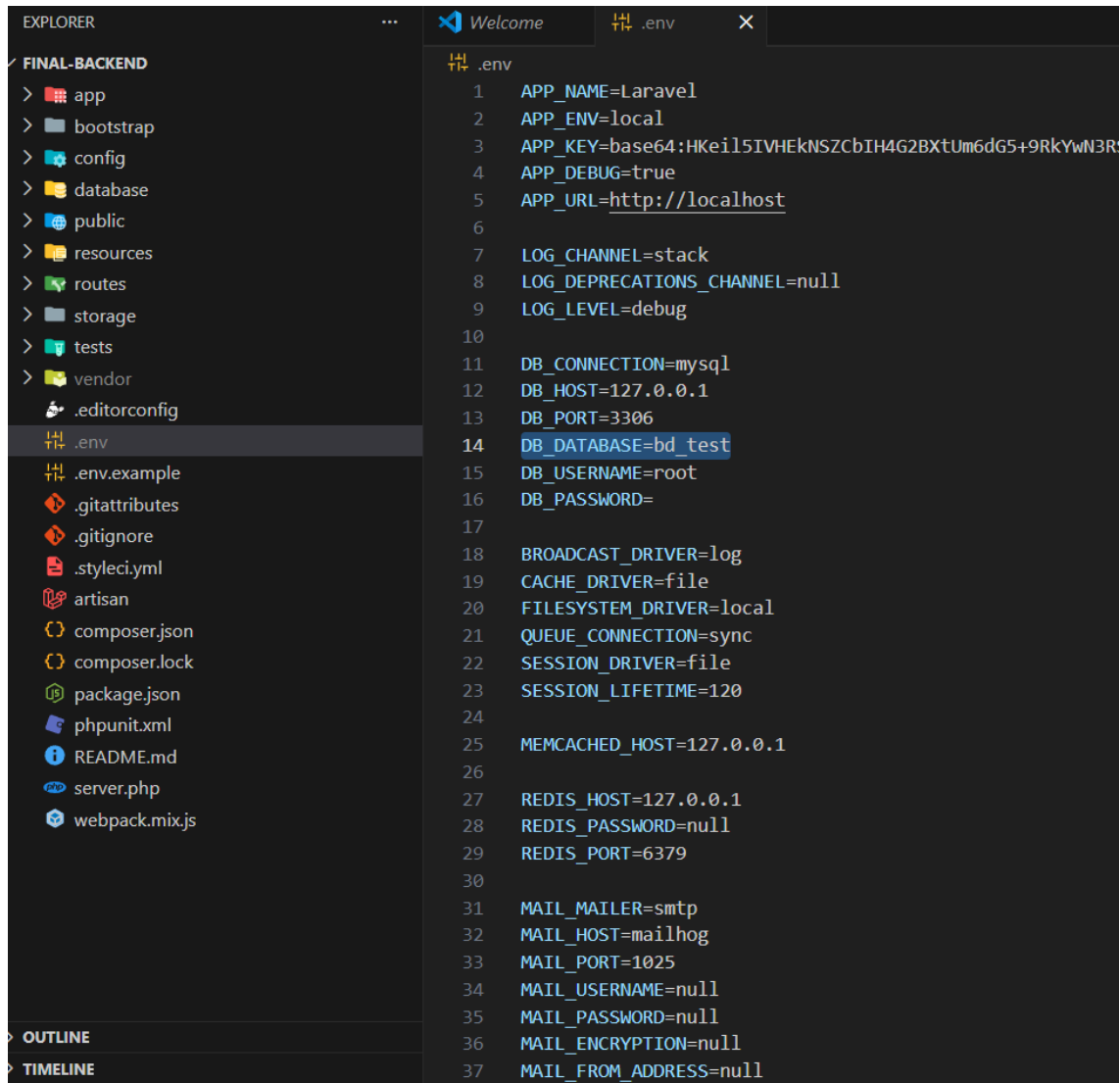
 **Crear base de datos** 



Crear



Paso 8: En la raíz del proyecto backend crear un archivo llamado .env y copiamos el siguiente código, asegurándonos que pongamos el nombre de nuestra base de datos:



The screenshot shows a code editor with a sidebar on the left and a main editor area on the right. The sidebar, labeled 'EXPLORER', shows a file tree for a project named 'FINAL-BACKEND'. The tree includes folders like 'app', 'bootstrap', 'config', 'database', 'public', 'resources', 'routes', 'storage', 'tests', and 'vendor', along with various configuration files like '.editorconfig', '.env', '.env.example', '.gitattributes', '.gitignore', '.styleci.yml', 'artisan', 'composer.json', 'composer.lock', 'package.json', 'phpunit.xml', 'README.md', 'server.php', and 'webpack.mix.js'. The '.env' file is selected and highlighted. The main editor area shows the content of the '.env' file, which consists of 37 lines of configuration. The line 'DB_DATABASE=bd_test' is highlighted in blue. The configuration includes settings for the application name, environment, key, debug mode, URL, logging, database connection, cache, filesystem, queue, session, memcached, redis, and mail.

```
1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=base64:Hkeil5IVHEkNSZCbIH4G2BXtUm6dG5+9RkYwN3R
4 APP_DEBUG=true
5 APP_URL=http://localhost
6
7 LOG_CHANNEL=stack
8 LOG_DEPRECATIONS_CHANNEL=null
9 LOG_LEVEL=debug
10
11 DB_CONNECTION=mysql
12 DB_HOST=127.0.0.1
13 DB_PORT=3306
14 DB_DATABASE=bd_test
15 DB_USERNAME=root
16 DB_PASSWORD=
17
18 BROADCAST_DRIVER=log
19 CACHE_DRIVER=file
20 FILESYSTEM_DRIVER=local
21 QUEUE_CONNECTION=sync
22 SESSION_DRIVER=file
23 SESSION_LIFETIME=120
24
25 MEMCACHED_HOST=127.0.0.1
26
27 REDIS_HOST=127.0.0.1
28 REDIS_PASSWORD=null
29 REDIS_PORT=6379
30
31 MAIL_MAILER=smtp
32 MAIL_HOST=mailhog
33 MAIL_PORT=1025
34 MAIL_USERNAME=null
35 MAIL_PASSWORD=null
36 MAIL_ENCRYPTION=null
37 MAIL_FROM_ADDRESS=null
```

Paso 9: En la raíz del proyecto frontend también debemos crear un archivo llamado .env y copiamos el siguiente código, solo que este no se conectará a base de datos:

```

1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=base64:saAu9kjWAFtmAVzYCCACXdCS0WjvR4S3U0fsNsv2U1o=
4 APP_DEBUG=true
5 APP_URL=http://localhost//8001
6 API_URL=http://localhost:8000/api
7 API_BASE_URL=http://localhost:8000
8
9 LOG_CHANNEL=stack
10 LOG_DEPRECATIONS_CHANNEL=null
11 LOG_LEVEL=debug
12
13 DB_CONNECTION=sqlite
14 #DB_HOST=127.0.0.1
15 #DB_PORT=3306
16 #DB_DATABASE=
17 #DB_USERNAME=
18 #DB_PASSWORD=
19
20 BROADCAST_DRIVER=log
21 CACHE_DRIVER=file
22 FILESYSTEM_DRIVER=local
23 QUEUE_CONNECTION=sync
24 SESSION_DRIVER=file
  
```

3.3 Creación de la base de datos

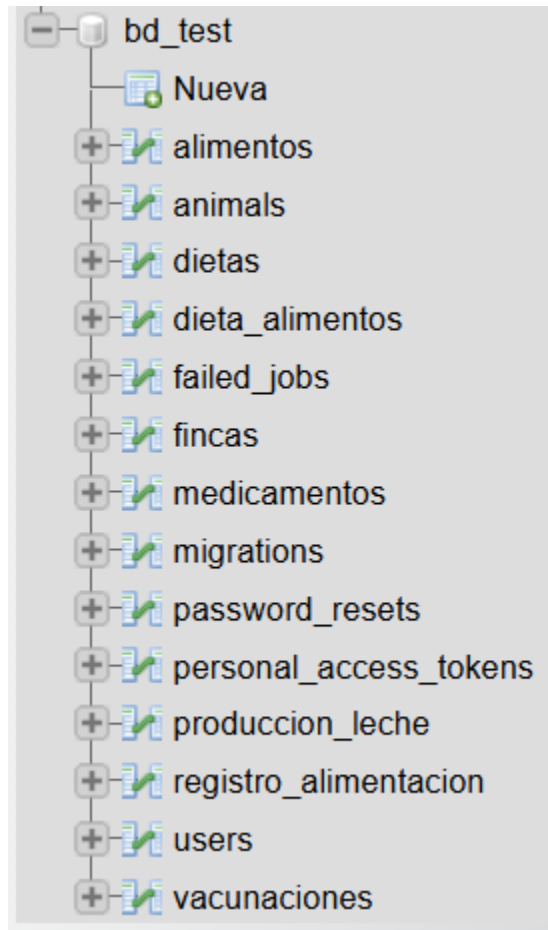
Paso 1: Ejecutamos el siguiente código en el backend:

php artisan migrate

```

PS C:\Users\Ever Corazón\Desktop\Test-Ganaderos\Final-Backend> php artisan migrate
Migration table created successfully.
Migrating: 2014_10_12_000000_create_users_table
Migrated: 2014_10_12_000000_create_users_table (15.87ms)
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated: 2014_10_12_100000_create_password_resets_table (20.47ms)
Migrating: 2019_08_19_000000_create_failed_jobs_table
Migrated: 2019_08_19_000000_create_failed_jobs_table (18.07ms)
  
```

Ahora ya tendremos nuestras tablas creadas:



Paso 2: Ejecutamos los siguientes comandos para llenar las tablas principales de la base de datos:

1. `php artisan db:seed --class=SensorDataSeeder UserSeeder`
2. `php artisan db:seed --class=SensorDataSeeder FincaSeeder`
3. `php artisan db:seed --class=SensorDataSeeder AnimalSeeder`
4. `php artisan db:seed --class=SensorDataSeeder MedicamentoSeeder`
5. `php artisan db:seed --class=SensorDataSeeder VacunacionSeeder`
6. `php artisan db:seed --class=SensorDataSeeder ProduccionLecheSeeder`

```

PS C:\Users\Ever Corazón\Desktop\Test-Ganaderos\Final-Backend> 1. php artisan db:seed --class=SensorDataSeeder UserSeeder

PS C:\Users\Ever Corazón\Desktop\Test-Ganaderos\Final-Backend> php artisan db:seed --class=SensorDataSeeder UserSeeder

Database seeding completed successfully.
Database seeding completed successfully.
PS C:\Users\Ever Corazón\Desktop\Test-Ganaderos\Final-Backend> php artisan db:seed --class=SensorDataSeeder MedicamentoSeeder
Database seeding completed successfully.
PS C:\Users\Ever Corazón\Desktop\Test-Ganaderos\Final-Backend> php artisan db:seed --class=SensorDataSeeder VacunacionSeeder
Database seeding completed successfully.
PS C:\Users\Ever Corazón\Desktop\Test-Ganaderos\Final-Backend> php artisan db:seed --class=SensorDataSeeder ProduccionLecheSeeder
Database seeding completed successfully.

```

3.4 Inicialización de los servidores

Paso 1: Ahora ejecutamos el siguiente comando en el backend:

php artisan serve

```

PS C:\Users\Ever Corazón\Desktop\Test-Ganaderos\Final-Backend> php artisan serve
Starting Laravel development server: http://127.0.0.1:8000
[Thu Nov 20 01:59:28 2025] PHP 7.4.33 Development Server (http://127.0.0.1:8000) started

```

Paso 2: Ejecutamos el siguiente comando en el frontend:

php artisan serve --port 8001

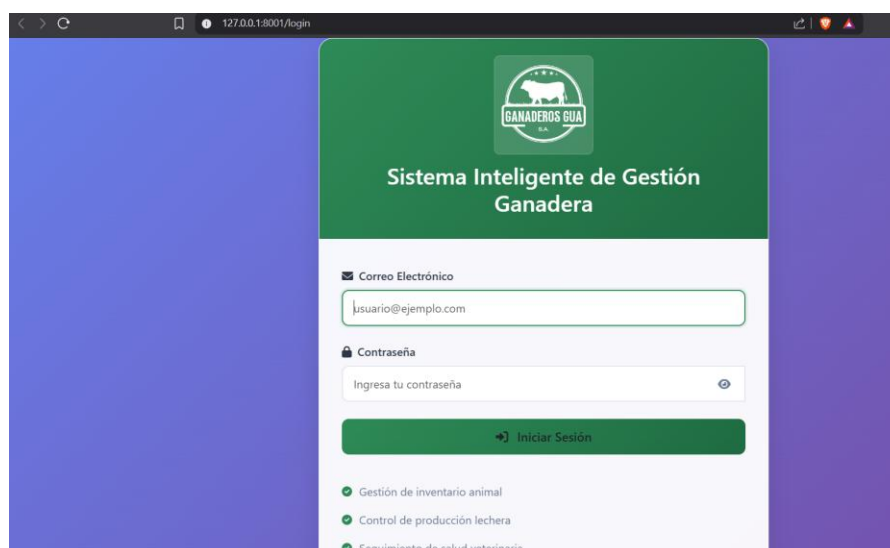
```

PS C:\Users\Ever Corazón\Desktop\Test-Ganaderos\Final-Frontend> php artisan serve --port 8001
Starting Laravel development server: http://127.0.0.1:8001
[Thu Nov 20 02:01:07 2025] PHP 7.4.33 Development Server (http://127.0.0.1:8001) started

```

Paso 3: Ingresamos a la URL

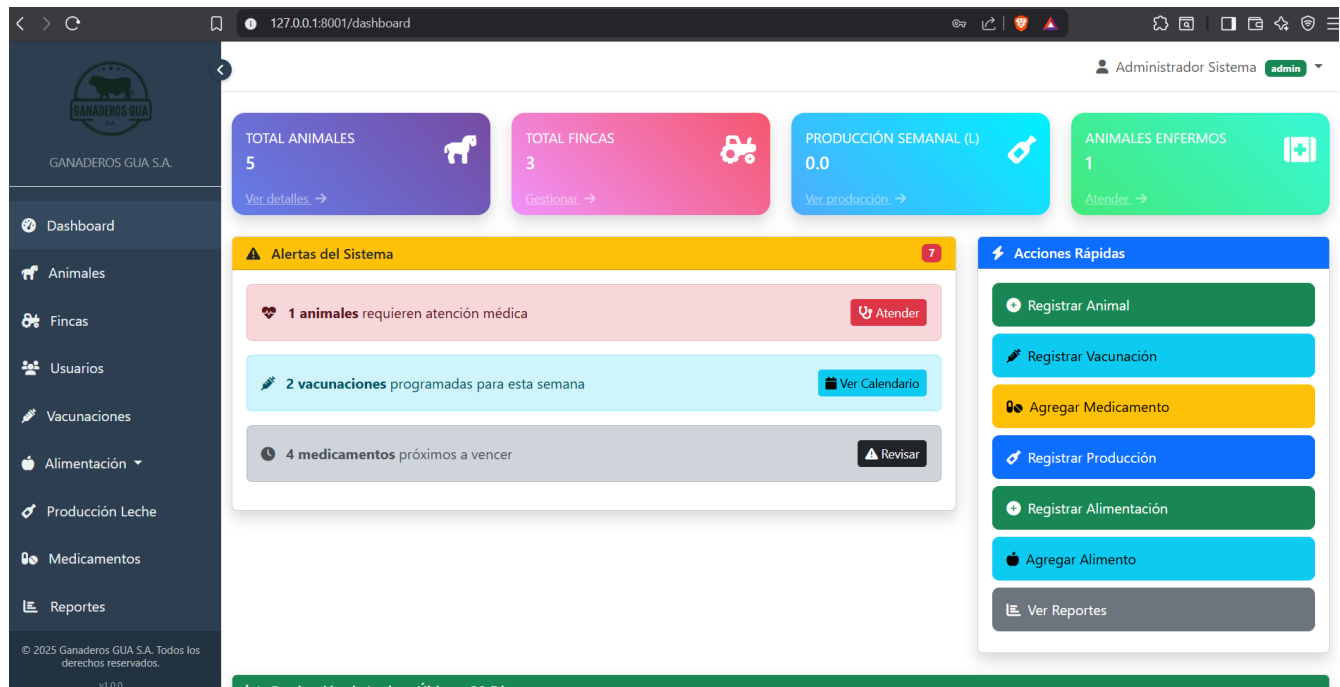
<http://127.0.0.1:8001>



Iniciamos sesión con los siguientes datos:

Correo: admin@cunor.edu.gt

Contraseña: password123



¡LISTO! YA PODEMOS ACCEDER AL SISTEMA Y MODIFICAR LO QUE NECESITEMOS

ANEXOS

Script de la base de datos inicial

Solo algunas tablas, el script completo se encuentra en el código GitHub

```
SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
START TRANSACTION;
SET time_zone = "+00:00";

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;

--
-- Base de datos: `bd_privado`
--

--
-- Estructura de tabla para la tabla `alimentos`
--

CREATE TABLE `alimentos` (
  `id` bigint(20) UNSIGNED NOT NULL,
  `finca_id` bigint(20) UNSIGNED NOT NULL,
  `nombre` varchar(255) NOT NULL,
  `tipo` enum('concentrado','forraje','suplemento','mineral','otro') NOT NULL,
  `descripcion` text DEFAULT NULL,
```

```

`unidad_medida` varchar(255) NOT NULL,
`stock_actual` int(11) NOT NULL,
`stock_minimo` int(11) NOT NULL,
`precio_unitario` decimal(10,2) DEFAULT NULL,
`fecha_vencimiento` date DEFAULT NULL,
`proveedor` varchar(255) DEFAULT NULL,
`created_at` timestamp NULL DEFAULT NULL,
`updated_at` timestamp NULL DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

```

```
-----
```

```
--
```

```
-- Estructura de tabla para la tabla `animals`
```

```
--
```

```

CREATE TABLE `animals` (
  `id` bigint(20) UNSIGNED NOT NULL,
  `finca_id` bigint(20) UNSIGNED NOT NULL,
  `identificacion` varchar(255) NOT NULL,
  `nombre` varchar(255) DEFAULT NULL,
  `especie` enum('bovino','porcino','caprino','ovina') NOT NULL,
  `raza` enum('holstein','brahman','angus','criolla','otra') NOT NULL,
  `fecha_nacimiento` date NOT NULL,
  `sexo` enum('macho','hembra') NOT NULL,
  `estado` enum('activo','vendido','muerto','enfermo') NOT NULL DEFAULT 'activo',
  `peso_inicial` decimal(8,2) DEFAULT NULL,
  `peso_actual` decimal(8,2) DEFAULT NULL,

```

```
`observaciones` text DEFAULT NULL,
`created_at` timestamp NULL DEFAULT NULL,
`updated_at` timestamp NULL DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

```
-----
```

```
--
```

```
-- Estructura de tabla para la tabla `dietas`
```

```
--
```

```
CREATE TABLE `dietas` (
  `id` bigint(20) UNSIGNED NOT NULL,
  `finca_id` bigint(20) UNSIGNED NOT NULL,
  `nombre` varchar(255) NOT NULL,
  `descripcion` text DEFAULT NULL,
  `tipo_animal` enum('bovino','porcino','caprino','ovina') NOT NULL,
  `categoria` enum('ternero','desarrollo','adulto','lactancia','gestacion') NOT NULL,
  `costo_estimado_kg` decimal(8,2) NOT NULL DEFAULT 0.00,
  `activa` tinyint(1) NOT NULL DEFAULT 1,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

```
-----
```

```
--
```

```
-- Estructura de tabla para la tabla `dieta_alimentos`
```

--

```
CREATE TABLE `dieta_alimentos` (
  `id` bigint(20) UNSIGNED NOT NULL,
  `dieta_id` bigint(20) UNSIGNED NOT NULL,
  `alimento_id` bigint(20) UNSIGNED NOT NULL,
  `cantidad` decimal(8,2) NOT NULL,
  `frecuencia` varchar(255) NOT NULL,
  `observaciones` text DEFAULT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

--

-- Estructura de tabla para la tabla `failed_jobs`

--

```
CREATE TABLE `failed_jobs` (
  `id` bigint(20) UNSIGNED NOT NULL,
  `uuid` varchar(255) NOT NULL,
  `connection` text NOT NULL,
  `queue` text NOT NULL,
  `payload` longtext NOT NULL,
  `exception` longtext NOT NULL,
  `failed_at` timestamp NOT NULL DEFAULT current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

--

-- Estructura de tabla para la tabla `fincas`

--

```
CREATE TABLE `fincas` (
  `id` bigint(20) UNSIGNED NOT NULL,
  `nombre` varchar(255) NOT NULL,
  `ubicacion` varchar(255) NOT NULL,
  `telefono` varchar(255) DEFAULT NULL,
  `responsable` varchar(255) NOT NULL,
  `zona` enum('norte','sur','este') NOT NULL,
  `ip_subred` varchar(255) DEFAULT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

--

-- Estructura de tabla para la tabla `medicamentos`

--

```
CREATE TABLE `medicamentos` (
  `id` bigint(20) UNSIGNED NOT NULL,
  `finca_id` bigint(20) UNSIGNED NOT NULL,
```

```
`nombre` varchar(255) NOT NULL,  
`tipo` varchar(255) NOT NULL,  
`descripcion` text DEFAULT NULL,  
`stock_actual` int(11) NOT NULL,  
`stock_minimo` int(11) NOT NULL,  
`precio_unitario` decimal(8,2) DEFAULT NULL,  
`fecha_vencimiento` date DEFAULT NULL,  
`proveedor` varchar(255) DEFAULT NULL,  
`created_at` timestamp NULL DEFAULT NULL,  
`updated_at` timestamp NULL DEFAULT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

--

-- Estructura de tabla para la tabla `migrations`

--

```
CREATE TABLE `migrations` (  
  `id` int(10) UNSIGNED NOT NULL,  
  `migration` varchar(255) NOT NULL,  
  `batch` int(11) NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```