MANUAL TÉCNICO

SISTEMA DE ADMINISTRACION DE CARTERA

“SISDACA” v1.0

DIRECCION NACIONAL DE VIVIENDA SOCIAL RESIDUAL

Contenido

1. Introducción 3
2. Características 3
3. Requisitos del sistema 3
   3.1. HARDWARE 3
   3.2. software 3
   3.3. instalacion y Configuraciones recomendadas 4
4. seguridad 5
   4.1. Frameworks Utilizados 5
   4.2. Protección de Rutas 5
   4.3. Gestión de Sesión 5
   4.4. Autenticación de Dos Factores (2FA) 5
   4.5. Hashing de Contraseñas 5
   4.6. Protección contra CSRF (Cross-Site Request Forgery) 5
   4.7. Prevención de Inyección SQL 5
5. Mantenimiento y actualizaciones 5
6. Esquema estructural del sistema 5

1. INTRODUCCIÓN

SISDACA tiene como objetivo facilitar la administración de la cartera registrada en la dirección nacional de vivienda social residual, en el cual se cuenta con las opciones de administrar el perfil de beneficiario, obtener sus registros de pagos y planes de pago.

2. CARACTERÍSTICAS

SISDACA fue desarrollado empleando LARAVEL 11 como framework de PHP.

3. REQUISITOS DEL SISTEMA

3.1. HARDWARE

Los requisitos mínimos para el lanzamiento funcional (puesta a producción) del sistema son los siguientes:

- MEMORIA RAM: 4 GB
- PROCESADORES: 4 NUCLEOS/CORE CUALQUIER PROVEEDOR
- ALMACENAMIENTO: 64GB SSD (DISCO ESTADO SOLIDO)
- SISTEMA OPERATIVO: CUALQUIER DISTRIBUCION DE LINUX
- RED: ANCHO DE BANDA DE 30 MBPS MEDIANTE CABLE ETHERNET

Si los requisitos de hardware (dispositivos físicos) son satisfechos con los mínimos necesarios expresados en la lista previa, SISDACA podrá ejecutarse de manera óptima.

3.2. SOFTWARE

SISDACA cuenta con la siguiente lista de librerías y entornos necesarios para su despliegue:

- PHP 8.3^
  - **EXTENSIONES DE PHP NECESARIAS**
  - 10-pdo.ini
  - 15-xml.ini
  - 20-bcmath.ini
  - 20-calendar.ini
  - 20-ctype.ini
  - 20-curl.ini
  - 20-dom.ini
  - 20-exif.ini
  - 20-ffi.ini
  - 20-fileinfo.ini
  - 20-ftp.ini
  - 20-gd.ini
  - 20-gettext.ini
  - 20-iconv.ini
  - 20-intl.ini
  - 20-mbstring.ini

  - 20-pdo_pgsql.ini
  - 20-pgsql.ini
  - 20-phar.ini
  - 20-posix.ini
  - 20-readline.ini
  - 20-shmop.ini
  - 20-simplexml.ini
  - 20-sockets.ini
  - 20-sysvmsg.ini
  - 20-sysvsem.ini
  - 20-sysvshm.ini
  - 20-tokenizer.ini
  - 20-xmlreader.ini
  - 20-xmlwriter.ini
  - 20-xsl.ini
  - 20-zip.ini
- APACHE2
- SSH
- SUPERVISOR
- POSTGRESQL 17^
- COMPOSER
- NODEJS 22.10^

3.3. INSTALACION Y CONFIGURACIONES RECOMENDADAS

Una vez clonado el repositorio del sistema, se deberá configurar de la siguiente manera:

- Configurar archivo .ENV de acuerdo a las circunstancias
- Crear un link seguro a la carpeta de archivos: php artisan storage:link
- Ejecutar las migraciones para las tablas en la base de datos: php artisan migrate
- Compilar los recursos del proyecto: composer install | npm install |npm run build
- Opcionalmente, se ejecutar el siguiente comando para precargar los recursos limpiamente: php artisan optimize:clear | php artisan optimize

El sistema cuenta con microservicios y colas de procesos para evitar la saturación de recursos del servidor, estos mismos deben ser ejecutados con la siguiente instrucción:

$ php artisan queue:listen --timeout=300

Así mismo, se deberán ejecutar periódicamente los siguientes microservicios para actualizar los estados crediticios de los beneficiarios:

$ php artisan app:plan-status | php artisan app:beneficiary-status

4. SEGURIDAD

4.1. FRAMEWORKS UTILIZADOS

La autenticación se gestiona con Laravel Fortify, que provee la lógica de backend, y Laravel Jetstream, que implementa la interfaz de usuario (scaffolding) para el registro, inicio de sesión, etc., utilizando la pila Livewire.

4.2. PROTECCIÓN DE RUTAS

La mayoría de las rutas web críticas están agrupadas bajo el middleware auth:sanctum. Esto asegura que solo los usuarios autenticados puedan acceder a las funcionalidades principales de la aplicación, como la gestión de beneficiarios, proyectos y pagos.

4.3. GESTIÓN DE SESIÓN

Laravel Sanctum se utiliza junto con el guard web para la autenticación de peticiones AJAX (realizadas por Livewire), una gestión de sesión segura y stateful (basada en cookies) sin necesidad de tokens de API para la interfaz principal.

4.4. AUTENTICACIÓN DE DOS FACTORES (2FA)

La configuración de Fortify (config/fortify.php) muestra que la funcionalidad de 2FA está habilitada. Esto permite a los usuarios añadir una capa extra de seguridad a sus cuentas.

4.5. HASHING DE CONTRASEÑAS

Como es estándar en Laravel, las contraseñas se almacenan de forma segura mediante hashing con Bcrypt en el cual especifica que el coste del hashing (BCRYPT_ROUNDS) está configurado en 12, lo cual es un valor fuerte y recomendado.

4.6. PROTECCIÓN CONTRA CSRF (CROSS-SITE REQUEST FORGERY)

Todas las rutas definidas en el grupo web están protegidas automáticamente por el middleware VerifyCsrfToken.

4.7. PREVENCIÓN DE INYECCIÓN SQL

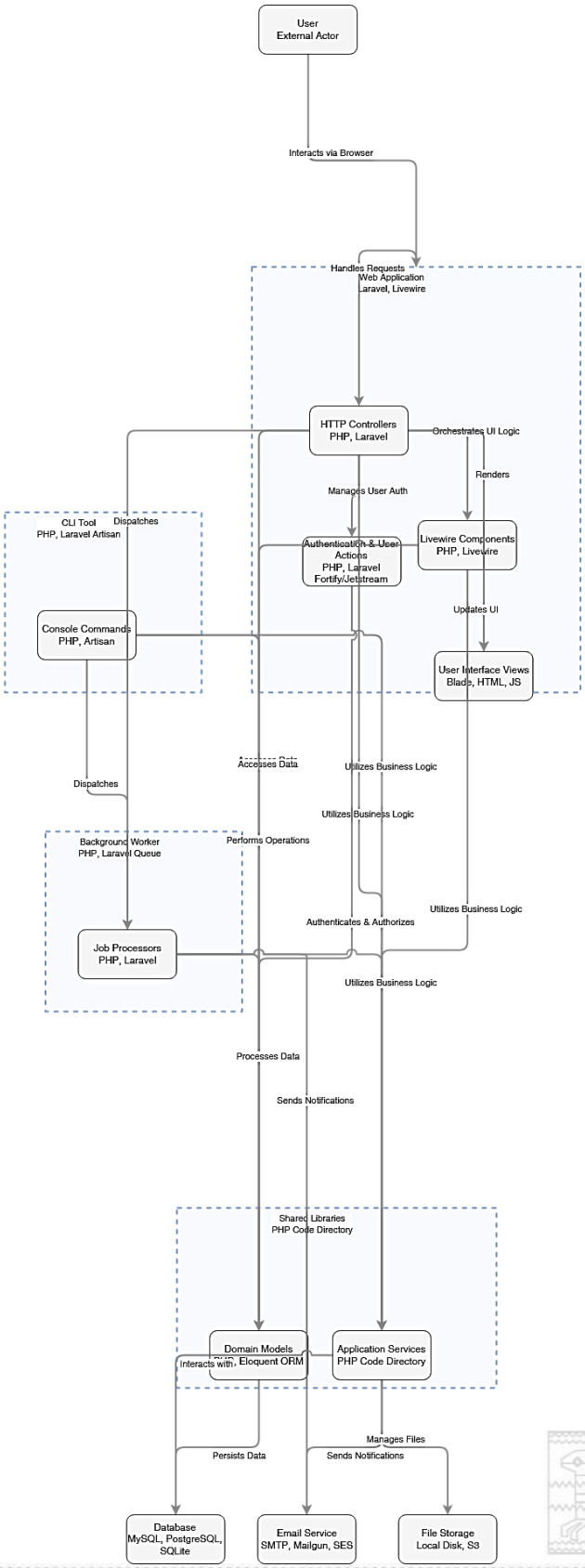
El uso del ORM Eloquent en todo el proyecto utiliza el enlace de parámetros (parameter binding) de forma nativa, lo que elimina eficazmente el riesgo de ataques de inyección SQL.

5. MANTENIMIENTO Y ACTUALIZACIONES

Las actualizaciones estarán completamente reflejadas en el repositorio del sistema, así mismo, cada sincronización de estos cambios requerirá la ejecución del siguiente comando para aplicar cambios:

$ composer update | npm install

6. ESQUEMA ESTRUCTURAL DEL SISTEMA


User
External Actor
Interacts via Browser
Handles Requests
Web Application
Laravel, Livewire
HTTP Controllers
PHP, Laravel
Orchestrates UI Logic
Renders
Manages User Auth
CLI Tool
PHP, Laravel Artisan
Dispatches
Authentication & User Actions
PHP, Laravel Fortify/Jetstream
Livewire Components
PHP, Livewire
Updates UI
Console Commands
PHP, Artisan
User Interface Views
Blade, HTML, JS
Accesses Data
Utilizes Business Logic
Dispatches
Utilizes Business Logic
Background Worker
PHP, Laravel Queue
Performs Operations
Utilizes Business Logic
Authenticates & Authorizes
Job Processors
PHP, Laravel
Utilizes Business Logic
Processes Data
Sends Notifications
Shared Libraries
PHP Code Directory
Domain Models
PHP, Eloquent ORM
Application Services
PHP Code Directory
Interacts with
Manages Files
Persists Data
Sends Notifications
Database
MySQL, PostgreSQL, SQLite
Email Service
SMTP, Mailgun, SES
File Storage
Local Disk, S3

7. GLOSARIO TÉCNICO

- Framework – Estructura base para desarrollo de software (ej. Laravel 11).
- Laravel Fortify – Paquete para autenticación (registro, inicio de sesión).
- Laravel Jetstream – Scaffolding para interfaces de usuario (Livewire/Inertia).
- Middleware – Filtro de peticiones HTTP (ej. auth:sanctum).
- Laravel Sanctum – Autenticación para APIs y sesiones basadas en cookies.
- 2FA (Autenticación de Dos Factores) – Doble verificación de identidad.
- Bcrypt – Algoritmo seguro de hashing para contraseñas.
- CSRF – Protección contra falsificación de solicitudes entre sitios.
- ORM (Eloquent) – Mapeo objeto-relacional para bases de datos.
- Microservicios – Componentes independientes para tareas específicas (ej. colas de procesos).
- Composer – Gestor de dependencias para PHP.
- Node.js – Entorno para ejecutar JavaScript en el servidor.
- PostgreSQL – Sistema de gestión de bases de datos relacional.
- Livewire – Framework para interfaces dinámicas con PHP.
- Storage Link – Enlace simbólico para archivos almacenados en Laravel.
- Scaffolding – Estructura base generada automáticamente para acelerar el desarrollo (ej. Jetstream en Laravel).
- Stateful – Autenticación basada en estado (como cookies) en lugar de tokens sin estado.
- Parameter Binding – Técnica para prevenir inyección SQL al vincular parámetros en consultas.
- Hashing – Proceso irreversible para convertir datos (como contraseñas) en cadenas seguras.
- Bcrypt Rounds – Número de iteraciones para fortalecer el hashing (ej. 12 rondas).
- Guard (Laravel) – Mecanismo que define cómo se autentican los usuarios (ej. web, sanctum).
- Livewire (Pila) – Enfoque para construir interfaces dinámicas sin escribir JavaScript.
- Enlace Simbólico (Symlink) – Acceso directo del sistema a archivos o carpetas (ej. storage:link).
- Queue (Cola) – Procesamiento asíncrono para tareas pesadas (ej. queue:listen).
- Middleware CSRF – Capa que verifica solicitudes para evitar falsificación.
- Eloquent ORM – Implementación de Laravel para interactuar con bases de datos usando objetos.
- Artisan – CLI de Laravel para ejecutar comandos de mantenimiento (ej. migrate).
- Jetstream Teams – Funcionalidad para gestión de equipos en autenticación.
- npm run build – Comando para compilar assets frontend (JS/CSS).
- Supervisor – Demonio para monitorear y reiniciar procesos críticos (ej. colas).