Ministerio
de Ambiente

División información ambiental

Uruguay

Documentación

Sistema de recepción discreto (SRD)

Link al repositorio:

https://github.com/ambiente-uy-github/SID-SRD

2023

ÍNDICE

Introducción

En el siguiente documento se detalla la arquitectura propuesta para comenzar el traspaso del SIA a una nueva plataforma que recibe los datos discretos de las empresas, sus plantas y datos de relevamientos de los técnicos.

Propósito

El propósito de este documento es proveer una especificación de la arquitectura de la aplicación “SRD”(Sistema de recepción discreta). Se mostrarán distintas vistas arquitectónicas que expondrán las características del sistema.

Antecedentes

Propósito del sistema

"**SRD**" es un sistema de recepción de datos discretos de las empresas, estaciones y datos relevados en salidas de campo, estos datos son obtenidos del sistema actual (SIA) pero permitiendo a futuro conectarse con otro sistema para obtener los datos. La aplicación cuenta con un amplio abanico de funcionalidades;

Los principales usuarios y entidades que utilizarán la solución son:

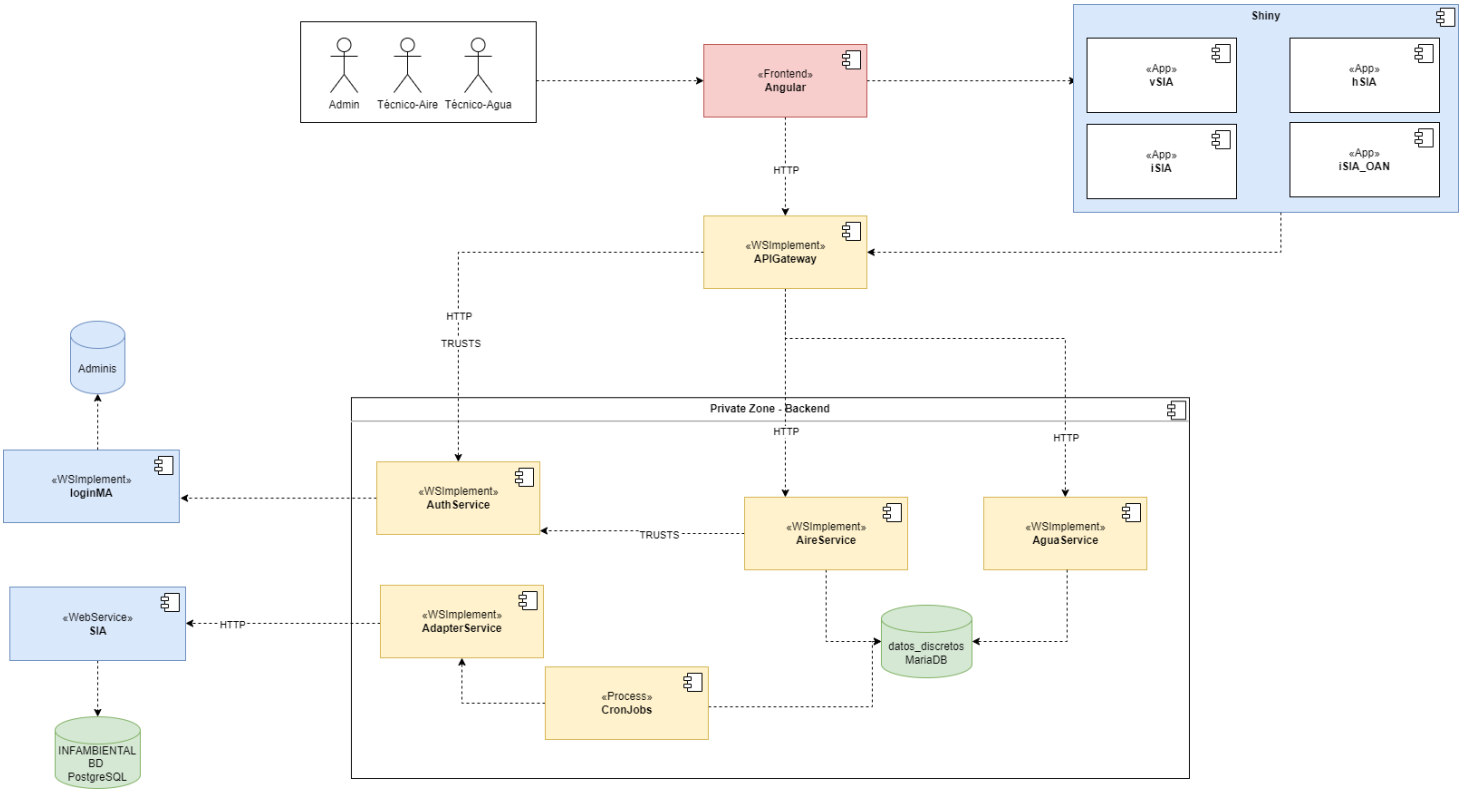
• **Admin:** Tienen acceso a toda la aplicación sin ninguna restricción.
• **Técnico:** Son los encargados de realizar las diferentes tareas de acuerdo al área que estos pertenecen. Y estos son:

- **Técnico-Aire:** Pueden visualizar todas las funcionalidades relacionadas a aire
- **Técnico-Agua:** Pueden visualizar todas las funcionalidades relacionadas a agua

Documentación de la arquitectura

En las diferentes etapas de este capítulo se documenta la arquitectura del sistema desde distintas perspectivas: módulos y componentes y conectores.

Vista de componentes y conectores general



Catálogo de elementos

ELEMENTO	TIPO	RESPONSABILIDADES
Auth service	Servicio	Es un intermediario con el loginMA. Proveedor de token de acceso para los usuarios.
loginMA	Servicio	Servicio externo, encargado de proveer los token de acceso.
Adapter service	Servicio	Es un intermediario entre el sistema y el SIA.
Cron Jobs	Tarea periodica	Este componente permite realizar la migración y actualización diaria de los datos del SIA al nuevo sistema.
Aire service	Servicio	Se encarga de la creación de nuevos reportes de calidad de aire, estaciones de aire, de visualizar los datos de las empresas y plantas.
Agua Service	Servicio	
API Gateway	Servicio	Es el gestor de tráfico que interactúa con el servicio

		backend real y aplica políticas, autenticación y control de acceso general para las llamadas de una API.
Shiny	Aplicación Web	Es un paquete de R que contiene una aplicación web con datos, validación de los mismos y gráficos.
Adminis	Base de datos	Base de datos guarda información de los usuarios del SIA.
Datos_discretos	Base de datos	Base de datos MariaDB guarda información de empresas, plantas, datos de calidad de aire y estaciones.
InfambientalBD	Base de datos	Base de datos PostgreSQL contiene toda la información de los datos del SIA.

Decisiones de diseño

En cuanto a las decisiones de diseño se opta por una arquitectura en microservicios. Esto se debe a que una arquitectura de microservicios consta de una colección de servicios autónomos y pequeños. Cada uno de los servicios es independiente y debe implementar una funcionalidad de negocio individual. Esta decisión se basa principalmente por:

- **Escalabilidad**:

Los servicios se pueden escalar de forma independiente, lo que permite escalar horizontalmente los subsistemas que requieren más recursos, sin tener que escalar horizontalmente toda la aplicación.

- **Modificabilidad/Mantenibilidad:**

La arquitectura de microservicios minimiza las dependencias y resulta más fácil agregar nuevas características. Dado que los microservicios se implementan de forma independiente, resulta más fácil administrar las correcciones de errores y las versiones de características. Puede actualizar un servicio sin volver a implementar toda la aplicación y revertir una actualización si algo va mal.

Logs

Para la auditoría se utiliza Winston, es una popular biblioteca de registro de eventos en el ecosistema de Node.js que se utiliza ampliamente para gestionar registros y registros de aplicaciones. Estos son manejados de forma independiente en cada uno de los componentes de la aplicación. Dentro de cada componente se encuentra una carpeta de Logs, estos son agrupados por día en diferentes archivos, y los mismos tienen una duración de 30 días luego de esta fecha se eliminan. Existen dos tipos de logs LogError o LogInfo, en cada uno de estos logs se guarda el día y hora de la acción que fue ejecutada y a su vez el nombre de usuario.

Seguridad

Este sistema cuenta con varias verificaciones en el aspecto de seguridad que son relevantes a tener en cuenta.

1. **Fuerza bruta en login:** Se introduce un captcha de google que permite el bloqueo en caso de varios intentos fallidos desde el frontend.

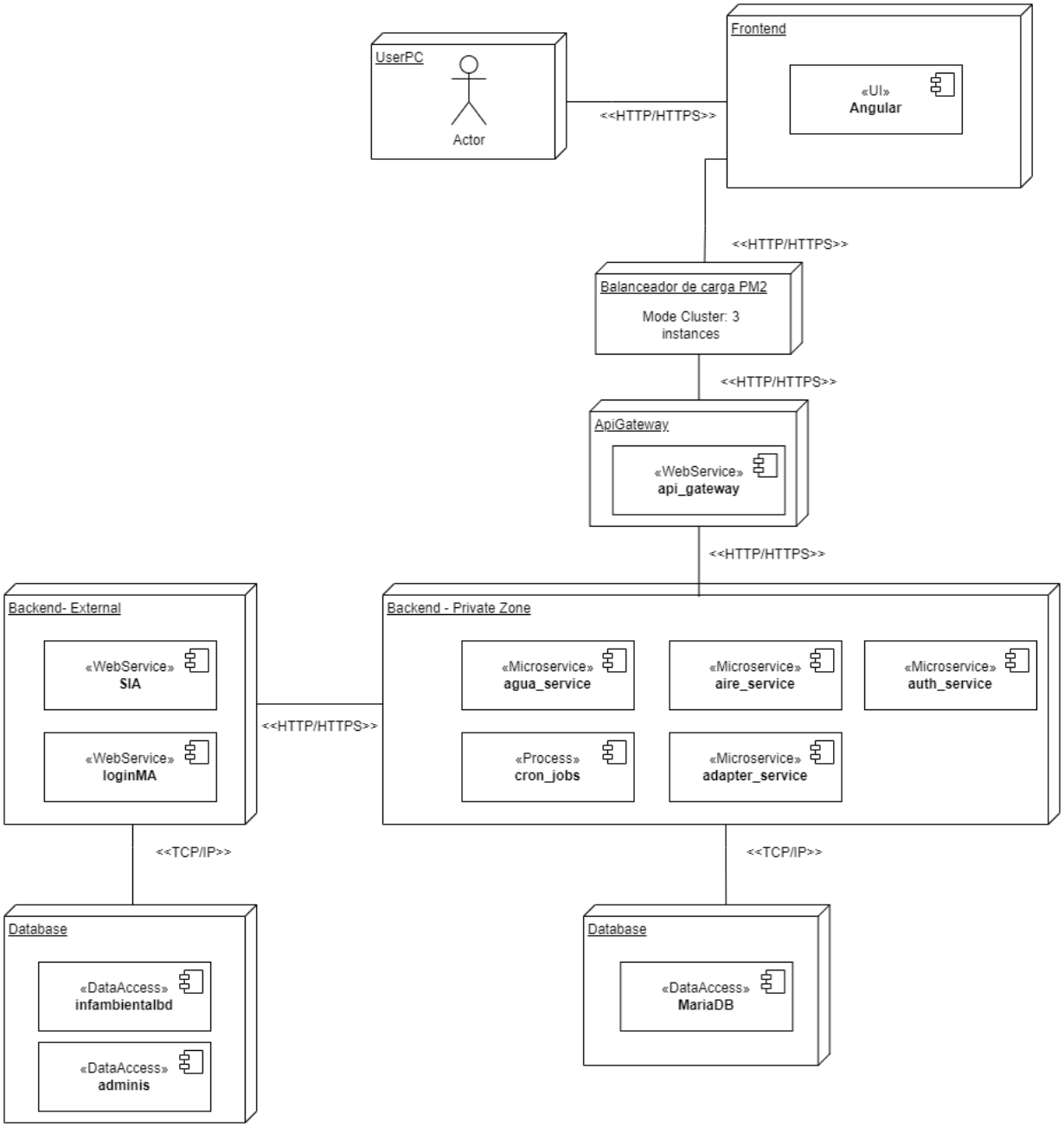
2. **Rate Limiter:** Se introduce en cada servicio en Node js, para evitar ataques del tipo DDos. La configuración es modificable, se puede observar la misma dentro de cada servicio. Ya que el tiempo y la cantidad de requests soportados por IP fue pensada para cada uno de estos servicios.

3. **XSS(Sanitización):** Dentro de la carpeta shared se encuentra sanitizer_middleware, este intermediario verifica que las request, no tengan Cross Site Scripting previniendo que los body o queryparams tengan esa malformación. Este middleware es llamado dentro de cada controlador que contiene información proveniente del exterior al backend.

4. **Encabezados de seguridad:** Se agregan varios encabezados de seguridad en toda la aplicación para todos los servicios. Estos son:
    - HSTS
    - X-FRAME-OPTIONS
    - CSP

Vista de despliegue

El uso del patrón Multi-Tier ha llevado a que el software se divida en tres capas distintas de componentes. La de presentación web, que es el frontend de nuestra solución, donde se encuentra la interfaz de usuario en la que los usuarios finales interactúan. La otra capa es la de aplicación, que contiene el back-end de la solución, donde se procesan las solicitudes del usuario y se realizan las operaciones de lógica de negocios. Finalmente, última capa es la de base de datos, que compone el nivel más bajo de la solución, y es responsable de la gestión y almacenamiento de los datos utilizados por la aplicación.



Catálogo de elementos

NODO	RESPONSABILIDADES
Frontend	Nodo dedicado a renderizar componentes hacia el usuario.
UserPC	Nodo el cual representa al cliente, el mismo interactúa con nuestro sistema por medio del navegador web.
Balanceador de carga PM2	Nodo el cual se encarga de balancear la carga proveniente del frontend.
ApiGateway	Nodo encargado de redirigir las peticiones a cada uno de los microservicios correspondientes. Abstrae las ubicaciones de los mismos.
Backend - Private zone	Nodo el cual contiene toda la lógica del negocio de este sistema.
Backend - External	Nodo el cual contiene la lógica del negocio de los sistemas externos.
Database	Nodo el cual representa el almacenamiento de datos.

Vista de modelo de datos

Para la base de datos se utilizó una base de datos relacional MariaDB. La misma cuenta con varias tablas. A su vez cuenta con algunas vistas que se realizaron ya que algunas filas tenían claves foráneas ya asociadas, y en esta nueva aplicación se deseaban omitir para evitar errores de ingreso.

Deploy

El deployment de esta aplicación está realizado en un servidor Debian 11. Los servicios son levantados con un balanceador de carga (PM2) todos están en modo fork, a excepción del api_gateway que está en modo cluster, y se pusieron 3 instancias, estos fueron inicializados desde un archivo ubicado dentro de Backend/PM2/ecosystem.config.js.

Comandos para controlar el PM2 desde la consola del servidor:

- pm2 start ecosystem.config.js (Hay que estar parados en la carpeta PM2)
    - Este comando ejecuta e inicializa todos los servicios que se encuentran declarados dentro de ese archivo

- pm2 stop nombreDelServicio
    - Detiene el servicio al cual se hace referencia
    - Si se quiere detener todos en vez de nombreDelServicio se debe poner all

- pm2 delete nombreDelServicio
    - Elimina el servicio al cual se hace referencia
    - Si se quiere eliminar todos en vez de nombreDelServicio se debe poner all

- pm2 list
    - Enumera los servicios que se encuentran y el estado de los mismos, tambien informacion acerca del uso de cpu,memoria,etc

- pm2 monit
    - Permite en tiempo real observar los logs de la app e información relacionada a las métricas del servicio.

Nota: Por más información y otras flags se puede ver la documentación oficial:
https://pm2.keymetrics.io/docs/usage/quick-start/