DOUBLE V PARTNERS

Manual técnico

[BackOffice CVU][Frontend][WEB]

Elaborado por Double V Partners: Equipo de desarrollo

2024

Tabla de contenido

Introducción

El presente manual técnico ofrece una guía sobre la estructura y desarrollo del proyecto. Este documento está diseñado para proporcionar a los desarrolladores y demás miembros del equipo una referencia completa que les permita comprender y trabajar de manera eficiente con el código del proyecto.

En este manual, se explorará en detalle la arquitectura del proyecto, destacando sus componentes fundamentales, estructura de directorios, patrones de diseño utilizados y buenas prácticas de programación implementadas. Además, se proporcionarán instrucciones claras para la instalación, configuración y mantenimiento del sistema, con el objetivo de garantizar su eficiencia y escalabilidad a lo largo del tiempo.

Objetivo del manual técnico

El objetivo principal de este manual es proporcionar una referencia completa y detallada sobre la estructura y desarrollo del proyecto. Este documento tiene como finalidad facilitar la comprensión y el trabajo eficiente con el código del proyecto, tanto para los desarrolladores como para otros miembros del equipo involucrados en su implementación y mantenimiento.

Audiencia

Este manual está destinado a ser utilizado por los siguientes grupos de personas:

Desarrolladores:

● Aquellos responsables de construir y mejorar el software. Necesitan una comprensión profunda de la estructura del proyecto, las tecnologías utilizadas y los estándares de codificación para contribuir eficazmente al desarrollo del proyecto.

● Administradores del sistema:
Los encargados de configurar y mantener el entorno de producción, asegurando que la aplicación se despliegue y funcione correctamente. Este manual les proporcionará información crucial sobre los requisitos del sistema, la configuración del entorno y las tareas de mantenimiento necesarias para garantizar un funcionamiento óptimo del sistema.

● Otros miembros del equipo: Cualquier persona que participe en el desarrollo, despliegue o mantenimiento del proyecto y que requiera una visión general y detalles técnicos sobre el mismo.

Estructura del manual técnico

Este manual técnico sigue una estructura organizada en diversas secciones temáticas que cubren distintos aspectos del proyecto. A continuación, se presenta una descripción general de cada sección:

● Requisitos del sistema:
Enumera los requisitos técnicos y de infraestructura necesarios para ejecutar el software de manera efectiva en distintas plataformas (Windows, macOS).

● Arquitectura y estructura del proyecto:
Describe la arquitectura general del proyecto, incluyendo los componentes principales, la estructura de directorios y los patrones de diseño utilizados en la implementación.

● Tecnologías utilizadas:
Detalla las tecnologías, frameworks y librerías empleadas en el desarrollo del proyecto, proporcionando información sobre su propósito y funcionalidad.

● Instalación y configuración:
Proporciona instrucciones detalladas y paso a paso sobre cómo configurar el entorno de desarrollo y llevar a cabo la instalación inicial del software en los diferentes sistemas operativos compatibles.

● Desarrollo y uso del software:

Explica cómo utilizar las funcionalidades principales del software y cómo interactuar con la interfaz de usuario, ofreciendo ejemplos prácticos y guías de uso.

● Despliegue y mantenimiento:
Describe los procesos y consideraciones relacionados con el despliegue inicial y el mantenimiento continuo del sistema en producción.

● Glosario de términos:
Incluye una lista de los términos técnicos y conceptos clave utilizados en el proyecto, junto con sus definiciones, para facilitar la comprensión del contenido técnico del manual.

1. Requisitos del Sistema

1.1 Requisitos del Sistema para Angular (Frontend)

Para la instalación y desarrollo del proyecto en Angular, el entorno de desarrollo debe cumplir con los siguientes requisitos mínimos en las plataformas Windows, Linux o macOS.

Windows

- Sistema operativo:
  - Windows 10 o posterior (64 bits), basado en x86-64.
- Espacio en disco:
  - Al menos 10 GB de espacio libre en disco
- Herramientas: Es necesario contar con las siguientes herramientas en el entorno:
  - Windows PowerShell 5.0 o superior (preinstalado en Windows 10).
  - Node.js (última versión estable).
  - npm (Node Package Manager).
  - Git for Windows 2.x, con la opción "Usar Git desde el símbolo del sistema de Windows".

Linux

- Sistema operativo:
  Linux (64 bits).
- Espacio en disco:

    - Se requieren al menos 600MB de espacio libre (excluyendo el espacio para IDE/herramientas).
- Herramientas: Es necesario contar con las siguientes herramientas en el entorno:
    - bash
    - curl
    - git 2.x
    - Node.js (última versión estable).
    - npm (Node Package Manager).

Mac

- Sistema operativo:
    - macOS, versión 10.14 (Mojave) o posterior.
- Espacio en disco:
    - Se requieren al menos 2 GB de espacio libre
- Herramientas:
    - Xcode Command Line Tools (se pueden instalar a través de la terminal con xcode-select --install).
    - Node.js (última versión estable).
    - npm (Node Package Manager).

Es fundamental asegurarse de que el entorno cumpla con estos requisitos mínimos antes de iniciar el desarrollo del proyecto en cada plataforma respectiva. Además, se recomienda mantener actualizadas todas las herramientas y dependencias necesarias para el desarrollo continuo del proyecto.

2. Arquitectura y estructura del proyecto

El proyecto sigue una arquitectura basada en tres capas fundamentales: la Capa de Presentación, la Capa de Dominio y la Capa de Datos. Esta arquitectura se inspira en los principios de Clean Architecture, que promueven la separación clara de responsabilidades y la escalabilidad del sistema.

Capa de Presentación:

- Esta capa se encarga de la interfaz de usuario y la interacción con el usuario final.
- Incluye componentes de Angular como componentes, directivas y servicios relacionados con la presentación de la interfaz de usuario.
- Aquí se definen las vistas, la lógica de presentación y la interacción del usuario con la aplicación.

Capa de Dominio:

- En esta capa se encuentran las reglas de negocio y la lógica de la aplicación.
- Contiene los servicios, modelos y lógica de negocio que definen cómo funcionan las diferentes funcionalidades de la aplicación.

Capa de Datos:

- Esta capa se encarga de acceder y gestionar los datos de la aplicación.
- Incluye servicios y utilidades para interactuar con fuentes de datos externas, como APIs REST, bases de datos u otros sistemas.
- Aquí se implementa la lógica para recuperar, almacenar y manipular los datos necesarios para el funcionamiento de la plataforma.

2.1 Estructura de carpetas y archivos principales

El proyecto se estructuró de manera organizada separando las características principales de los demás componentes. A continuación se detalla la estructura principal:

- **src/**: Es la carpeta principal que contiene el código fuente del proyecto.
- **src/assets/**: En está carpeta se encuentran todos los archivos multimedia utilizados en el proyecto.
- **src/data**: Contiene la estructura de los modelos, repositorios y data sources que serán usados en sus correspondientes features.
- **src/domain**: Contiene la estructura de las entidades, repositorios casos de usos entre otros que serán usados en sus correspondientes features.
- **src/presentation**: Contiene todas las vistas que representan las diferentes features de la aplicación. Dentro de cada feature contiene los componentes específicos del módulo.
- **src/environments**: Carpeta que contiene los archivos de configuración de entorno, como los archivos de configuración para desarrollo, producción.
- **src/presentation/app**: Acá se encuentran las características o funcionalidades

principales de la aplicación.
- **src/presentation/app/shared**: Contiene componentes, directivas, pipes y módulos compartidos entre diferentes partes de la aplicación.
- **src/presentation/app/shared/services:** Contiene los archivos con los diferentes servicios como almacenamiento, tema e inicio de sesión entre otros.
- **src/presentation/app/shared/guards:** Contiene los guards de ruta utilizados para la seguridad de la aplicación.
- **src/presentation/app/shared/interceptors:** Contiene los interceptores HTTP utilizados para modificar las peticiones HTTP.
- **src/styles.css**: Contiene los archivos de estilos globales.

2.2 Funcionalidades

En la arquitectura que se maneja en el proyecto las funcionalidades principales de la aplicación se separan y cada una de estás tiene su propia estructura de Clean Architect. Las funcionalidades desarrolladas se definen a continuación:

i. **Login**: Los usuario administradores pueden realizar el inicio con su usuario (email) y contraseña

ii. **Recuperar contraseña**: Permite a los usuarios recuperar su contraseña en caso de olvido.
Sus principales funciones:
   - Envío de correo electrónico de recuperación.
   - Restablecimiento de la contraseña del usuario.

iii. **Home**: Vista principal donde se muestra una lista de usuarios registrados.
Sus principales funciones:
   - Mostrar lista de usuarios registrados.
   - Opción para ir a los detalles de cada usuario.
   - Menú para ir al perfil del usuario o cerrar sesión.

iv. **Detalles usuario**: Vista donde se muestra información del usuario seleccionado.
Sus principales funciones:
   - Mostrar datos personales del usuario.
   - Ver planes activos, borradores y finalizados del usuario.

v. **Perfil administrador**: Vista donde se muestra la información básica del administrador.
Sus principales funciones:
   - Mostrar nombre y correo electrónico del administrador.

- Opción para cambiar la contraseña.

vi. **Edición de contraseña**: Funcionalidad que permite a los usuarios cambiar su contraseña.
Sus principales funciones:
- Formulario para ingresar la contraseña actual.
- Formulario para ingresar y confirmar la nueva contraseña.
- Validación y actualización de la nueva contraseña.

vii. **Cerrar Sesión:** Permite a los usuarios cerrar su sesión de manera segura.
Sus principales funciones:
- Cerrar la sesión del usuario.
- Redirección a la pantalla de inicio de sesión.

3. Instalación y Configuración

El proceso de instalación de Angular es muy similar en los sistemas operativos Windows, macOS y Linux. Sin embargo, sigue las instrucciones específicas de tu sistema para obtener los mejores resultados.

1. Instalar Node.js y npm
   Antes de poder instalar Angular, necesitarás tener Node.js y npm (Node Package Manager) instalados en tu máquina.
2. Instalar Angular CLI Globalmente
   A continuación, tendrás que instalar la herramienta Angular CLI. Para instalar la CLI de Angular globalmente en tu sistema, abre el símbolo del sistema y ejecuta el siguiente comando:

   ```
   npm install -g @angular/cli
   ```

   Este comando instala la última versión estable de la herramienta CLI de Angular y hace que esté disponible para su uso en todo tu sistema.
3. Ejecutar los Comandos de la CLI de Angular
   Una vez que tengas instalada la CLI de Angular, puedes utilizar sus comandos para gestionar tus proyectos de Angular. Para comprobar que la instalación se ha realizado correctamente, ejecuta el siguiente comando en el símbolo del sistema:

   ```
   ng --version
   ```

   Este comando muestra la versión instalada de la CLI de Angular, junto con otra

información relevante sobre tu entorno.

3.2 Clonación y configuración del repositorio

1. Abra una terminal o línea de comandos y navegue hasta el directorio donde desea clonar el repositorio.
2. Abre sesión en el servicio de CodeCommit AWS o ingresa al siguiente link https://us-east-1.console.aws.amazon.com/codesuite/codecommit/repositories/Cvu.Backoffice/browse?region=us-east-1.
3. En la página principal de CodeCommit, seleccione el repositorio que desea clonar.
4. Haga clic en el botón "Clonar URL HTTP" o "Clonar URL SSH", dependiendo de la opción que desee utilizar para la autenticación.
5. Copie la URL del repositorio proporcionada.
6. Vuelva a la terminal o línea de comandos y ejecute el siguiente comando para clonar el repositorio utilizando la URL que copió en el paso anterior: Para clonar utilizando HTTPS:

   $ git clone url_clone

7. Se le pedirá que proporcione sus credenciales de Git para la autenticación. Ingrese sus credenciales de Git cuando se le solicite.
8. Una vez completada la clonación, el repositorio estará disponible localmente en su sistema.

3.3 Configuración de dependencias y librerías adicionales

En el desarrollo de una aplicación Angular, se pueden utilizar diversas dependencias y librerías adicionales para ampliar las funcionalidades y mejorar la experiencia del usuario. A continuación, se presenta una breve descripción de cada una de las dependencias mencionadas:

- **@angular/fire:** Una integración de Angular con Firebase que simplifica la creación de aplicaciones web en tiempo real, almacenamiento en la nube, autenticación de usuarios y mucho más.

- **@ngrx/effects:** Esta librería proporciona una forma de manejar efectos secundarios en las aplicaciones Angular utilizando la arquitectura Redux.

- **@ngrx/store:** Implementa un patrón de administración de estado inspirado en Redux para Angular, lo que facilita el manejo del estado de la aplicación de manera predecible y eficiente.

- **html-to-image:** Una librería que permite convertir elementos HTML en

imágenes, útil para generar capturas de pantalla o imágenes de elementos de la interfaz de usuario.

- **ng2-pdf-viewer:** Un visor de PDF para Angular que permite la visualización de archivos PDF dentro de una aplicación web Angular.
- **rxjs:** Una biblioteca para programación reactiva en JavaScript que proporciona un conjunto de operadores para trabajar con secuencias de datos asíncronas.
- **bootstrap:** Un popular framework de CSS para el diseño de interfaces de usuario responsivas y móviles primero.
- **natural-compare:** Una librería que proporciona una comparación de cadenas que respeta el orden natural, útil para ordenar listas alfanuméricas.
- **jest:** Un framework de pruebas unitarias para JavaScript, que proporciona una experiencia completa y fácil de usar para escribir y ejecutar tests.
- **jest-preset-angular:** Un preset para configurar Jest con Angular, facilitando la integración de pruebas unitarias en aplicaciones Angular.
- **sonarqube:** Herramienta para el análisis continuo de la calidad del código, integrándose con SonarQube para medir y analizar la calidad del código fuente.
- **sonarqube-scanner:** Un cliente para ejecutar análisis de código con SonarQube desde la línea de comandos.

4. Uso del desarrollo

4.1 Ejecución del proyecto

En esta sección, se detallan los pasos necesarios para ejecutar el proyecto en el entorno de desarrollo. Se requiere haber completado la configuración previa antes de seguir estos pasos. A continuación, se describen los pasos para ejecutar el proyecto.

1. **Seleccionar la versión de Angular:** Se recomienda utilizar la versión 15 de Angular.
2. **Construir el proyecto**: Ejecuta el comando npm install en la terminal para descargar todas las dependencias y paquetes necesarios para el proyecto.
3. **Ejecutar el servidor de desarrollo:** Después de instalar las dependencias,

puedes ejecutar el servidor de desarrollo de Angular utilizando el comando ng serve en la terminal. Esto iniciará el servidor y tu aplicación estará disponible en http://localhost:4200/, donde podrás verla en tu navegador web.

4.2 Estructura y organización de la capa de datos

En la arquitectura de Clean Architecture, los Data Sources son componentes que actúan como intermediarios entre la capa de dominio y las fuentes de datos externas, como APIs o bases de datos. Su función principal es proporcionar una abstracción que facilite el acceso y manipulación de los datos, permitiendo a la capa de dominio mantenerse independiente de las implementaciones específicas de las fuentes de datos. Los Data Sources garantizan una separación clara de responsabilidades y promueven la reutilización y la estabilidad en el desarrollo de aplicaciones. A continuación se listan los data source utilizados en el desarrollo de la aplicación.

- **Auth**
  - Descripción: Este módulo gestiona la autenticación de usuarios y la recuperación de contraseñas.
  - Métodos clave:
    - **login**

Url	https://dev.back.cvu.com.co/api/ms-admin/v1/Auth/login
Método http	POST

body	{
	"email": "string",
	"password": "string"
	}

    - **refresh-token**

Url	https://dev.back.cvu.com.co/api/ms-admin/v1/Auth/refresh-token
Método http	POST

body	{
	"refreshToken": "string"
	}

- **forgot-password**

Url	https://dev.back.cvu.com.co/api/ms-admin/v1/Auth/forgot-password
Método http	POST

body	{
	"email": "string"
	}

- **Catalog**
  - Descripción: Este módulo se encarga de obtener información relevante del catálogo, incluyendo tipos de documentos de identificación y lista de avances disponibles en un plan de ahorro.
  - Métodos clave:
    - **getCatalogDocumentTypes**

Url	https://dev.back.cvu.com.co/api/ms-admin/v1/Catalog/document-types
Método http	GET

    - **getCataloAdvancesAvailable**

Url	https://dev.back.cvu.com.co/api/ms-admin/v1/Catalog/advances-available/savings-plan/{savingsPlanId}
Método http	GET

- **TravelPlan**
  - Descripción: Este módulo gestiona los planes borradores, como la eliminación, edición y además tenemos un servicio para firma de contratos de suscripción.
  - Métodos clave:
    - **getPlan**:

Url	https://dev.back.cvu.com.co/api/ms-admin/v1/TravelPlan/{id}

Método http	GET

- **getPlans**:

Url	https://dev.back.cvu.com.co/api/ms-admin/v1/TravelPlan
Método http	GET
Params	UserId:string State:string

- **getPlansSummary**:

Url	https://dev.back.cvu.com.co/api/ms-admin/v1/TravelPlan/summary
Método http	GET
Params	UserId:string State:string

- **User**
  - Descripción: Este módulo gestiona la información y las operaciones relacionadas con los usuarios, incluyendo la obtención y actualización de datos.
  - Métodos clave:

- **getUser**:

Url	https://dev.back.cvu.com.co/api/ms-admin/v1/User/{idUser}
Método http	GET

- **getUsersPlansSummary**:

Url	https://dev.back.cvu.com.co/api/ms-admin/v1/User

Método http	POST
Params	Role:string

- **postChangePassword**:

Url	https://dev.back.cvu.com.co/api/ms-admin/v1/User/{userID}/change-password
Método http	POST

Body	{
	currentPassword: string,
	newPassword: string,
	}

5. Despliegue y Mantenimiento

5.1 Despliegue

Consulta los documentos adjuntos titulados "FUNCIONAMIENTO DE IMPLEMENTACIÓN DE CI/CD EN CVU" e "INFORME TÉCNICO DE INFRAESTRUCTURA DE CVU" para obtener una guía detallada sobre la infraestructura de la aplicación en la nube de AWS y el funcionamiento de la implementación de CI/CD en CVU.
Estos documentos proporcionan información valiosa sobre los componentes utilizados, los procesos de compilación, despliegue y automatización.

5.2 Mantenimiento

- **Actualización de Dependencias:**
  Regularmente, revisa y actualiza las dependencias de tu proyecto Angular, incluyendo Angular CLI, bibliotecas externas, etc.
  Utiliza herramientas como npm outdated para identificar las dependencias desactualizadas y npm update para actualizarlas.

- **Monitoreo y Registro:**

Implementa sistemas de monitoreo para estar al tanto de la salud y el rendimiento de tu aplicación Angular en producción.
Configura el registro de errores y eventos para poder detectar y solucionar problemas rápidamente.

- **Respaldo y Recuperación:**
  Realiza copias de seguridad regulares de tus datos y archivos críticos relacionados con el proyecto Angular.
  Ten un plan de recuperación en caso de que ocurra un fallo grave o una pérdida de datos.
- **Optimización del Rendimiento:**
  Monitorea el rendimiento de tu aplicación Angular y realiza ajustes según sea necesario para mejorar la velocidad de carga y la experiencia del usuario.
  Optimiza los activos estáticos, como imágenes y archivos CSS/JS, para reducir los tiempos de carga.

- **Gestión de Versiones:**
  Utiliza un sistema de control de versiones como Git para gestionar y mantener un historial de cambios en tu proyecto Angular.
  Implementa prácticas de ramificación y fusión para gestionar el desarrollo de nuevas funciones y correcciones de errores de manera eficiente.

6. Glosario de términos

- **Angular:** Framework de desarrollo web de código abierto, mantenido por Google, que permite construir aplicaciones web dinámicas y de alta calidad utilizando TypeScript.
- **Component (Componente):** Unidad básica de la interfaz de usuario en Angular, que incluye una plantilla HTML, lógica de control en TypeScript, y estilos CSS asociados.
- **Directive (Directiva):** Instrucción que se aplica a un elemento del DOM para modificar su comportamiento o apariencia. Las directivas en Angular pueden ser estructurales (como *ngIf y *ngFor) o de atributo (como ngClass y ngStyle).
- **Interceptor:** Servicio que implementa el patrón de diseño interceptor, utilizado para modificar solicitudes y respuestas HTTP globalmente antes de que lleguen a la aplicación o al servidor.
- **Module (Módulo):** Contenedor que agrupa componentes, directivas, pipes y servicios relacionados, organizando el código en unidades funcionales. El módulo raíz es AppModule.
- **NgModule:** Decorador utilizado para definir un módulo en Angular. Especifica las declaraciones, importaciones, exportaciones, proveedores y componentes bootstrap de un módulo.
- **Pipe (Tubería):** Función que transforma datos en plantillas Angular. Los pipes

pueden ser utilizados para formatear cadenas, números, fechas, y otros datos en las vistas.

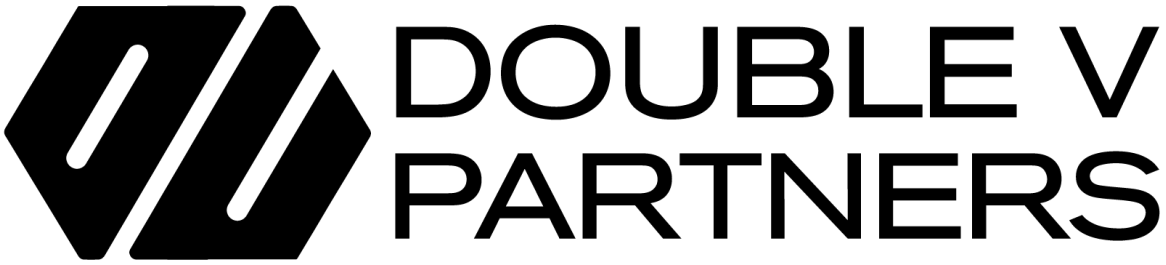
- **Reactive Forms:** Modelo de formularios en Angular que utiliza programación reactiva para gestionar la validación y el estado del formulario. Proporciona mayor control y flexibilidad en comparación con los formularios basados en templates.
- **Router (Enrutador):** Módulo que gestiona la navegación y las rutas dentro de una aplicación Angular. Permite definir rutas, asociar componentes a rutas y gestionar la navegación entre diferentes vistas de la aplicación.
- **RxJS:** Biblioteca para la programación reactiva que proporciona observables, operadores y herramientas para manejar eventos asíncronos y flujos de datos. Angular utiliza RxJS para manejar operaciones asíncronas como solicitudes HTTP.
- **API**: Interfaz de Programación de Aplicaciones (Application Programming Interface, en inglés) que define los métodos y protocolos utilizados para la comunicación entre componentes de software. Las APIs permiten la interacción entre distintas aplicaciones y servicios.
- **Base de datos**: Una base de datos es un sistema organizado para almacenar y gestionar grandes cantidades de datos. Se utiliza para almacenar y recuperar información de manera eficiente, y puede ser utilizado por las aplicaciones para almacenar y gestionar datos persistentes.
- **Caché**: La caché es una técnica utilizada para almacenar temporalmente datos en un lugar más rápido y accesible, con el objetivo de acelerar la recuperación de información. En el contexto de las aplicaciones, se utiliza para almacenar datos previamente obtenidos, como imágenes o respuestas de API, para evitar la necesidad de volver a descargarlos y mejorar el rendimiento de la aplicación.
- **Casos de uso:** Los casos de uso son escenarios específicos o funcionalidades que una aplicación puede realizar. Representan las acciones o interacciones que los usuarios pueden realizar en la aplicación para lograr un objetivo determinado.
- **Clase**: En programación orientada a objetos, una clase es una plantilla o prototipo que define las propiedades y comportamientos de un objeto. Las clases se utilizan para crear objetos y encapsular datos y funcionalidades relacionadas.
- **Clean Architecture: Metodología** de diseño de software que busca separar las preocupaciones en capas y mantener una estructura modular y desacoplada. Clean Architecture promueve la separación de la lógica de negocio de los detalles de implementación técnica.
- **Código fuente**: Conjunto de instrucciones escritas en un lenguaje de programación que constituyen un programa de software. El código fuente es editable y puede ser compilado o interpretado para generar un programa ejecutable.
- **Controlador**: En el contexto de una aplicación, un controlador es una clase o componente responsable de manejar la lógica y el flujo de datos relacionados con una determinada funcionalidad o vista.
- **CRUD**: CRUD es un acrónimo que representa las operaciones básicas de

gestión de datos en un sistema: Create (crear), Read (leer), Update (actualizar) y Delete (eliminar). Se utiliza para describir las operaciones estándar que se pueden realizar en una base de datos u otras fuentes de datos.

- **DataSource**: En el contexto de la capa de datos de una aplicación, un DataSource es un componente encargado de proporcionar y gestionar los datos utilizados por la aplicación. Puede interactuar con diferentes fuentes de datos, como bases de datos, APIs o sistemas externos.
- **Dominio**: En el desarrollo de software, el dominio se refiere al ámbito o área de conocimiento específica en la cual se encuentra una aplicación. Representa el problema o la temática principal que la aplicación resuelve.
- **Feature**: En el contexto del desarrollo de software, una feature (característica) se refiere a una funcionalidad específica o conjunto de funcionalidades que se agregan a una aplicación para proporcionar un valor adicional o cumplir con un requisito específico. Puede incluir cambios en la interfaz de usuario, mejoras en el rendimiento, nuevas capacidades, entre otros.
- **Framework**: Conjunto de herramientas, librerías y utilidades que proporcionan una estructura para el desarrollo de aplicaciones. Los frameworks permiten agilizar el proceso de desarrollo al proporcionar funcionalidades comunes y una arquitectura predefinida.
- **Git**: Sistema de control de versiones distribuido ampliamente utilizado para el seguimiento de cambios en el código fuente de proyectos de desarrollo de software. Git permite la colaboración entre desarrolladores, la gestión de ramas y la reversión de cambios, entre otras funcionalidades.
- **HTTP**: HTTP (Hypertext Transfer Protocol) es un protocolo utilizado para la comunicación entre clientes y servidores en la web. Permite la transferencia de información, como solicitudes y respuestas, entre un navegador web y un servidor, facilitando la visualización de páginas web y la interacción con aplicaciones en línea.
- **Interfaz**: En programación, una interfaz es un conjunto de métodos y/o propiedades que define un contrato para la comunicación entre diferentes componentes. Sirve como una especificación de cómo interactuar con un objeto sin preocuparse por su implementación concreta.
- **Inyección de dependencias**: Patrón de diseño y técnica que permite separar la creación y gestión de objetos de su utilización. La inyección de dependencias se utiliza para facilitar la modularidad, la reutilización y la prueba unitaria del código.
- **Librería**: Conjunto de código predefinido y reutilizable que proporciona funcionalidades específicas para ser utilizadas en el desarrollo de aplicaciones. Las librerías permiten ahorrar tiempo y esfuerzo al utilizar código ya existente en lugar de tener que escribirlo desde cero. Material: Material es un conjunto de directrices y principios de diseño desarrollados por Google para la creación de interfaces de usuario coherentes en aplicaciones. Estas directrices se basan en el uso de elementos de diseño, como colores, tipografía y disposición de elementos, para proporcionar una experiencia visualmente agradable y consistente.
- **Modelo**: En el desarrollo de software, un modelo es una representación estructurada de los datos y reglas de negocio de una aplicación. Los modelos

se utilizan para almacenar y manipular los datos de la aplicación de manera coherente y organizada.

- **Repositorio**: En el contexto de la arquitectura de software, un repositorio es un componente encargado de interactuar con los DataSources y proporcionar una capa de abstracción para acceder a los datos. Gestiona la obtención, creación, actualización y eliminación de datos, y proporciona una interfaz para que los otros componentes de la aplicación accedan a ellos.
- **Script**: En programación, un script es un conjunto de instrucciones o comandos que se ejecutan de manera secuencial para realizar una tarea o función específica. Los scripts suelen ser escritos en un lenguaje de programación y pueden ser utilizados para automatizar tareas, realizar cálculos o manipular datos.
- **Servicio**: En el desarrollo de software, un servicio es un componente que encapsula una funcionalidad específica y se utiliza para llevar a cabo tareas o procesos específicos. Puede involucrar operaciones complejas, interacciones con sistemas externos o lógica de negocio especializada.
- **Terminal**: El terminal es una interfaz de línea de comandos en la que los usuarios pueden interactuar con un sistema operativo mediante la ejecución de comandos. Permite realizar diversas tareas, como la navegación por el sistema de archivos, la compilación de programas y la administración de recursos.
- **URL**: Una URL (Uniform Resource Locator) es una dirección que se utiliza para identificar de manera única la ubicación de un recurso en Internet. Se compone de diferentes componentes, como el protocolo, el nombre de dominio, la ruta y los parámetros, y permite acceder a recursos como páginas web, archivos o servicios en línea.
- **Web**: En el contexto de desarrollo de aplicaciones, el término "web" se refiere a la plataforma y tecnologías utilizadas para desarrollar aplicaciones que se ejecutan en navegadores web. Estas aplicaciones son accesibles a través de Internet y no requieren una instalación específica en el dispositivo del usuario.



Manual técnico

[CVU][Backend][CVU.Admin]

Elaborado por Double V Partners: Equipo de desarrollo

2024

Tabla de contenido

1. Introducción

El presente manual técnico proporciona una guía detallada sobre el desarrollo, implementación y mantenimiento del proyecto CVU.Admin. Este documento está diseñado para brindar a los desarrolladores, administradores y otros miembros del equipo una referencia completa que les permita comprender y trabajar de manera efectiva con el software.

El manual aborda aspectos cruciales de la arquitectura, la configuración del entorno de desarrollo, la implementación de las diversas capas, así como las mejores prácticas específicas para optimizar el rendimiento y la escalabilidad del sistema. Se espera que este manual sirva como una guía completa y detallada para asegurar un entendimiento profundo del proyecto y facilitar un desarrollo eficiente y sostenible del backend de las plataformas de clientes y administradores.

1.1. Objetivo del manual técnico

El objetivo principal de este manual es proporcionar una guía completa sobre la estructura y las integraciones necesarias para el desarrollo del backend del proyecto. Este recurso está diseñado para ofrecer a los desarrolladores una comprensión profunda de la aplicación, facilitando así la construcción de un backend robusto, eficiente y escalable.

1.2. Audiencia y requisitos previos

Este manual está destinado a ser utilizado por los siguientes grupos de personas:

- **Desarrolladores:** Aquellos encargados de construir y mejorar el software, necesitan una comprensión profunda de la estructura del proyecto, las tecnologías usadas y los estándares de codificación.
- **Administradores del sistema:** Los encargados de configurar y mantener el entorno de producción, asegurando que la aplicación se despliegue y funcione correctamente.
- **Otros miembros del equipo:** Cualquier persona que participe en el desarrollo, despliegue o mantenimiento del proyecto y que requiera una visión general y detalles técnicos sobre el mismo.

Se presupone que los lectores cuentan con una familiaridad previa en los conceptos básicos de desarrollo web, así como en patrones de diseño y buenas prácticas de programación, además con experiencia en el lenguaje de programación C# y con la plataforma .NET Core.

1.3. Visión general del proyecto

Este proyecto se desarrolla abordando los principios básicos del Domain Driven Design (DDD) con un énfasis en CQRS (Command Query Responsibility Segregation). La arquitectura se divide en tres capas principales:

- **Web API:** En esta capa algo importante a detallar es que es aquí donde se implementa el patrón CQRS para gestionar comandos y consultas de manera separada. Los comandos, responsables de las modificaciones en la base de datos, se optimizan con Entity Framework. Las consultas se manejan específicamente para proporcionar respuestas eficientes a las solicitudes de lectura. Además, esta capa administra la conexión con servicios externos y microservicios, así como también incluye controladores para facilitar las interacciones con los clientes. Detalles específicos se abordarán más adelante.

- **Domain:** Define Aggregates, Events, Finders, Repositorios, y clases para la lógica de negocio esencial. Asegura una representación coherente de las entidades clave.

- **Infrastructure:** Implementa interfaces de Finders y Repositorios, y configura entidades y contexto. Gestiona consultas y modificaciones en bases de datos mediante Entity Framework.

Este enfoque proporciona una sólida estructura para el desarrollo, mantenimiento y evolución del proyecto, destacando la separación clara de responsabilidades y una representación coherente del dominio de negocio.

En la sección posterior se explicará más a detalle la arquitectura del proyecto, mencionando cómo se comunican estas capas entre sí, y además explicando a profundidad los patrones mencionados anteriormente.

Además de la estructura básica del proyecto, también se han integrado varias herramientas y servicios externos para mejorar la funcionalidad del sistema. Estas integraciones incluyen servicios de AWS, Firebase, Click&Sign, entre otros, de los cuales se hablará en el siguiente apartado.

En las secciones siguientes de este manual, se detallarán los pasos necesarios para configurar el entorno de desarrollo, implementar las diferentes capas de la aplicación, realizar pruebas y depuración.

Esperamos que este manual técnico sea una guía útil y completa para comprender y desarrollar eficientemente el backend del proyecto CVU.Admin.

2. Arquitectura del proyecto

En esta sección, se proporciona una descripción de la arquitectura del proyecto, la cual se inspira en los principios del Domain Driven Design (DDD), con un enfoque que prioriza la simplicidad y la adaptabilidad. Se hace un énfasis especial en la adopción del patrón arquitectónico CQRS (Command Query Responsibility Segregation), el cual promueve la separación de las operaciones de escritura, denominadas comandos, de las operaciones de lectura, conocidas como consultas. Esta combinación de enfoques arquitectónicos proporciona una base sólida para abordar la complejidad del dominio de negocio de manera efectiva. Si bien el proyecto no sigue una implementación rigurosa de DDD, la integración de sus principios básicos junto con CQRS permite una gestión eficiente de los datos y una adaptación flexible a las necesidades del negocio, asegurando una representación coherente y eficiente del dominio.

2.1. Descripción del patrón CQRS

CQRS (Command Query Responsibility Segregation) es un patrón de arquitectura que propone la división de la lógica de la aplicación en dos partes claramente diferenciadas: comandos y consultas. Los comandos representan las operaciones destinadas a la modificación de datos, como la creación, actualización o eliminación de registros. Por otro lado, las consultas se centran en obtener información del sistema sin realizar modificaciones en su estado.

La esencia de CQRS radica en reconocer y abordar de manera separada las responsabilidades asociadas con las operaciones de escritura y lectura. Esta separación facilita la optimización individual de cada parte de la aplicación, permitiendo ajustar y escalar de manera más eficiente tanto las operaciones de modificación como las de consulta. Al implementar CQRS, se busca maximizar la flexibilidad y el rendimiento de la aplicación al adaptarse de manera específica a los requisitos y características de cada tipo de operación.

2.2. Descripción del patrón DDD

La esencia del DDD radica en la identificación de límites en el diseño y definición de un proyecto. El enfoque de DDD nos ayuda a comprender la complejidad del dominio y a establecer límites significativos. Dentro de cada contexto delimitado, definimos entidades, objetos de valor y agregados que modelan el dominio específico.

Es crucial destacar que, mientras DDD se centra en la coherencia y la encapsulación a través de conceptos como agregados, CQRS apunta a proporcionar más flexibilidad en las consultas al liberarlas de las

restricciones impuestas por estos patrones. Esta combinación estratégica permite optimizar el rendimiento del sistema al tiempo que mantiene una representación coherente del dominio de negocio.

El objetivo final es lograr un equilibrio entre la consistencia del modelo de dominio proporcionado por DDD y la flexibilidad en las operaciones de lectura facilitada por CQRS. De esta manera, este proyecto busca obtener lo mejor de ambos patrones, asegurando una arquitectura robusta y adaptativa.

2.3. Descripción de las capas del proyecto

En el contexto del Domain Driven Design (DDD), la organización del código se basa en la definición de dominios y subdominios, en lugar de una estructura de capas estricta. Sin embargo, para facilitar la comprensión y mantenimiento del sistema, se reconoce la utilidad de una estructura clara y escalable para gestionar la complejidad del dominio.

A continuación, se detallarán las capas clave dentro de este enfoque, así como su contribución a la creación de un modelo de software coherente y orientado al negocio. Los directorios y archivos relevantes dentro de cada capa se explicarán en detalle en la sección de estructura del proyecto.

2.3.1. Capa de Web API

La capa de Web API en el proyecto desempeña un papel central al implementar el patrón CQRS. Aquí, los controladores son elementos clave que gestionan tanto los comandos, que representan acciones de modificación de datos iniciadas por los clientes, como las consultas, que responden a solicitudes de información.

Bajo la perspectiva de comandos, los controladores, siguiendo el principio de CQRS, encapsulan la lógica de negocio esencial y se comunican con los Aggregates en la capa de Domain para realizar modificaciones en la base de datos mediante Entity Framework. Por otro lado, en el ámbito de las consultas, estos mismos controladores aplican la lógica de CQRS al interactuar con interfaces de Aggregates implementadas en la capa de infraestructura, realizando consultas eficientes a las bases de datos mediante Finders o Repositories.

Esta separación clara de responsabilidades entre comandos y consultas en los controladores no solo optimiza el rendimiento y la escalabilidad del sistema, sino que también refleja la adaptabilidad y coherencia necesarias para gestionar eficientemente las operaciones de modificación y lectura en el proyecto.

En esta misma capa, se coordinan las conexiones e integraciones con los servicios externos empleados en este proyecto, que incluyen:

- **Amazon S3:** Amazon S3 es un servicio para almacenamiento en la nube que ofrece escalabilidad, disponibilidad de datos, seguridad y alto rendimiento. Utilizado en el proyecto para almacenar los contratos antes y después de su firma, correspondientes a los planes adquiridos por un cliente.
- **Atenea:** Web services que provee los métodos para la gestión de los contratos de ahorro de un cliente.
- **Authentication (Google Identity Platform):** Mediante el API de Identity Platform se gestionan los tokens de autorización de los usuarios.
- **Email (Zoho Mail):** Se utiliza el servidor SMTP de Zoho para enviar los diferentes correos según la necesidad de la funcionalidad, como verificación de cuenta, restablecimiento de contraseña, notificaciones, etc.
- **ESignature (click&sign):** A través de la API de click&sign de Lleida, se realiza el proceso de firma digital de los nuevos contratos adquiridos por un cliente.
- **Firebase:** El SDK de administración es un conjunto de bibliotecas de servidor que permite interactuar con Firebase desde entornos privilegiados para realizar acciones como gestión de usuarios, autenticación y autorización.

2.3.2. Capa de Dominio

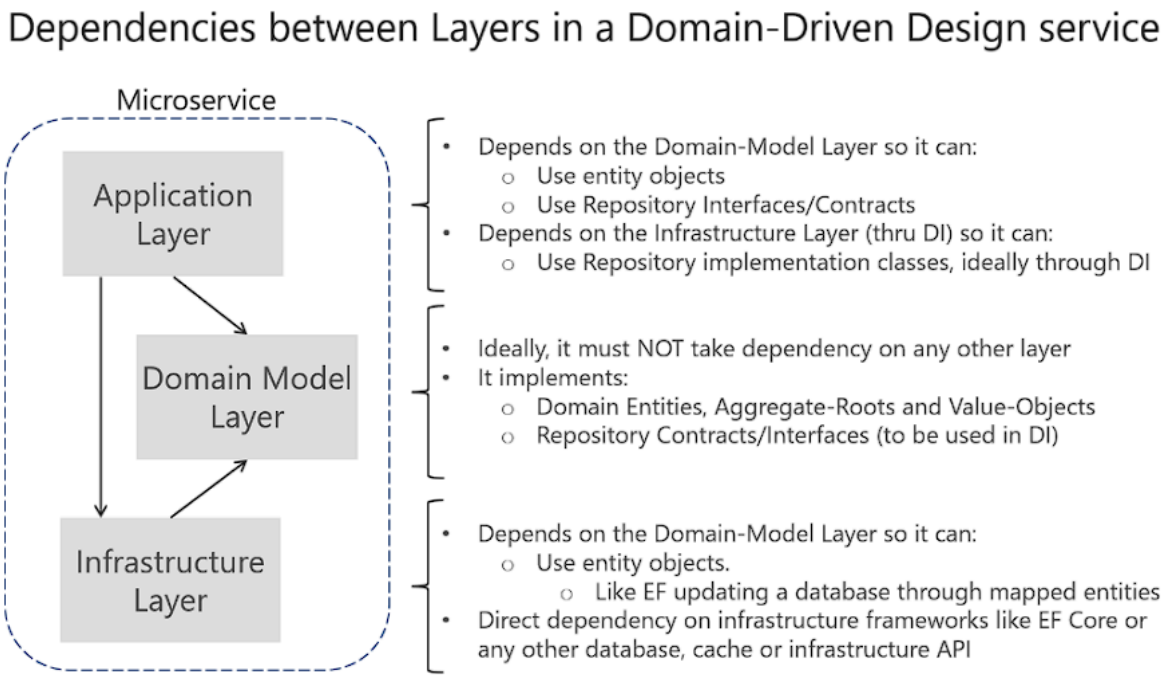
La capa de dominio en el proyecto es esencial y sigue el enfoque del DDD para crear software. Esta capa es el núcleo del sistema empresarial y se encarga de representar de manera sólida y coherente los conceptos, la información y las reglas de negocio. En esta capa, tenemos lo que llamamos Aggregates (Agregados), que son los elementos principales del dominio. Los Agregados agrupan la lógica y las reglas de validación para asegurar que los datos se mantengan consistentes y para manejar las operaciones de modificación del sistema. Cada Agregado tiene una raíz que actúa como punto de entrada exclusivo para hacer cambios en el grupo de objetos.

Esta capa de modelo de dominio, que está escrita en .NET como una biblioteca de clases, se enfoca en representar fielmente el negocio, capturando tanto los datos como los comportamientos importantes. Además, seguimos el principio de mantener esta capa independiente de la infraestructura de almacenamiento de datos. De esta manera, nos aseguramos de que las entidades de dominio no dependan de ningún marco específico para acceder a los datos, lo que ayuda a mantener su integridad y cohesión.

2.3.3. Capa de Infraestructura

La capa de Infraestructura, en el contexto del proyecto, desempeña una función crucial al definir la manera en que los datos, inicialmente almacenados en las entidades de dominio en memoria, se preservan en bases de datos o en almacenes permanentes. Un ejemplo práctico de esta implementación es el uso de código de Entity Framework Core (EF) para desarrollar las clases del patrón de repositorio, empleando DBContext para conservar los datos en una base de datos relacional. Es importante destacar que, en concordancia con los principios de Omisión de persistencia y Omisión de infraestructura, la capa de Infraestructura se mantiene independiente del nivel de modelo de dominio. Evita "contaminar" este nivel al no depender en exceso de los marcos para mantener las clases de entidad del modelo de dominio separadas de la infraestructura utilizada para conservar datos, ya sea mediante EF u otros marcos.

A continuación, una representación de la comunicación entre estas capas:



3. Infraestructura del proyecto

En este apartado se describe la infraestructura sobre la que se encuentra montado el proyecto CVU.Admin que está alojado en la Cloud AWS utilizando ECS Fargate.

3.1. Arquitectura

La arquitectura del proyecto consta de los siguientes componentes:

- **AWS CodeCommit:** Se utiliza como servicio para almacenar los repositorios del proyecto manejando así un control de versiones para gestionar el código fuente del proyecto.
  https://us-east-1.console.aws.amazon.com/codesuite/codecommit/repositories?region=us-east-1

- **Amazon Elastic Container Registry (ECR):** Se utiliza para almacenar y administrar las imágenes de contenedores Docker de la aplicación. Las imágenes se actualizan y versionan según sea necesario.
  https://us-east-1.console.aws.amazon.com/ecr/repositories?region=us-east-1

- **Load Balancer:** Balanceador de carga de AWS que distribuye el tráfico de red entrante a través de varios contenedores en ECS Fargate.
  https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LoadBalancers:

- **ECS Fargate:** Servicio de contenedores administrado por AWS que nos permite ejecutar contenedores en la nube sin necesidad de administrar servidores.
  https://us-east-1.console.aws.amazon.com/ecs/v2/clusters?region=us-east-1

- **Sonarqube:** Se integra para evaluar y mejorar la calidad del código de la aplicación. SonarQube realiza análisis estáticos del código y proporciona informes detallados sobre problemas de calidad y deuda técnica.
  La implementación de SonarQube se realizó a través de un contenedor ECS Fargate en AWS. Esta elección de arquitectura ofrece varias ventajas clave para asegurar un despliegue eficiente y escalable de SonarQube.
  https://us-east-1.console.aws.amazon.com/ecs/v2/clusters/dev-cvu/services/sonarqube/health?region=us-east-1

- **RDS Postgres:** Base de datos gestionada por AWS que utiliza el motor de base de datos PostgreSQL.
  https://us-east-1.console.aws.amazon.com/rds/home?region=us-east-1#databases:

3.2. Plan de automatización y despliegue

En el proyecto se implementó una estrategia de despliegue continuo (CI/CD) con herramientas como AWS CodePipeline, AWS Codebuild y AWS CodeDeploy. Con el objetivo de establecer un proceso de automatización y despliegue que

permita entregar de manera rápida y confiable nuevas versiones de la aplicación, manteniendo la calidad y la estabilidad del software.

3.3. Monitorización y registro

El servicio que se está utilizando para la monitorización y la generación de registros es Amazon CloudWatch el cual nos permite ver y analizar los logs dentro de la tarea de ECS Fargate.

4. Configuración del entorno de desarrollo

En esta sección, se detallan los requisitos del sistema, la instalación de las dependencias necesarias y la configuración del entorno de desarrollo para el proyecto. Es fundamental contar con un entorno de desarrollo adecuado para asegurar un proceso de desarrollo fluido y eficiente.

4.1. Requisitos del sistema

Para instalar y utilizar C# y .NET Core, es necesario asegurarse de que el entorno de desarrollo cumpla con los siguientes requisitos mínimos en las plataformas Windows, Linux y macOS.

4.1.1. Windows

- Sistema operativo:
    - Windows 10 o posterior (64 bits), basado en x86-64.
- Espacio en disco:
    - Se requieren al menos 4 GB de espacio libre para la instalación de .NET y las herramientas relacionadas.
- Herramientas
    - Descarga e instala el SDK de .NET 6 para Windows.
- Entorno de Desarrollo Integrado (IDE) (Opcional):
    - Visual Studio: Proporciona una experiencia de desarrollo completa para C# y .NET. Instala la carga de trabajo "Desarrollo de .NET" durante la instalación de Visual Studio.

4.1.2. Linux

- Sistema operativo:
  - Linux (64 bits).
- Espacio en disco:
  - Se requieren al menos 4 GB de espacio libre para la instalación de .NET y las herramientas relacionadas.
- Herramientas
  - Descarga e instala el SDK de .NET 6 para Linux, sigue los pasos de la documentación oficial de .NET para linux para instalar el paquete correspondiente a tu distribución.

4.1.3. MacOS

- Sistema operativo:
  - MacOS, versión 10.14 (Mojave) o posterior.
- Espacio en disco:
  - Se requieren al menos 4 GB de espacio libre para la instalación de .NET y las herramientas relacionadas.
- Herramientas
  - Descarga e instala el SDK de .NET 6 para MacOS.
- Entorno de Desarrollo Integrado (IDE) (Opcional, pero recomendado):
  - Visual Studio para Mac: Ofrece una experiencia de desarrollo similar a Visual Studio en Windows, con soporte completo para C# y .NET.

Es importante tener en cuenta que Visual Studio es opcional, y puedes utilizar otros editores como Visual Studio Code para desarrollar en .NET. Consulta la documentación oficial de .NET y del IDE seleccionado para obtener información detallada y actualizada.

4.2. Instalación de .NET 6

Puede descargar el SDK y el Runtime de .NET 6 del sitio oficial en https://dotnet.microsoft.com/es-es/download/dotnet/6.0.

Para realizar la instalación, siga las instrucciones detalladas en la documentación oficial dependiendo del sistema operativo.

Para comprobar que la instalación se realizó correctamente, ejecute en una terminal el siguiente comando:

```
Unset

$ dotnet --version
```

Debería tener una salida como la siguiente:

```
Unset

6.0.422
```

4.3. Clonación y configuración del proyecto

1. Clone el repositorio.

```
Unset

$ git clone https://git-codecommit.us-east-1.amazonaws.com/v1/repos/Cvu.Back
```

2. Navegue al directorio del proyecto.

```
Unset

$ cd Cvu.Back
```

3. Compile el proyecto.

```
Unset

$ dotnet build
```

Esto restaurará todas las dependencias del proyecto y lo compilará.

4.4. Otras dependencias y librerías adicionales

- **Autofac:** Autofac es un contenedor de .NET IoC que gestiona las dependencias entre clases, de modo que las aplicaciones se pueden modificar fácilmente incluso a medida que su escala y complejidad crecen.
- **AWSSDK:** AWS SDK para .NET simplifica el uso de los servicios de AWS ya que ofrece un conjunto de bibliotecas que resultan lógicas y familiares al desarrollar con .NET. AWS SDK para .NET ofrece soporte para S3, Dynamo DB, EC2, SQS, entre otras.

- **CsvHelper:** Es una librería para lectura y escritura de archivos CSV. Rápida, flexible y fácil de usar.
- **EntityFrameworkCore:** Permite mapear entidades de base de datos a objetos C#.
- **Figgle:** Generador de banners ASCII.
- **FirebaseAdmin:** El SDK de administración es un conjunto de bibliotecas de servidor que permite interactuar con Firebase desde entornos privilegiados para realizar acciones como gestión de usuarios y autenticación.
- **FluentValidation:** Es una librería de .NET que permite crear reglas de validación fuertemente tipadas, de forma sencilla e intuitiva.
- **LinqKit:** Es un conjunto gratuito de extensiones avanzadas de LINQ para SQL y Entity Framework.
- **Mapster:** Mapea un objeto de un tipo específico a un objeto de un tipo diferente de forma sencilla y automática.
- **MediatR:** Soporta solicitudes/respuestas, comandos, consultas, notificaciones y eventos, síncronos y asíncronos con despacho inteligente a través de la variación genérica de C#.
- **Moq:** Es una biblioteca de mocking para .NET que permite crear objetos simulados (mocks) de manera fácil y flexible durante las pruebas unitarias, ayudando a aislar el código bajo prueba y simular comportamientos específicos para mejorar la eficacia de las pruebas.
- **Quartz:** Framework para la creación y gestión de tareas programadas.
- **RazorLight:** Manejador de templates basado en Razor.
- **Serilog:** Framework de Logging para .NET.
- **xUnit:** Es un marco de pruebas unitarias para .NET que permite escribirlas y ejecutarlas de manera eficiente y flexible. Ofrece una sintaxis simple y potente para crear y organizar pruebas, facilitando la detección de errores y el mantenimiento del código.

4.5. Uso del entorno de desarrollo

1. Estando en la raíz del proyecto, ejecute el siguiente comando para compilarlo:

```
Unset

$ dotnet build
```

2. Ejecute la aplicación.

```
Unset

$ dotnet run --project CVU.Admin.Api/CVU.Admin.Api.csproj
```

Esto levantará un servidor web, una vez que esté en funcionamiento, se podrá acceder a la documentación interactiva de la API generada por Swagger en https://localhost:4000/swagger/index.html, esto si el puerto 4000 se encuentra libre o si no se ha especificado otro.

5. Estructura del proyecto

El proyecto sigue una arquitectura basada en capas, lo que ayuda a separar las responsabilidades y promover la modularidad y la reutilización del código. A continuación, se describen los principales directorios y archivos de cada capa del proyecto.

5.1. Estructura de directorios

```
Unset
📁Cvu.Back
└── 📁.docker
    └── .dockerignore
    └── .gitignore
    └── Dockerfile
└── 📁CVU.Admin.Api
    └── 📁Application
        └── 📁Adapters
        └── 📁Commands
        └── 📁Constants
        └── 📁Helpers
        └── 📁Mappers
        └── 📁Queries
        └── 📁Services
            └── 📁AmazonS3
            └── 📁Atenea
            └── 📁Authentication
            └── 📁Catalog
            └── 📁ESignature
            └── 📁Email
            └── 📁Firebase
            └── 📁ScheduledTasks
            └── 📁TravelPlanFare
            └── 📁TravelPlanTransactions
        └── 📁Utils
        └── 📁Validators
        └── 📁Wrappers
```

```
    └── 📁Configuration
        └── 📁Templates
    └── 📁Controllers
        └── 📁V1
            └── AuthController.cs
            └── CatalogController.cs
            └── CvuController.cs
            └── PlaceController.cs
            └── PlanContractController.cs
            └── QuestionsToRateController.cs
            └── TransactionController.cs
            └── TravelPlanController.cs
            └── UserController.cs
    └── 📁Infrastructure
    └── 📁Properties
    └── 📁SeedWork
    └── CVU.Admin.Api.csproj
    └── Program.cs
└── 📁CVU.Admin.Domain
    └── 📁AggregatesModel
    └── 📁Constants
    └── 📁Exception
    └── 📁Models
    └── 📁SeedWork
    └── CVU.Admin.Domain.csproj
└── 📁CVU.Admin.Infrastructure
    └── 📁ConnectedServices
        └── 📁CPAVServiceReference
        └── 📁ClientServiceReference
        └── 📁SaleServiceReference
        └── 📁UtilServiceReference
    └── 📁EntityConfigurations
    └── 📁Finder
    └── 📁Quartz
    └── 📁Repository
    └── AdminContext.cs
    └── CVU.Admin.Infrastructure.csproj
    └── MediatorExtension.cs
└── 📁CVU.Admin.UnitTest
    └── 📁Api
        └── 📁Commands
        └── 📁Queries
        └── 📁Services
    └── CVU.Admin.UnitTest.csproj
    └── GlobalUsings.cs
└── .gitignore
└── build.sh
└── buildspec.yml
└── CVU.Admin.sln
└── README.md
```

5.2. Directorios y archivos de la capa de Web API

- **Application:** En este directorio, se encuentran todos los elementos necesarios para el funcionamiento de la capa correspondiente. El componente principal en esta estructura son los comandos (Commands) y las consultas (Queries), los cuales son utilizados por el controlador (Controller) mediante el uso del patrón Mediator. El Mediator actúa como intermediario para lanzar peticiones a estos componentes, facilitando la comunicación entre el controlador y los comandos/consultas, lo que contribuye a una arquitectura más desacoplada y mantenible. Esta organización permite una clara separación de responsabilidades y promueve la reutilización de código al encapsular la lógica de manejo de solicitudes dentro de comandos y consultas específicas.
    - **Adapters:** En este caso los adaptadores son utilizados para facilitar la comunicación y la integración con otros sistemas. Estos archivos o componentes actúan como intermediarios, permitiendo que la Web API se conecte de manera eficiente con servicios externos, maneje la transformación de datos según sea necesario y garantice una interfaz coherente para la comunicación. Los adaptadores simplifican la integración con servicios de terceros (en este caso Amazon S3), normalizan las interfaces y contribuyen a la flexibilidad y modularidad del sistema al encapsular la lógica específica de la comunicación en un componente dedicado.
    - **Commands:** Los comandos (Commands) son utilizados en este proyecto para representar las acciones que modifican el estado de la aplicación. Estos comandos encapsulan las intenciones del usuario para realizar operaciones de escritura, como la creación, actualización o eliminación de datos. Al separar las operaciones de lectura (queries) de las operaciones de escritura, el patrón CQRS permite una gestión más eficiente y especializada de cada tipo de operación.
    - **Queries:** Las consultas (Queries) son utilizadas en este proyecto para representar las operaciones de lectura que solicitan información sin modificar el estado de la aplicación. Estas consultas están diseñadas para recuperar datos específicos y proporcionan una interfaz especializada para la obtención de información.
    - **Services:** En este directorio están todas las integraciones que se realizaron, podemos ver los subdirectorios de cada servicio

con su interfaz y la implementación de esta, con el fin de poder usar estos servicios en esta capa.

- **AmazonS3:** Servicio que se encarga de cargar y eliminar archivos a un bucket específico.
- **Atenea:** Consumo de las referencias a los web services de Atenea.
- **Authentication:** Manejo de autenticación de un usuario y gestión de códigos TOTP (No utilizados).
- **Catalog:** Encargado de realizar validaciones a los catálogos necesarios para el proyecto, como los tipos de documentos de identificación.
- **Email:** Encargado del envío de correos electrónicos.
- **ESignature:** A través de la API de click&sign se realiza el proceso de firma digital de los nuevos contratos adquiridos por un cliente.
- **Firebase:** Gestión de usuarios y de cuenta.
- **ScheduledTasks:** Se definen las tareas que luego se programarán para su ejecución, como enviar notificaciones o eliminar planes por vencimiento de tiempo.
- **TravelPlanFare:** Métodos relacionados con las tarifas y precios correspondientes a un plan.
- **TravelPlanTransactions:** Servicio encargado de gestionar los aportes y avances de un plan.

- **Configuration:** En este directorio están todos los archivos necesarios para la configuración de los diferentes servicios y base de datos, en donde podemos encontrar las credenciales necesarias para poder acceder sin problemas a cada servicio, además se encuentra la configuración básica del proyecto.
  - **Templates:** En este directorio se encuentran las plantillas necesarias en diferentes procesos, por ejemplo, las plantillas de cada uno de los mensajes de los correos electrónicos que se enviarán a los usuarios.
- **Controllers:** Esta carpeta contiene los controladores de la API, que manejan las solicitudes de los clientes y orquestan las operaciones correspondientes.

- **Infrastructure:** En esta sección, se lleva a cabo la configuración de autofac, para facilitar la inyección de dependencias, en este caso para los repositorios, finders y servicios. El propósito principal de este proceso es habilitar el acceso a estos componentes en los comandos y consultas dentro de la capa de Web API. La inyección de dependencias es una práctica que permite que las instancias necesarias de los repositories, finders y servicios sean proporcionadas a los commands y queries que los requieran. También aquí se realiza la configuración y programación de los CronJobs, y se define el motor de renderización de plantillas.
- **Program.cs:** En este archivo se configura la ejecución del proyecto, es aquí donde se van a definir los servicios con sus credenciales, también donde se realizará la conexión con la base de datos, además de otros ajustes que se consideren necesarios.

5.3. Directorios y archivos de la capa de Dominio

- **Aggregates:** En esta carpeta se definen los Aggregates, que representan las entidades principales del dominio y encapsulan la lógica de negocio y las reglas de validación, además se definen las interfaces de los Finders y Repositories.
- **Constants:** Se definen las constantes que contienen valores relacionados con las reglas de negocio. Estas nos ayudan a manejar definiciones centralizadas, asegurando la atomicidad de cualquier cambio en estas definiciones.
- **Exception:** Se definen las clases auxiliares para el manejo de excepciones dentro de la aplicación,
- **Models:** En este directorio se encuentran todas las clases usadas como DTOs y algunas otras auxiliares, las cuales nos ayudan con un manejo fácil y eficiente de los datos a través de la aplicación.
- **SeedWork:** Contiene interfaces las cuales proporcionan métodos que serán reutilizados en varios lugares, esto con el fin de reutilizar código y mejorar la legibilidad de este.

5.4. Directorios y archivos de la capa de Infraestructura

- **ConnectedServices:** Referencias a servicios de Atenea (Cliente, CPAV, Venta y Util).

- **EntityConfigurations:** Esta carpeta almacena la configuración y mapeo de las tablas a las clases que se encuentran en Aggregates, esto con el fin de poder realizar las transacciones a las bases de datos sin inconvenientes y de forma fácil.
- **Finders:** Esta carpeta contiene las implementaciones de las interfaces definidas en la capa de dominio, en el apartado de Aggregates.
- **Repositorios:** Aquí se encuentran las implementaciones de las interfaces de los repositorios definidos en Aggregates..
- **AdminContext:** Este archivo es esencial ya que es donde se configura Entity Framework, el cual nos ayuda con todo el tema transaccional con la base de datos, este define las configuraciones de las entidades.

5.5. Capa de test unitarios

En esta capa se definen las pruebas unitarias para los comandos, consultas y servicios.

Para ejecutar los test unitarios, desde el directorio CVU.Admin.UnitTest utilice el siguiente comando:

```
Unset

$ dotnet test
```

5.6. Archivos de configuración

- **appsettings.json:** Este archivo almacena la configuración general de la aplicación, como cadenas de conexión, claves de API, configuración de servicios externos, etc.
- **appsettings.{environment}.json:** Estos archivos contienen configuraciones específicas para cada ambiente (por ejemplo, desarrollo, QA, producción), lo que permite una configuración diferenciada según el entorno.
- **cvu-firebase-keys.{environment}.json:** Contiene las llaves y parámetros de configuración necesarios para utilizar Firebase Admin SDK. Según el proyecto de firebase correspondiente a cada ambiente.
- **launchSettings.json:** Este archivo define la configuración de depuración y lanzamiento de la aplicación.

6. Guía de implementación

En este apartado se hablará sobre las consideraciones importantes para implementar el proyecto o cuando se requiera agregar alguna funcionalidad adicional.

6.1. Implementación del proyecto de Web API

Para asegurar un desarrollo fluido del proyecto, es imprescindible contar con la instalación de .NET 6 y las herramientas necesarias, las cuales fueron detalladas en la sección de Requerimientos.

El proyecto sigue una estructura organizativa que comprende las capas de Web API, Domain e Infrastructure, tal como se describe en la sección de Estructura del Proyecto.

Es fundamental instalar las dependencias necesarias mediante el administrador de paquetes NuGet de .NET. Esto permite la utilización de los métodos requeridos en el proyecto. Es importante destacar que cada integración con servicios externos posee sus propias dependencias, por lo que se recomienda revisar la documentación correspondiente para una implementación efectiva y seguir las instrucciones proporcionadas.

6.2. Definición de los Aggregates y Events en la capa de Dominio

En esta etapa, se procede a identificar las entidades principales del dominio y a crear los Agregados correspondientes en la capa de dominio. Además, se agregan las interfaces de Finder y/o Repository para su posterior implementación en la capa de infraestructura.

Paralelamente, se definen los eventos que representan los sucesos ocurridos en el dominio y que capturan los cambios de estado en los Agregados. Estos eventos son fundamentales para mantener la integridad y coherencia del modelo de dominio a lo largo del tiempo y en diversas situaciones.

6.3. Implementación de los Controllers, Commands y Queries en la capa de Web API

En la capa de Web API, se lleva a cabo la creación de controladores encargados de gestionar las solicitudes provenientes de los clientes, siguiendo las prácticas recomendadas de diseño Restful. Durante este proceso, se incorpora el patrón Mediator para administrar las peticiones hacia los Comandos y Consultas que serán definidos.

Cada controlador actúa como un punto de entrada, utilizando el Mediator para orquestar las operaciones específicas mediante comandos y consultas. Esta implementación fomenta la separación de responsabilidades y facilita la extensibilidad, ya que los controladores se enfocan en la interacción con el cliente, mientras delegan la lógica de negocio y el acceso a datos a través del Mediator y sus respectivos Comandos y Consultas.

Este enfoque contribuye a una arquitectura modular y mantenible en la capa de Web API. Posteriormente, se procede a la implementación de los Comandos y Consultas para las operaciones de modificación de datos. Para ello, se emplea Entity Framework, cuya configuración se detallará en el apartado correspondiente a la configuración con la base de datos, para facilitar la interacción con la misma.

6.4. Implementación de servicios externos

Cuando se considera la implementación de un nuevo servicio externo, es crucial tener en cuenta que cada servicio puede presentar sus propias especificidades. Por lo tanto, su integración se deja a criterio del desarrollador. Sin embargo, en términos generales, se sigue manteniendo el principio de inyección de dependencias.

Para ello, se crean interfaces e implementaciones correspondientes a estos servicios externos. Este enfoque no solo facilita la modularidad y la extensibilidad del sistema, sino que también promueve una mayor claridad en el código y una mejor gestión de las dependencias.

6.5. Consideraciones generales

1. **Configuración de Entity Framework:** Es fundamental configurar Entity Framework, el cual se encarga de mapear las entidades del dominio con la base de datos, permitiendo así realizar transacciones de manera eficiente.

2. **Implementación de Interfaces en la Capa de Infraestructura:** Para mantener la arquitectura CQRS, es necesario implementar las interfaces definidas en los Agregados del dominio en la capa de infraestructura. Estas implementaciones concretas interactúan con el almacenamiento persistente, como bases de datos, ejecutando las operaciones definidas en los Agregados del dominio. Es importante destacar que la configuración de Entity Framework se realiza en la capa de infraestructura.

3. **Uso de DTOs (Objetos de Transferencia de Datos):** Se deben agregar DTOs que faciliten el manejo de la información en la aplicación. Estos objetos deben diseñarse según las necesidades específicas de la aplicación y contener únicamente la información necesaria para cada operación.

4. **Manejo de Excepciones:** Es recomendable crear clases dedicadas para el manejo de excepciones en el proyecto. Estas clases permiten encapsular la lógica del manejo de errores, lo que proporciona claridad y facilita el mantenimiento del código.

5. **Gestión de Credenciales:** Las credenciales necesarias para acceder a los diferentes servicios deben agregarse en la configuración dentro de la capa de Web API. Dado que es desde esta capa que se realizará el acceso a los servicios externos, la configuración de las credenciales en este nivel garantiza una gestión centralizada y segura de la información.

7. Configuración de la base de datos

En esta sección, se detalla la configuración de Entity Framework, un paso crucial para mapear las clases del dominio a las tablas de bases de datos. Este proceso permite aprovechar las capacidades de Entity Framework para realizar transacciones de manera más sencilla y eficiente.

7.1. Implementación de Entity Framework

Como se explicó anteriormente, la adición de los EntityConfigurations en la capa de infraestructura es esencial. Estos archivos sirven para mapear los parámetros de las clases a las columnas de cada tabla, construir las relaciones, y demás configuraciones que apliquen. Deben ser creados para cada entidad que se desee manejar en la base de datos. Una vez configuradas estas entidades, se procede a implementar el DbContext. Esta interfaz en .NET es fundamental, ya que nos permite definir los mapeos previamente configurados y las entidades creadas en el proyecto. Además, el DbContext proporciona varios métodos útiles en la misma interfaz, permitiendo modificar y obtener información de cada tabla mapeada. En resumen, este enfoque facilita la integración efectiva entre la lógica del dominio y la persistencia de datos, siguiendo buenas prácticas de diseño y asegurando una gestión eficiente de la base de datos. Por último, cabe mencionar que cuando se quiere adicionar una entidad se debe realizar el procedimiento realizado anteriormente, este luego de haber creado la tabla en la base de datos.

7.2. Configuración de Entity Framework

Una vez completada la implementación del apartado anterior, la configuración final del DbContext se realiza en el archivo Program.cs de la

aplicación, ubicado en la capa de Web API. En este archivo es donde se configura el DbContext para que la aplicación pueda interactuar con la base de datos. Es en este mismo contexto donde se especifica la conexión con la base de datos mediante la cadena de conexión. Esta configuración centralizada en el archivo Program.cs asegura que la aplicación esté correctamente vinculada con la base de datos

7.3. Consideraciones adicionales

La base de datos utilizada en este proyecto es Postgresql, en este caso alojada en AWS, se recomienda utilizar un administrador de bases de datos como pgAdmin o DataGrip, los cuales facilitan la creación de tablas y modificación de estas mismas.

8. Manejo de errores y excepciones

La gestión de errores y excepciones en el código del proyecto se aborda con diligencia, implementando prácticas que aseguran un manejo adecuado de situaciones imprevistas. Esto incluye la captura y registro de errores, la provisión de mensajes claros y apropiados para los usuarios, y la adopción de medidas correctivas.

Se emplean estructuras de control de excepciones para prevenir bloqueos o comportamientos incorrectos del sistema ante eventos inesperados. Además, se garantiza que los mensajes de error no revelen información sensible o crítica sobre el sistema, salvaguardando detalles como configuración o estructura interna. Esta atención cuidadosa a la gestión de errores contribuye a la robustez y confiabilidad del sistema, asegurando una experiencia de usuario fluida y segura.

Además se implementa un filtro que intercepta las excepciones de tipo BaseException, y retornan un código de error http según corresponda, más el mensaje de error con el formato de respuesta definido.

Las excepciones se definen en la capa de dominio, en el directorio Exception, mientras que el filtro se encuentra en la capa Web Api, directorio Infrastructure > Exceptions.

9. Mantenimiento y actualizaciones

En esta sección, se abordarán los aspectos relacionados con el mantenimiento continuo del proyecto y las actualizaciones necesarias para garantizar su rendimiento, seguridad y compatibilidad con las últimas

tecnologías. A continuación, se detallan los principales puntos a considerar en este proceso.

9.1. Proceso de mantenimiento y soporte

- Establece un proceso de mantenimiento regular que incluya tareas de monitoreo, respaldo de datos, ajustes de rendimiento y solución de problemas.
- Define los roles y responsabilidades del equipo encargado del mantenimiento, asegurando una comunicación efectiva y una respuesta oportuna a las necesidades del sistema.
- Documenta los procedimientos de mantenimiento, incluyendo las acciones a realizar, las frecuencias recomendadas y los pasos para solucionar problemas comunes.

9.2. Actualizaciones del framework y dependencias

- Mantén actualizado el framework utilizado en el proyecto, como .NET, siguiendo las recomendaciones del proveedor y aplicando las actualizaciones correspondientes.
- Realiza un seguimiento de las dependencias utilizadas en el proyecto, como bibliotecas y paquetes de terceros, y mantén actualizadas sus versiones para aprovechar las mejoras y correcciones de errores.
- Realiza pruebas exhaustivas después de cada actualización para asegurarte de que el sistema siga funcionando correctamente y de que no se hayan introducido incompatibilidades o problemas de rendimiento.

10. Recursos adicionales

10.1. Referencias para tener en cuenta

- DDD - CQRS: https://learn.microsoft.com/es-es/dotnet/architecture/microservices/microservice-ddd-cqrs-patterns/
- CQRS: https://learn.microsoft.com/en-us/azure/architecture/patterns/cqrs

- .NET: https://dotnet.microsoft.com/
- Entity Framework: https://learn.microsoft.com/en-us/ef/
- Swagger: https://swagger.io/docs/
- AWS: https://docs.aws.amazon.com/
- Click&Sign: https://api.lleida.net/dtd/clickandsign/v1/es/index.html

10.2. Documentación de los endpoints

Este proyecto está integrado con Swagger que es una herramienta poderosa para diseñar, documentar y probar APIs. Proporciona una manera estandarizada y eficiente de crear documentación interactiva, generar código cliente y realizar pruebas automatizadas. Con Swagger, puedes mejorar la usabilidad y adopción de tu API al ofrecer una experiencia de desarrollo más fácil y accesible para los consumidores de la API.

En el siguiente enlace encontrarás la documentación de los endpoints de todo el proyecto con sus respectivos datos de entrada y de salida, separados por los controladores. Solo se puede acceder cuando se esté ejecutando el proyecto en modo desarrollo.

https://dev.back.cvu.com.co/swagger/index.html

10.3. Credenciales, accesos y otros

10.3.1. Base de datos

- Host: cvudev-instance-1.ciyuj5kxgbi3.us-east-1.rds.amazonaws.com
- Username: postgres
- Password: Cvu2024*
- Port: 5432

10.3.2. Atenea

- userName: cvuws
- password: cVu8gOoZ9oOzIWS

10.3.3. Lleida

- User: circulodeviajes@co
- Password: .b224_E}FV

10.3.4. Click&Sign

JSON de ejemplo para creación de configuración a través del endpoint https://api.lleida.net/cs/v1/set_config

JavaScript

```
{
  "request": "SET_CONFIG",
  "request_id": "set-config-req",
  "user": "circulodeviajes@co",
  "config": {
    "name": "Firma contrato test v2",
    "expire_lapse": 216,
    "auto_cancel": "Y",
    "default_sms_sender": "Ahorro%CVU",
    "default_email_from_name": "CVU - Círculo de Viajes Universal",
    "signature_cb_url":
"https://dev.back.cvu.com.co/api/ms-admin/v1/TravelPlan/contract-signature/s
tatus",
    "registered_company_name": "Círculo de Viajes Universal S.A.",
    "registered_company_vat_number": "860029002-1",
    "registered_langs": "ES",
    "lang": "ES",
    "color_background": "",
    "color_text": "",
    "color_button_background": "",
    "color_button_text": "",
    "logo": "",
    "signatory_tags": [
      "customer_name",
      "number_contract",
      "customer_surname",
      "customer_type_id",
      "customer_number_id",
      "number_input",
      "value_input",
      "total_saving",
      "sign_email",
      "sign_mobile",
      "encrypted_mail",
      "encrypted_mobile"
    ],
    "sms": [
      {
        "reminder_lapse": "0",
```

```
        "text": "#customer_name# tu Clave Temporal es #otp# para realizar la
firma de tu Contrato de Ahorro #number_contract# con CVU. Más info 350 734
6611",
        "recipient": "SIGNATORY",
        "type": "otp",
        "registered": "Y",
        "sender": "Ahorro%CVU"
      }
    ],
    "email": [
      {
        "bcc": "",
        "reminder_lapse": "72.0",
        "type": "reminder",
        "to": "SIGNATORY",
        "body_free_text": "",
        "registered": "N",
        "cc": "",
        "subject": "#customer_name# te invitamos a firmar tu Contrato de
Ahorro CVU #number_contract#",
        "attachment_file_group": [],
        "from_name": "CVU - Círculo de Viajes Universal"
      },
      {
        "bcc": "",
        "reminder_lapse": "0",
        "type": "signed",
        "to": "SIGNATORY",
        "body_free_text": "",
        "registered": "N",
        "cc": "",
        "subject": "#customer_name# tu contrato con CVU #number_contract# se
ha firmado",
        "attachment_file_group": ["SIGNATORY_STAMP"],
        "from_name": "CVU - Círculo de Viajes Universal"
      }
    ],
    "landing": {
      "decline_reason": "not_required",
      "enable_button": "always",
      "landing_access_otp": "N",
      "signature_type": "on_sign",
      "declinable_signature": "N",
      "display_signatories": "N",
      "default_upload_files": [],
      "default_upload_files_registered": "Y",
      "default_url_redirect": "",
      "landing_access_code": "Y",
      "landing_access_max_retries": 50,
      "url_redirect_delay": "7",
      "signature_on_sign_required_elements": {
        "handwritten": "Y",
        "simple": "N",
        "typed": "N",
        "accepted_pki": [],
```

```
        "allow_multiple_signature_methods": "N",
        "otp_sending": "on_landing",
        "otp_length": 4,
        "otp": "Y",
        "otp_max_retries": 2
      },
      "landing_access_pki": "N",
      "landing_access_accepted_pki": []
    }
  }
}
```

DOUBLE V PARTNERS

Manual técnico

[CVU][Frontend][WEB]

Elaborado por Double V Partners: Equipo de desarrollo

2024

Tabla de contenido

Introducción

El presente manual técnico ofrece una guía sobre la estructura y desarrollo del proyecto CVU. Este documento está diseñado para proporcionar a los desarrolladores y demás miembros del equipo una referencia completa que les permita comprender y trabajar de manera eficiente con el código del proyecto.

En este manual, se explorará en detalle la arquitectura del proyecto, destacando sus componentes fundamentales, estructura de directorios, patrones de diseño utilizados y buenas prácticas de programación implementadas. Además, se proporcionarán instrucciones claras para la instalación, configuración y mantenimiento del sistema, con el objetivo de garantizar su eficiencia y escalabilidad a lo largo del tiempo.

Objetivo del manual técnico

El objetivo principal de este manual es proporcionar una referencia completa y detallada sobre la estructura y desarrollo del proyecto. Este documento tiene como finalidad facilitar la comprensión y el trabajo eficiente con el código del proyecto, tanto para los desarrolladores como para otros miembros del equipo involucrados en su implementación y mantenimiento.

Audiencia

Este manual está destinado a ser utilizado por los siguientes grupos de personas:

Desarrolladores:

● Aquellos responsables de construir y mejorar el software. Necesitan una comprensión profunda de la estructura del proyecto, las tecnologías utilizadas y los estándares de codificación para contribuir eficazmente al desarrollo del proyecto.

● Administradores del sistema:
Los encargados de configurar y mantener el entorno de producción, asegurando que la aplicación se despliegue y funcione correctamente. Este manual les proporcionará información crucial sobre los requisitos del sistema, la configuración del entorno y las tareas de mantenimiento necesarias para garantizar un funcionamiento óptimo del sistema.

● Otros miembros del equipo: Cualquier persona que participe en el desarrollo, despliegue o mantenimiento del proyecto y que requiera una visión general y detalles técnicos sobre el mismo.

Estructura del manual técnico

Este manual técnico sigue una estructura organizada en diversas secciones temáticas que cubren distintos aspectos del proyecto. A continuación, se presenta una descripción general de cada sección:

● Requisitos del sistema:
Enumera los requisitos técnicos y de infraestructura necesarios para ejecutar el software de manera efectiva en distintas plataformas (Windows, macOS).

● Arquitectura y estructura del proyecto:
Describe la arquitectura general del proyecto, incluyendo los componentes principales, la estructura de directorios y los patrones de diseño utilizados en la implementación.

● Tecnologías utilizadas:
Detalla las tecnologías, frameworks y librerías empleadas en el desarrollo del proyecto, proporcionando información sobre su propósito y funcionalidad.

● Instalación y configuración:
Proporciona instrucciones detalladas y paso a paso sobre cómo configurar el entorno de desarrollo y llevar a cabo la instalación inicial del software en los diferentes sistemas operativos compatibles.

● Desarrollo y uso del software:

Explica cómo utilizar las funcionalidades principales del software y cómo interactuar con la interfaz de usuario, ofreciendo ejemplos prácticos y guías de uso.

● Despliegue y mantenimiento:
Describe los procesos y consideraciones relacionados con el despliegue inicial y el mantenimiento continuo del sistema en producción.

● Glosario de términos:
Incluye una lista de los términos técnicos y conceptos clave utilizados en el proyecto, junto con sus definiciones, para facilitar la comprensión del contenido técnico del manual.

1. Requisitos del Sistema

1.1 Requisitos del Sistema para Angular (Frontend)

Para la instalación y desarrollo del proyecto en Angular, el entorno de desarrollo debe cumplir con los siguientes requisitos mínimos en las plataformas Windows, Linux o macOS.

Windows

- Sistema operativo:
  - Windows 10 o posterior (64 bits), basado en x86-64.
- Espacio en disco:
  - Al menos 10 GB de espacio libre en disco
- Herramientas: Es necesario contar con las siguientes herramientas en el entorno:
  - Windows PowerShell 5.0 o superior (preinstalado en Windows 10).
  - Node.js (última versión estable).
  - npm (Node Package Manager).
  - Git for Windows 2.x, con la opción "Usar Git desde el símbolo del sistema de Windows".

Linux

- Sistema operativo:
  Linux (64 bits).
- Espacio en disco:

    - Se requieren al menos 600MB de espacio libre (excluyendo el espacio para IDE/herramientas).
- Herramientas: Es necesario contar con las siguientes herramientas en el entorno:
    - bash
    - curl
    - git 2.x
    - Node.js (última versión estable).
    - npm (Node Package Manager).

Mac

- Sistema operativo:
    - macOS, versión 10.14 (Mojave) o posterior.
- Espacio en disco:
    - Se requieren al menos 2 GB de espacio libre
- Herramientas:
    - Xcode Command Line Tools (se pueden instalar a través de la terminal con xcode-select --install).
    - Node.js (última versión estable).
    - npm (Node Package Manager).

Es fundamental asegurarse de que el entorno cumpla con estos requisitos mínimos antes de iniciar el desarrollo del proyecto en cada plataforma respectiva. Además, se recomienda mantener actualizadas todas las herramientas y dependencias necesarias para el desarrollo continuo del proyecto.

2. Arquitectura y estructura del proyecto

El proyecto sigue una arquitectura basada en tres capas fundamentales: la Capa de Presentación, la Capa de Dominio y la Capa de Datos. Esta arquitectura se inspira en los principios de Clean Architecture, que promueven la separación clara de responsabilidades y la escalabilidad del sistema.

Capa de Presentación:

- Esta capa se encarga de la interfaz de usuario y la interacción con el usuario final.
- Incluye componentes de Angular como componentes, directivas y servicios relacionados con la presentación de la interfaz de usuario.
- Aquí se definen las vistas, la lógica de presentación y la interacción del usuario con la aplicación.

Capa de Dominio:

- En esta capa se encuentran las reglas de negocio y la lógica de la aplicación.
- Contiene los servicios, modelos y lógica de negocio que definen cómo funcionan las diferentes funcionalidades de la aplicación.

Capa de Datos:

- Esta capa se encarga de acceder y gestionar los datos de la aplicación.
- Incluye servicios y utilidades para interactuar con fuentes de datos externas, como APIs REST, bases de datos u otros sistemas.
- Aquí se implementa la lógica para recuperar, almacenar y manipular los datos necesarios para el funcionamiento de la plataforma.

2.1 Estructura de carpetas y archivos principales

El proyecto se estructuró de manera organizada separando las características principales de los demás componentes. A continuación se detalla la estructura principal:

- **src/**: Es la carpeta principal que contiene el código fuente del proyecto.
- **src/assets/**: En está carpeta se encuentran todos los archivos multimedia utilizados en el proyecto.
- **src/data**: Contiene la estructura de los modelos, repositorios y data sources que serán usados en sus correspondientes features.
- **src/domain**: Contiene la estructura de las entidades, repositorios casos de usos entre otros que serán usados en sus correspondientes features.
- **src/presentation**: Contiene todas las vistas que representan las diferentes features de la aplicación. Dentro de cada feature contiene los componentes específicos del módulo.
- **src/environments**: Carpeta que contiene los archivos de configuración de entorno, como los archivos de configuración para desarrollo, producción.
- **src/presentation/app**: Acá se encuentran las características o funcionalidades principales de la aplicación.

- **src/presentation/app/app.module.ts:** Archivo que define el módulo principal de la aplicación.
- **src/presentation/app/shared**: Contiene componentes, directivas, pipes y módulos compartidos entre diferentes partes de la aplicación.
- **src/presentation/app/shared/services:** Contiene los archivos con los diferentes servicios como almacenamiento, tema e inicio de sesión entre otros.
- **src/presentation/app/shared/guards:** Contiene los guards de ruta utilizados para la seguridad de la aplicación.
- **src/presentation/app/shared/interceptors:** Contiene los interceptores HTTP utilizados para modificar las peticiones HTTP.
- **src/presentation/app/shared/pipes:** Contiene los pipes personalizados utilizados en la aplicación.
- **src/presentation/app/shared/directives:** Contiene las directivas personalizadas utilizadas en la aplicación.
- **src/presentation/app/app-routing.module.ts**: Archivo que define las rutas de navegación de la aplicación.
- **src/styles.css**: Contiene los archivos de estilos globales.

2.2 Funcionalidades

En la arquitectura que se maneja en el proyecto las funcionalidades principales de la aplicación se separan y cada una de estás tiene su propia estructura de Clean Architect. Las funcionalidades desarrolladas se definen a continuación:

i. **Landing**: Pantalla informativa.
Sus principales funciones:
   1. Guiar a los usuarios a través de los pasos para iniciar su plan de ahorro
   2. Mostrar información sobre los beneficios de iniciar un plan de ahorro.
   3. Redirigir al usuario a la simulación de plan de ahorro

ii. **Login**: Los usuario pueden realizar el inicio con su usuario (email) y contraseña

iii. **Registro**: Permite a los nuevos usuarios crear una cuenta.
Sus principales funciones:
   - Formulario de registro de nuevos usuarios.
   - Creación de nueva cuenta de usuario.

iv. **Recuperar contraseña**: Permite a los usuarios recuperar su contraseña en caso de olvido.
Sus principales funciones:
   - Envío de correo electrónico de recuperación.

- Restablecimiento de la contraseña del usuario.

v. **Home**: Vista principal donde se muestra un resumen de los planes de ahorro del usuario, tanto iniciados como en borradores.
Sus principales funciones:
   - Mostrar resumen de planes de ahorro
   - Navegación a detalles de cada plan.
   - Navegación a simulación de planes

vi. **Lista de Planes**: Muestra una lista de los planes de ahorro del usuario, categorizados como activos, borradores y finalizados.
Sus principales funciones:
   - Visualización de la lista de planes activos.
   - Visualización de la lista de planes en borrador.
   - Visualización de la lista de planes finalizados.
   - Navegación a los detalles de cada plan.

vii. **Detalles del Plan**: Proporciona una vista detallada de un plan de ahorro específico
Sus principales funciones:
   - Mostrar detalles completos del plan de ahorro.
   - Opciones para editar plan borrador

viii. **Simular Ahorro**: Permite a los usuarios simular un plan de ahorro seleccionando varias opciones.
Sus principales funciones:
   - Selección de opciones como días, destino y cantidad de personas.
   - Cálculo y simulación de un plan de ahorro basado en las opciones seleccionadas.

ix. **Resultados de Simulación**: Muestra los resultados de la simulación de ahorro, proporcionando opciones de planes de ahorro disponibles.
Sus principales funciones:
   - Mostrar resultados de la simulación.
   - Comparar diferentes opciones de planes de ahorro.

x. **Iniciar Plan**: Proporciona un flujo completo para activar un nuevo plan de ahorro.
Pantallas del flujo:
   - Formulario de datos para iniciar un plan.
   - Resumen de pago.
   - Proceso de pago
   - Firma de contrato.
   - Calificación asesor

xi. **Realizar aporte**: Proporciona un flujo para realizar un aporte a un plan de ahorro.
Pantallas del flujo:
- Resumen de pago.
- Proceso de pago

xii. **Certificaciones**: Proporciona opciones para descargar varios documentos, como el estado de cuenta, declaración de renta comercial y documentos para la embajada americana.

xiii. **Solicitud de Reingreso**: Permite a los usuarios solicitar el uso de su ahorro con agencias de viajes que no tienen convenio con la aplicación.
Sus principales funciones:
- Formulario de solicitud de reingreso.
- Validación y envío de solicitud.

xiv. **Cerrar Sesión:** Permite a los usuarios cerrar su sesión de manera segura.
Sus principales funciones:
- Cerrar la sesión del usuario.
- Redirección a la pantalla de inicio de sesión.

3. Instalación y Configuración

El proceso de instalación de Angular es muy similar en los sistemas operativos Windows, macOS y Linux. Sin embargo, sigue las instrucciones específicas de tu sistema para obtener los mejores resultados.

1. Instalar Node.js y npm
   Antes de poder instalar Angular, necesitarás tener Node.js y npm (Node Package Manager) instalados en tu máquina.
2. Instalar Angular CLI Globalmente
   A continuación, tendrás que instalar la herramienta Angular CLI. Para instalar la CLI de Angular globalmente en tu sistema, abre el símbolo del sistema y ejecuta el siguiente comando:

```
Unset
npm install -g @angular/cli
```

Este comando instala la última versión estable de la herramienta CLI de

Angular y hace que esté disponible para su uso en todo tu sistema.

3. Ejecutar los Comandos de la CLI de Angular
   Una vez que tengas instalada la CLI de Angular, puedes utilizar sus comandos para gestionar tus proyectos de Angular. Para comprobar que la instalación se ha realizado correctamente, ejecuta el siguiente comando en el símbolo del sistema:

```
Unset
ng --version
```

Este comando muestra la versión instalada de la CLI de Angular, junto con otra información relevante sobre tu entorno.

3.2 Clonación y configuración del repositorio

1. Abra una terminal o línea de comandos y navegue hasta el directorio donde desea clonar el repositorio.
2. Abre sesión en el servicio de CodeCommit AWS o ingresa al siguiente link https://us-east-1.console.aws.amazon.com/codesuite/codecommit/repositories/Cvu.Front/browse?region=us-east-1 .
3. En la página principal de CodeCommit, seleccione el repositorio que desea clonar.
4. Haga clic en el botón "Clonar URL HTTP" o "Clonar URL SSH", dependiendo de la opción que desee utilizar para la autenticación.
5. Copie la URL del repositorio proporcionada.
6. Vuelva a la terminal o línea de comandos y ejecute el siguiente comando para clonar el repositorio utilizando la URL que copió en el paso anterior: Para clonar utilizando HTTPS:

   $ git clone url_clone

7. Se le pedirá que proporcione sus credenciales de Git para la autenticación. Ingrese sus credenciales de Git cuando se le solicite.
8. Una vez completada la clonación, el repositorio estará disponible localmente en su sistema.

3.3 Configuración de dependencias y librerías adicionales

En el desarrollo de una aplicación Angular, se pueden utilizar diversas dependencias y librerías adicionales para ampliar las funcionalidades y mejorar la experiencia del usuario. A continuación, se presenta una breve descripción de cada una de las dependencias mencionadas:

- **@angular/fire:** Una integración de Angular con Firebase que simplifica la creación de aplicaciones web en tiempo real, almacenamiento en la nube, autenticación de usuarios y mucho más.

- **@ngrx/effects:** Esta librería proporciona una forma de manejar efectos secundarios en las aplicaciones Angular utilizando la arquitectura Redux.

- **@ngrx/store:** Implementa un patrón de administración de estado inspirado en Redux para Angular, lo que facilita el manejo del estado de la aplicación de manera predecible y eficiente.

- **html-to-image:** Una librería que permite convertir elementos HTML en imágenes, útil para generar capturas de pantalla o imágenes de elementos de la interfaz de usuario.

- **ng2-pdf-viewer:** Un visor de PDF para Angular que permite la visualización de archivos PDF dentro de una aplicación web Angular.

- **rxjs:** Una biblioteca para programación reactiva en JavaScript que proporciona un conjunto de operadores para trabajar con secuencias de datos asíncronas.

4. Uso del desarrollo

4.1 Ejecución del proyecto

En esta sección, se detallan los pasos necesarios para ejecutar el proyecto en el entorno de desarrollo. Se requiere haber completado la configuración previa antes de seguir estos pasos. A continuación, se describen los pasos para ejecutar el proyecto.

1. **Seleccionar la versión de Angular:** Se recomienda utilizar la versión 15 de Angular.
2. **Construir el proyecto**: Ejecuta el comando npm install en la terminal para descargar todas las dependencias y paquetes necesarios para el proyecto.
3. **Ejecutar el servidor de desarrollo:** Después de instalar las dependencias, puedes ejecutar el servidor de desarrollo de Angular utilizando el comando ng serve en la terminal. Esto iniciará el servidor y tu aplicación estará disponible en http://localhost:4200/, donde podrás verla en tu navegador web.

4.2 Estructura y organización de la capa de datos

En la arquitectura de Clean Architecture, los Data Sources son componentes que actúan como intermediarios entre la capa de dominio y las fuentes de datos externas, como APIs o bases de datos. Su función principal es proporcionar una abstracción que facilite el acceso y manipulación de los datos, permitiendo a la capa de dominio mantenerse independiente de las implementaciones específicas de las fuentes de datos. Los Data Sources garantizan una separación clara de responsabilidades y promueven la reutilización y la estabilidad en el desarrollo de aplicaciones. A continuación se listan los data source utilizados en el desarrollo de la aplicación.

- **Auth**
  - Descripción: Este módulo gestiona la autenticación de usuarios y la recuperación de contraseñas.
  - Métodos clave:
    - **login**

Url	https://dev.back.cvu.com.co/api/ms-admin/v1/Auth/login
Método http	POST

body	{
	"email": "string",
	"password": "string"
	}

    - **refresh-token**

Url	https://dev.back.cvu.com.co/api/ms-admin/v1/Auth/refresh-token
Método http	POST

body	{
	"refreshToken": "string"
	}

    - **forgot-password**

Url	https://dev.back.cvu.com.co/api/ms-admin/v1/Auth/fo

	rgot-password
Método http	POST

body	{
	"email": "string"
	}

- **register**

Url	https://dev.back.cvu.com.co/api/ms-admin/v1/User
Método http	POST

body	{
	"email": "string",
	"password": "string"
	}

- **Catalog**
  - Descripción:
  - Métodos clave:
    - **getCatalogDocumentTypes**

Url	https://dev.back.cvu.com.co/api/ms-admin/v1/Catalog/document-types
Método http	GET

- **getCatalogContributionPrices**

Url	https://dev.back.cvu.com.co/api/ms-admin/v1/Catalog/contribution-prices
Método http	GET

- **getCataloAdvancesAvailable**

Url	https://dev.back.cvu.com.co/api/ms-admin/v1/Catalog/advances-available/savings-plan/{savingsPlanId}
Método http	GET

- **getCatalogContributionAvailableForSubscription**

Url	https://dev.back.cvu.com.co/api/ms-admin/v1/Catalog/cvu-payment/contributions-available-for-subscription
Método http	GET

- **getCatalogContributionAvailableForSContribution**

Url	https://dev.back.cvu.com.co/api/ms-admin/v1/Catalog/cvu-payment/contributions-available-for-contribution
Método http	GET

- **Simulation**
  - Descripción: Este módulo maneja la simulación de planes de viaje, incluyendo la obtención de lugares y la calculación de planes de viaje.
  - Métodos clave:
    - **getPlaces**

Url	https://dev.back.cvu.com.co/api/ms-admin/v1/Place
Método http	GET

    - **getPlaces**

Url	https://dev.back.cvu.com.co/api/ms-admin/v1/TravelPlan/calculate
Método http	POST

body	{
	"origin": {
	"latitude": number,
	"longitude": number
	},
	"destinationId": "string",
	"numberDays": number,
	"numberPeople": number
	}

- **getAvailableDays**

Url	https://dev.back.cvu.com.co/api/ms-admin/v1/Place/{destinationId}/available-days
Método http	GET

- **PlanContract**
  - Descripción: Este módulo gestiona los planes, incluyendo la obtención de planes, resúmenes de pagos, contribuciones y transacciones.
  - Métodos clave:
    - **getPlan**

Url	https://dev.back.cvu.com.co/api/ms-admin/v1/PlanContract/{id}
Método http	GET

    - **getPlans**

Url	https://dev.back.cvu.com.co/api/ms-admin/v1/PlanContract/state/{state}
Método http	GET

    - **getPaymentSummary**

Url	https://dev.back.cvu.com.co/api/ms-admin/v1/PlanContract//{planId}/Transaction/contribution/next

Método http	GET

- **postTransactionContribution**

Url	https://dev.back.cvu.com.co/api/ms-admin/v1/PlanContract//{planId/Transaction/contribution
Método http	POST

body	{
	"amount": 0,
	"numberOfContributionsToPay": 0,
	"paymentMethod": "string",
	"transactionCode": "string",
	"cvuPaymentPlansDebits": [
	{
	"filing": "string",
	"value": 0
	}
	]
	}

- **getPlansSummary**

Url	https://dev.back.cvu.com.co/api/ms-admin/v1/PlanContract/{planId}/Transaction
Método http	GET

- **getTransactionsList**

Url	https://dev.back.cvu.com.co/api/ms-admin/v1/PlanContract/cvu-payment/available-for-subscription
Método http	GET

- **getAvailableContractForSubscription**

Url	https://dev.back.cvu.com.co/api/ms-admin/v1/PlanContract/cvu-payment/available-for-subscription
Método http	GET

- **getAvailableContractForContribution**

Url	https://dev.back.cvu.com.co/api/ms-admin/v1/PlanContract/cvu-payment/available-for-contribution
Método http	GET

- **TravelPlan**
  - Descripción: Este módulo gestiona los planes borradores, como la eliminación, edición y además tenemos un servicio para firma de contratos de suscripción.
  - Métodos clave:
    - **deletePlan**:

Url	https://dev.back.cvu.com.co/api/ms-admin/v1/TravelPlan/{id}
Método http	DELETE

    - **patchEditPlan**:

Url	https://dev.back.cvu.com.co/api/ms-admin/v1/TravelPlan/{id}
Método http	PATCH

    - **signContract**:

Url	https://dev.back.cvu.com.co/api/ms-admin/v1/TravelPlan/{idPlan}/subscribe/sign-contract/start-signature

Método http	POST

- **User**
  - Descripción: Este módulo gestiona la información y las operaciones relacionadas con los usuarios, incluyendo la obtención y actualización de datos.
  - Métodos clave:
    - **getCertificationByType**:

Url	https://dev.back.cvu.com.co/api/ms-admin/v1/User/{idUser}/certification/{typeCertification}
Método http	GET

    - **getUser**:

Url	https://dev.back.cvu.com.co/api/ms-admin/v1/User/{idUser}
Método http	GET

    - **putUser**:

Url	https://dev.back.cvu.com.co/api/ms-admin/v1/User/{idUser}
Método http	PUT
body	{ "firstName": "string", "lastName": "string", "birthdate": "string", "gender": "string", "documentTypeId": "string", "documentNumber": "string", "department": "string", "municipality": "string", "phoneNumber": "string" }

- **postUserFirebase**:

Url	https://dev.back.cvu.com.co/api/ms-admin/v1/User/firebase
Método http	POST

body	{
	"email": "string",
	"firebaseUid": "string",
	"firstName": "string",
	"lastName": "string",
	"phoneNumber": "string"
	}

- **getUserFirebase**:

Url	https://dev.back.cvu.com.co/api/ms-admin/v1/User//firebase/{firebaseUid}
Método http	GET

- **postRefundRequest**:

Url	https://dev.back.cvu.com.co/api/ms-admin/v1/User/{userID}/refund-request
Método http	POST
Parámetros Form-Data	'refundForm' (File): El formulario de reembolso adjunto al usuario. 'description' (string): Una descripción opcional de la solicitud de reembolso.

- **SavingsPlan**
  - Descripción: Este módulo se encarga de la gestión de planes de ahorro, incluyendo la activación de planes, pagos y la calificación de los mismos.
  - Métodos clave:
    - **getDpto**:

Url	https://www.datos.gov.co/resource/gdxc-w37w.json?$select=distinct%20cod_depto,dpto&$where=dpto%20is%20not%20null
Método http	GET

- **getMpioByCodeDpto**:

Url	https://www.datos.gov.co/resource/gdxc-w37w.json?$select=cod_mpio,nom_mpio,tipo_municipio&cod_depto={code}
Método http	GET

- **createSavingsPlan**:

Url	https://dev.back.cvu.com.co/api/ms-admin/v1/TravelPlan/{idPlan}/subscribe
Método http	POST

body	{
	"savingsPlanId": "string",
	"firstName": "string",
	"lastName": "string",
	"birthdate": "string",
	"gender": "string",
	"documentTypeId": "string",
	"documentNumber": "string",
	"residenceDepartment": "string",
	"residenceMunicipality": "string",
	"phoneNumber": "string",
	"contribution": {
	"amount": 0,
	"numberOfContributionsToPay": 0,
	"paymentMethod": "string",
	"transactionCode": "string",
	"cvuPaymentPlansDebits": [
	{
	"filing": "string",
	"value": 0
	}
	]
	}
	}

- **postRatingPlan**:

Url	https://dev.back.cvu.com.co/api/ms-admin/v1/TravelPlan/{idPlan}/subscribe/rate
Método http	POST

body	{
	"advisor": "string",
	"questions": [
	{
	"questionId": "string",
	"answer": "string"
	}
	]
	}

- **getQuestionRating**:

Url	https://dev.back.cvu.com.co/api/ms-admin/v1/QuestionsToRate
Método http	GET

- **postRatingPlan**:

Url	https://dev.back.cvu.com.co/api/ms-admin/v1/TravelPlan/{idPlan}/subscribe/rate
Método http	POST

body	{
	"advisor": "string",
	"questions": [
	{
	"questionId": "string",
	"answer": "string"
	}
	]
	}

5. Despliegue y Mantenimiento

5.1 Despliegue

Consulta los documentos adjuntos titulados "FUNCIONAMIENTO DE IMPLEMENTACIÓN DE CI/CD EN CVU" e "INFORME TÉCNICO DE INFRAESTRUCTURA DE CVU" para obtener una guía detallada sobre la infraestructura de la aplicación en la nube de AWS y el funcionamiento de la implementación de CI/CD en CVU.
Estos documentos proporcionan información valiosa sobre los componentes utilizados, los procesos de compilación, despliegue y automatización.

5.2 Mantenimiento

- **Actualización de Dependencias:**
  Regularmente, revisa y actualiza las dependencias de tu proyecto Angular, incluyendo Angular CLI, bibliotecas externas, etc.
  Utiliza herramientas como npm outdated para identificar las dependencias desactualizadas y npm update para actualizarlas.

- **Monitoreo y Registro:**
  Implementa sistemas de monitoreo para estar al tanto de la salud y el rendimiento de tu aplicación Angular en producción.
  Configura el registro de errores y eventos para poder detectar y solucionar problemas rápidamente.

- **Respaldo y Recuperación:**
  Realiza copias de seguridad regulares de tus datos y archivos críticos relacionados con el proyecto Angular.
  Ten un plan de recuperación en caso de que ocurra un fallo grave o una pérdida de datos.
- **Optimización del Rendimiento:**
  Monitorea el rendimiento de tu aplicación Angular y realiza ajustes según sea necesario para mejorar la velocidad de carga y la experiencia del usuario.
  Optimiza los activos estáticos, como imágenes y archivos CSS/JS, para reducir los tiempos de carga.

- **Gestión de Versiones:**
  Utiliza un sistema de control de versiones como Git para gestionar y mantener un historial de cambios en tu proyecto Angular.
  Implementa prácticas de ramificación y fusión para gestionar el desarrollo de nuevas funciones y correcciones de errores de manera eficiente.

6. Glosario de términos

- **Angular:** Framework de desarrollo web de código abierto, mantenido por Google, que permite construir aplicaciones web dinámicas y de alta calidad utilizando TypeScript.
- **Component (Componente):** Unidad básica de la interfaz de usuario en Angular, que incluye una plantilla HTML, lógica de control en TypeScript, y estilos CSS asociados.
- **Directive (Directiva):** Instrucción que se aplica a un elemento del DOM para modificar su comportamiento o apariencia. Las directivas en Angular pueden ser estructurales (como *ngIf y *ngFor) o de atributo (como ngClass y ngStyle).
- **Interceptor:** Servicio que implementa el patrón de diseño interceptor, utilizado para modificar solicitudes y respuestas HTTP globalmente antes de que lleguen a la aplicación o al servidor.
- **Module (Módulo):** Contenedor que agrupa componentes, directivas, pipes y servicios relacionados, organizando el código en unidades funcionales. El módulo raíz es AppModule.
- **NgModule:** Decorador utilizado para definir un módulo en Angular. Especifica las declaraciones, importaciones, exportaciones, proveedores y componentes bootstrap de un módulo.
- **Pipe (Tubería):** Función que transforma datos en plantillas Angular. Los pipes pueden ser utilizados para formatear cadenas, números, fechas, y otros datos en las vistas.
- **Reactive Forms:** Modelo de formularios en Angular que utiliza programación reactiva para gestionar la validación y el estado del formulario. Proporciona mayor control y flexibilidad en comparación con los formularios basados en templates.
- **Router (Enrutador):** Módulo que gestiona la navegación y las rutas dentro de una aplicación Angular. Permite definir rutas, asociar componentes a rutas y gestionar la navegación entre diferentes vistas de la aplicación.
- **RxJS:** Biblioteca para la programación reactiva que proporciona observables, operadores y herramientas para manejar eventos asíncronos y flujos de datos. Angular utiliza RxJS para manejar operaciones asíncronas como solicitudes HTTP.
- **API**: Interfaz de Programación de Aplicaciones (Application Programming Interface, en inglés) que define los métodos y protocolos utilizados para la comunicación entre componentes de software. Las APIs permiten la

interacción entre distintas aplicaciones y servicios.

- **Base de datos**: Una base de datos es un sistema organizado para almacenar y gestionar grandes cantidades de datos. Se utiliza para almacenar y recuperar información de manera eficiente, y puede ser utilizado por las aplicaciones para almacenar y gestionar datos persistentes.
- **Caché**: La caché es una técnica utilizada para almacenar temporalmente datos en un lugar más rápido y accesible, con el objetivo de acelerar la recuperación de información. En el contexto de las aplicaciones, se utiliza para almacenar datos previamente obtenidos, como imágenes o respuestas de API, para evitar la necesidad de volver a descargarlos y mejorar el rendimiento de la aplicación.
- **Casos de uso:** Los casos de uso son escenarios específicos o funcionalidades que una aplicación puede realizar. Representan las acciones o interacciones que los usuarios pueden realizar en la aplicación para lograr un objetivo determinado.
- **Clase**: En programación orientada a objetos, una clase es una plantilla o prototipo que define las propiedades y comportamientos de un objeto. Las clases se utilizan para crear objetos y encapsular datos y funcionalidades relacionadas.
- **Clean Architecture: Metodología** de diseño de software que busca separar las preocupaciones en capas y mantener una estructura modular y desacoplada. Clean Architecture promueve la separación de la lógica de negocio de los detalles de implementación técnica.
- **Código fuente**: Conjunto de instrucciones escritas en un lenguaje de programación que constituyen un programa de software. El código fuente es editable y puede ser compilado o interpretado para generar un programa ejecutable.
- **Controlador**: En el contexto de una aplicación, un controlador es una clase o componente responsable de manejar la lógica y el flujo de datos relacionados con una determinada funcionalidad o vista.
- **CRUD**: CRUD es un acrónimo que representa las operaciones básicas de gestión de datos en un sistema: Create (crear), Read (leer), Update (actualizar) y Delete (eliminar). Se utiliza para describir las operaciones estándar que se pueden realizar en una base de datos u otras fuentes de datos.
- **DataSource**: En el contexto de la capa de datos de una aplicación, un DataSource es un componente encargado de proporcionar y gestionar los datos utilizados por la aplicación. Puede interactuar con diferentes fuentes de datos, como bases de datos, APIs o sistemas externos.
- **Dominio**: En el desarrollo de software, el dominio se refiere al ámbito o área de conocimiento específica en la cual se encuentra una aplicación. Representa el problema o la temática principal que la aplicación resuelve.
- **Feature**: En el contexto del desarrollo de software, una feature (característica) se refiere a una funcionalidad específica o conjunto de funcionalidades que se agregan a una aplicación para proporcionar un valor adicional o cumplir con un requisito específico. Puede incluir cambios en la interfaz de usuario, mejoras en el rendimiento, nuevas capacidades, entre otros.
- **Framework**: Conjunto de herramientas, librerías y utilidades que

proporcionan una estructura para el desarrollo de aplicaciones. Los frameworks permiten agilizar el proceso de desarrollo al proporcionar funcionalidades comunes y una arquitectura predefinida.

- **Git**: Sistema de control de versiones distribuido ampliamente utilizado para el seguimiento de cambios en el código fuente de proyectos de desarrollo de software. Git permite la colaboración entre desarrolladores, la gestión de ramas y la reversión de cambios, entre otras funcionalidades.
- **HTTP**: HTTP (Hypertext Transfer Protocol) es un protocolo utilizado para la comunicación entre clientes y servidores en la web. Permite la transferencia de información, como solicitudes y respuestas, entre un navegador web y un servidor, facilitando la visualización de páginas web y la interacción con aplicaciones en línea.
- **Interfaz**: En programación, una interfaz es un conjunto de métodos y/o propiedades que define un contrato para la comunicación entre diferentes componentes. Sirve como una especificación de cómo interactuar con un objeto sin preocuparse por su implementación concreta.
- **Inyección de dependencias**: Patrón de diseño y técnica que permite separar la creación y gestión de objetos de su utilización. La inyección de dependencias se utiliza para facilitar la modularidad, la reutilización y la prueba unitaria del código.
- **Librería**: Conjunto de código predefinido y reutilizable que proporciona funcionalidades específicas para ser utilizadas en el desarrollo de aplicaciones. Las librerías permiten ahorrar tiempo y esfuerzo al utilizar código ya existente en lugar de tener que escribirlo desde cero. Material: Material es un conjunto de directrices y principios de diseño desarrollados por Google para la creación de interfaces de usuario coherentes en aplicaciones. Estas directrices se basan en el uso de elementos de diseño, como colores, tipografía y disposición de elementos, para proporcionar una experiencia visualmente agradable y consistente.
- **Modelo**: En el desarrollo de software, un modelo es una representación estructurada de los datos y reglas de negocio de una aplicación. Los modelos se utilizan para almacenar y manipular los datos de la aplicación de manera coherente y organizada.
- **Repositorio**: En el contexto de la arquitectura de software, un repositorio es un componente encargado de interactuar con los DataSources y proporcionar una capa de abstracción para acceder a los datos. Gestiona la obtención, creación, actualización y eliminación de datos, y proporciona una interfaz para que los otros componentes de la aplicación accedan a ellos.
- **Script**: En programación, un script es un conjunto de instrucciones o comandos que se ejecutan de manera secuencial para realizar una tarea o función específica. Los scripts suelen ser escritos en un lenguaje de programación y pueden ser utilizados para automatizar tareas, realizar cálculos o manipular datos.
- **Servicio**: En el desarrollo de software, un servicio es un componente que encapsula una funcionalidad específica y se utiliza para llevar a cabo tareas o procesos específicos. Puede involucrar operaciones complejas, interacciones con sistemas externos o lógica de negocio especializada.

- **Terminal**: El terminal es una interfaz de línea de comandos en la que los usuarios pueden interactuar con un sistema operativo mediante la ejecución de comandos. Permite realizar diversas tareas, como la navegación por el sistema de archivos, la compilación de programas y la administración de recursos.
- **URL**: Una URL (Uniform Resource Locator) es una dirección que se utiliza para identificar de manera única la ubicación de un recurso en Internet. Se compone de diferentes componentes, como el protocolo, el nombre de dominio, la ruta y los parámetros, y permite acceder a recursos como páginas web, archivos o servicios en línea.
- **Web**: En el contexto de desarrollo de aplicaciones, el término "web" se refiere a la plataforma y tecnologías utilizadas para desarrollar aplicaciones que se ejecutan en navegadores web. Estas aplicaciones son accesibles a través de Internet y no requieren una instalación específica en el dispositivo del usuario.

DOUBLE V PARTNERS

FUNCIONAMIENTO DE IMPLEMENTACIÓN DE CI/CD EN CVU

Elaborado por:
Equipo DVP

índice

Introducción

El informe proporciona una guía detalla sobre el funcionamiento de la implementación de Continuous Integration/Continuous Deployment (CI/CD) en el proyecto CVU. CVU ha adoptado una arquitectura de microservicios para su aplicación, y utiliza una combinación de servicios de AWS CodeBuild, AWS CodePipeline y Amplify para automatizar el proceso de desarrollo, integración y despliegue de sus microservicios en un cluster de Ecs fargate y Amplify.

Panorama General

La implementación de CI/CD en CVU se basa en la automatización de los procesos de desarrollo, integración y despliegue de sus microservicios. Este enfoque se adopta para garantizar una entrega de software eficiente, confiable y consistente.

Componentes de la Implementación CI/CD

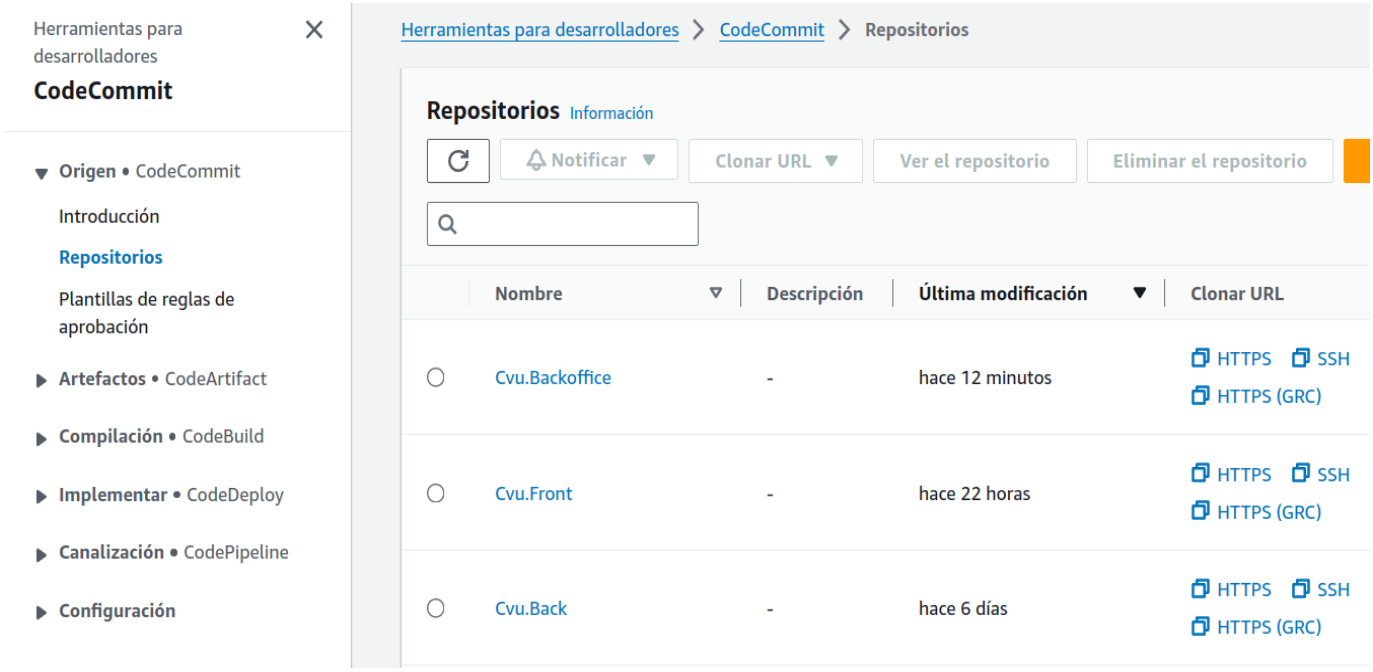
AWS CodeBuild

AWS CodeBuild se utiliza para compilar, probar y preparar los microservicios para su despliegue. Cada microservicio tiene su propia configuración de compilación en CodeBuild, donde se realizan acciones como pruebas unitarias, integración y generación de artefactos necesarios para el despliegue. Estos artefactos pueden incluir imágenes de contenedor, paquetes de aplicaciones u otros recursos necesarios para la ejecución del microservicio en los ambientes asignados

- Implementación:

Ingresamos al servicio de AWS CodeCommit en la opción de repositorios.

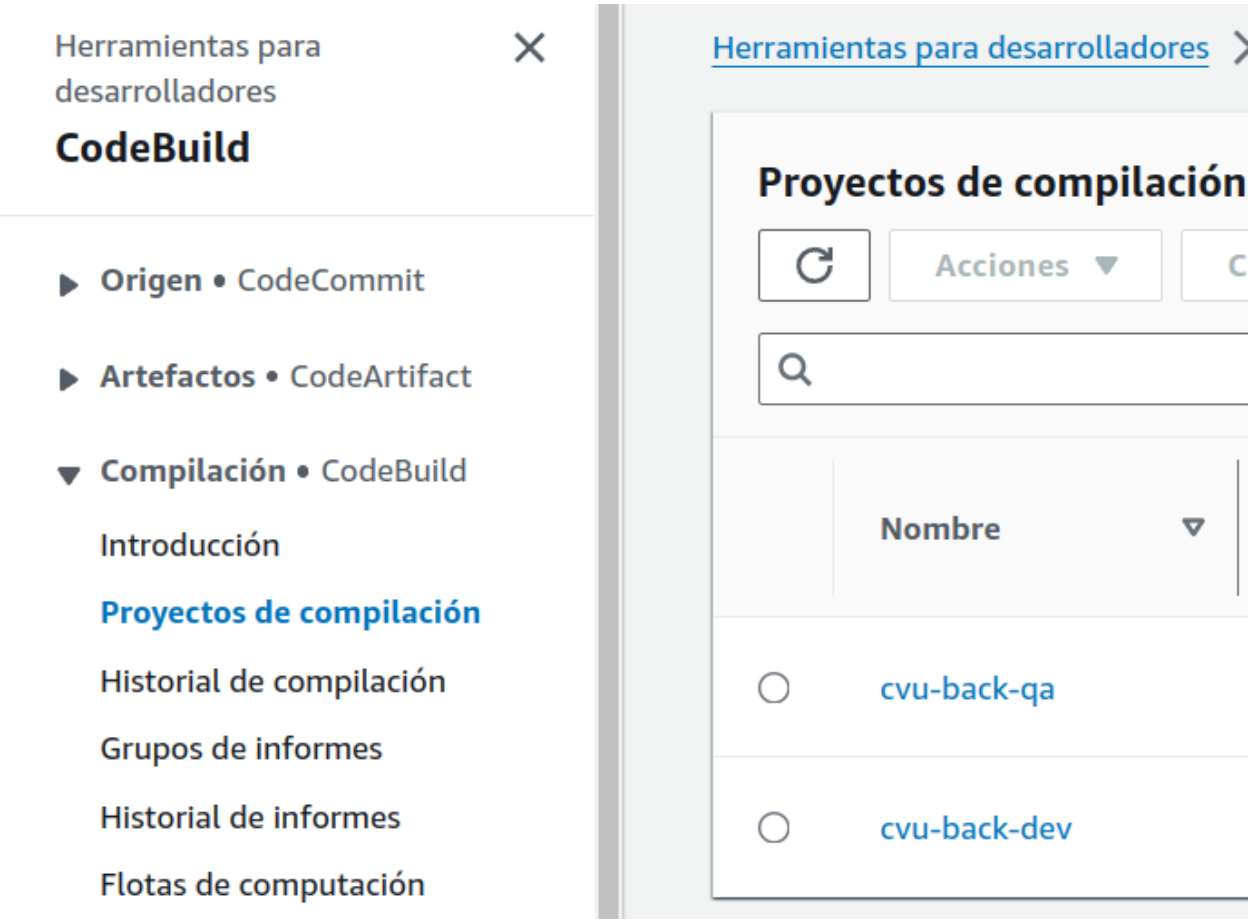
En este ejemplo vamos a utilizar el repositorio de CVU.backoffice Pero se puede utilizar cualquier repositorio



- Ingresa y crear el proceso en el servicio de Codebuild

Para acceder a CodeBuild, se puede utilizar el buscador de servicios de AWS o en efecto cuando ingresas a alguna de las opciones de CodeCommit, CodeDeploy o CodePipeline se despliegan las demás opciones disponibles dentro del servicio.

Ingrese a Compilación CodeBuild (Compilation CodeBuild) y se despliegan las opciones que tiene el servicio de codebuild



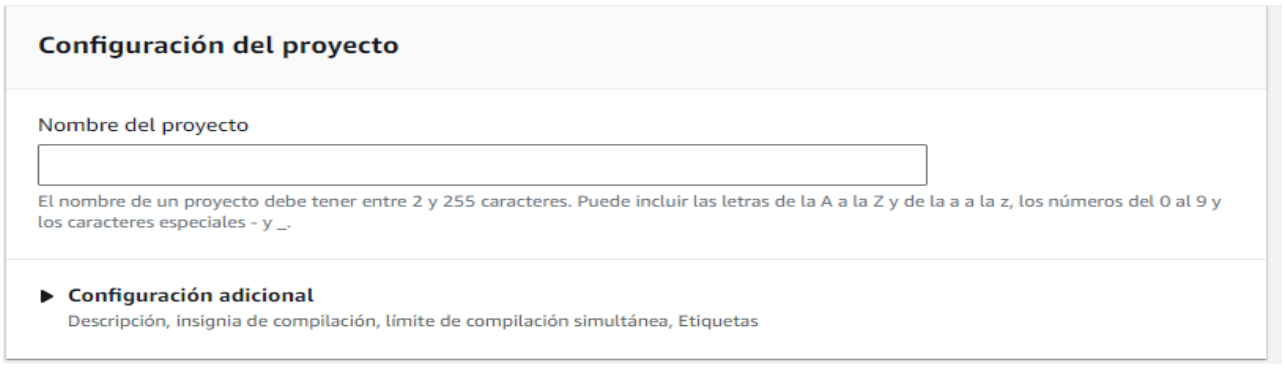
Seleccione o Ingrese a proyectos de compilación

En esta sección, se presenta un panorama completo de los proyectos ya integrados con AWS CodeBuild, resaltando su configuración específica en el entorno. Esta visualización permite un examen detallado de la estructura y los parámetros de compilación de cada proyecto, brindando una comprensión completa de cómo se ejecuta el proceso de construcción en el contexto de CI/CD. Además, tenemos la capacidad de crear y configurar nuevos proyectos según nuestras necesidades específicas, definiendo parámetros de construcción, pasos de compilación y pruebas, así como lógica personalizada. Esto nos brinda un control total sobre nuestro flujo de trabajo de CI/CD.

Ejemplo de cómo crear el proyecto de compilación

Cuando creamos un nuevo proyecto de compilación en AWS CodeBuild, nos encontramos con un proceso organizado por etapas que nos guía en la configuración detallada del proyecto. Este proceso se inicia con la creación de un nuevo proyecto desde cero, lo que nos permite personalizar cada aspecto de la configuración de CodeBuild para satisfacer las necesidades específicas del proyecto.

- Definición de Proyecto: En esta etapa se define un proyecto de CodeBuild, especificando parámetros como el entorno de compilación, la ubicación del código fuente, las acciones de compilación y pruebas a ejecutar, y las opciones de salida, como la generación de artefactos o la integración con otros servicios de AWS.



—

Origen

Agregar el origen

Origen 1: principal

Proveedor del origen

AWS CodeCommit

Repositorio

Cvu.Backoffice

Tipo de referencia

Seleccione el tipo de referencia de la versión de origen que contiene el código fuente.

- (•) Ramificación
- ( ) Etiqueta de Git
- ( ) ID de confirmación

Ramificación

Elija una ramificación que contenga el código para crear.

develop

ID de confirmación - *opcional*

Elija una ID de la confirmación. Esto puede reducir la duración de la compilación.

Versión de origen Información

refs/heads/develop

463e5902 ajuste-sonar

▶ **Configuración adicional**

Profundidad del clon de Git, Submódulos de Git

Entorno

Modelo de aprovisionamiento Información

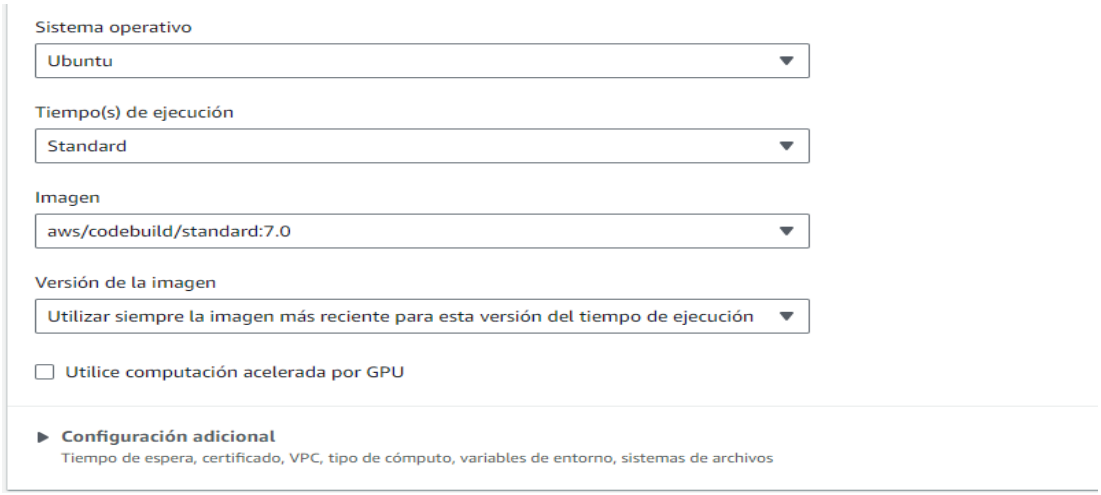
- (•) Bajo demanda
  Aprovisione automáticamente la infraestructura de compilación en respuesta a las nuevas compilaciones.
- ( ) Capacidad reservada
  Utilice una flota de instancias dedicada para las compilaciones. El tipo de entorno e informática de flotas se utilizará para el proyecto.

Imagen del entorno

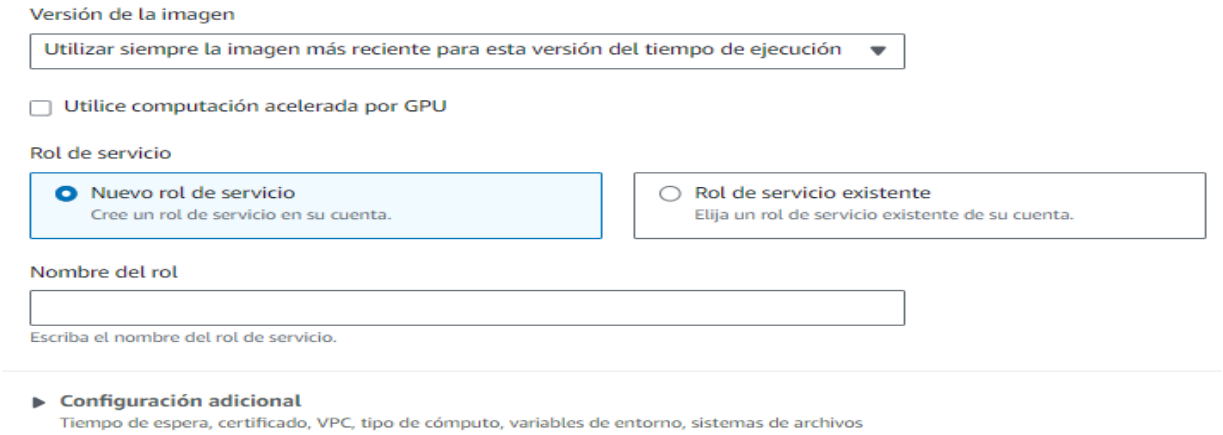
- (•) Imagen administrada
  Utilizar una imagen administrada por AWS CodeBuild
- ( ) Imagen personalizada
  Especificar una imagen de Docker

Computación

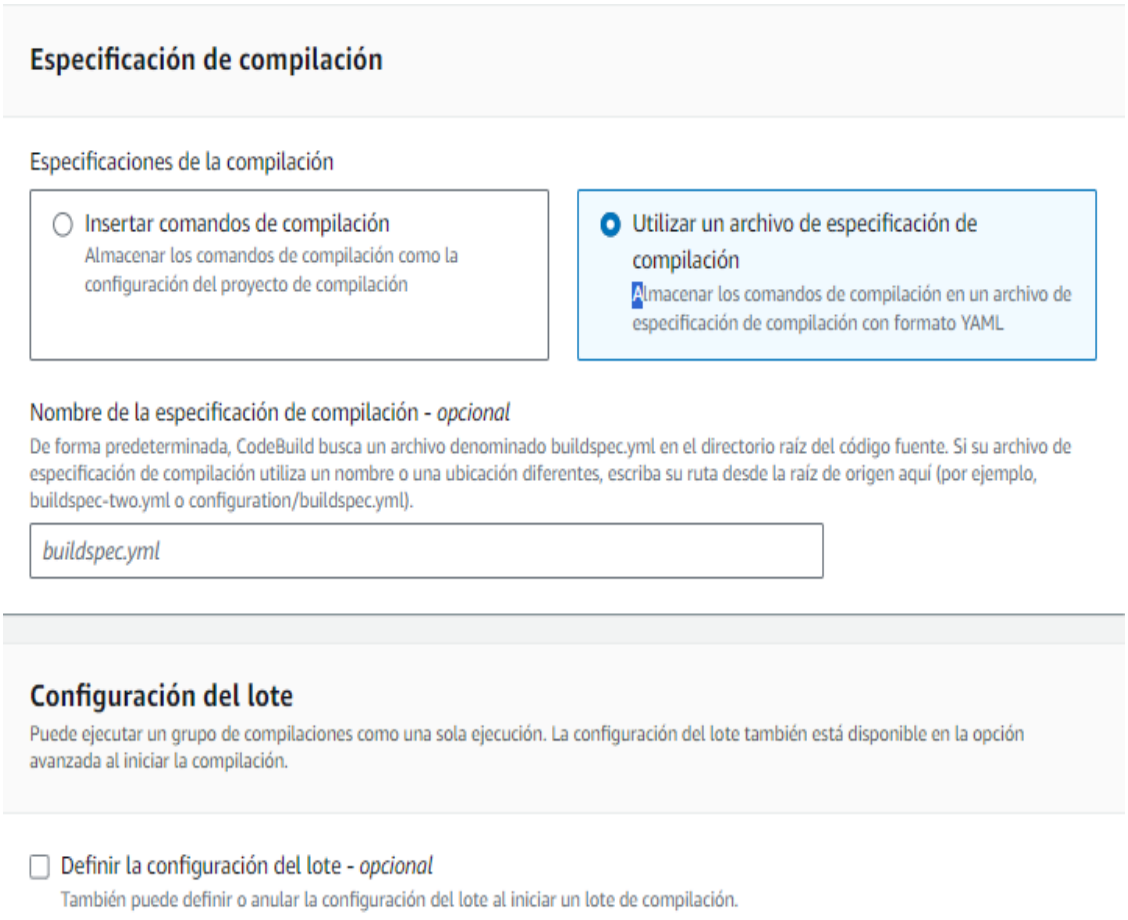
- (•) EC2
  Optimizado para obtener flexibilidad durante las ejecuciones de acciones
- ( ) Lambda
  Optimizado para mayor velocidad y minimiza el tiempo de inicio de las acciones de flujo de trabajo



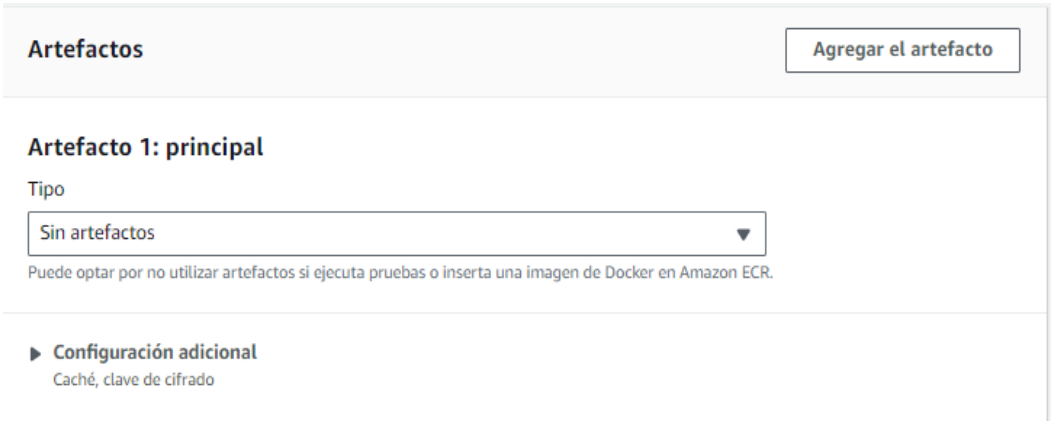
- Configuración de Roles IAM: Se definen los roles de IAM (Identity and Access Management) necesarios para que CodeBuild pueda acceder a los recursos requeridos durante el proceso de compilación, como el acceso al código fuente, a los servicios de almacenamiento y a otras API de AWS.



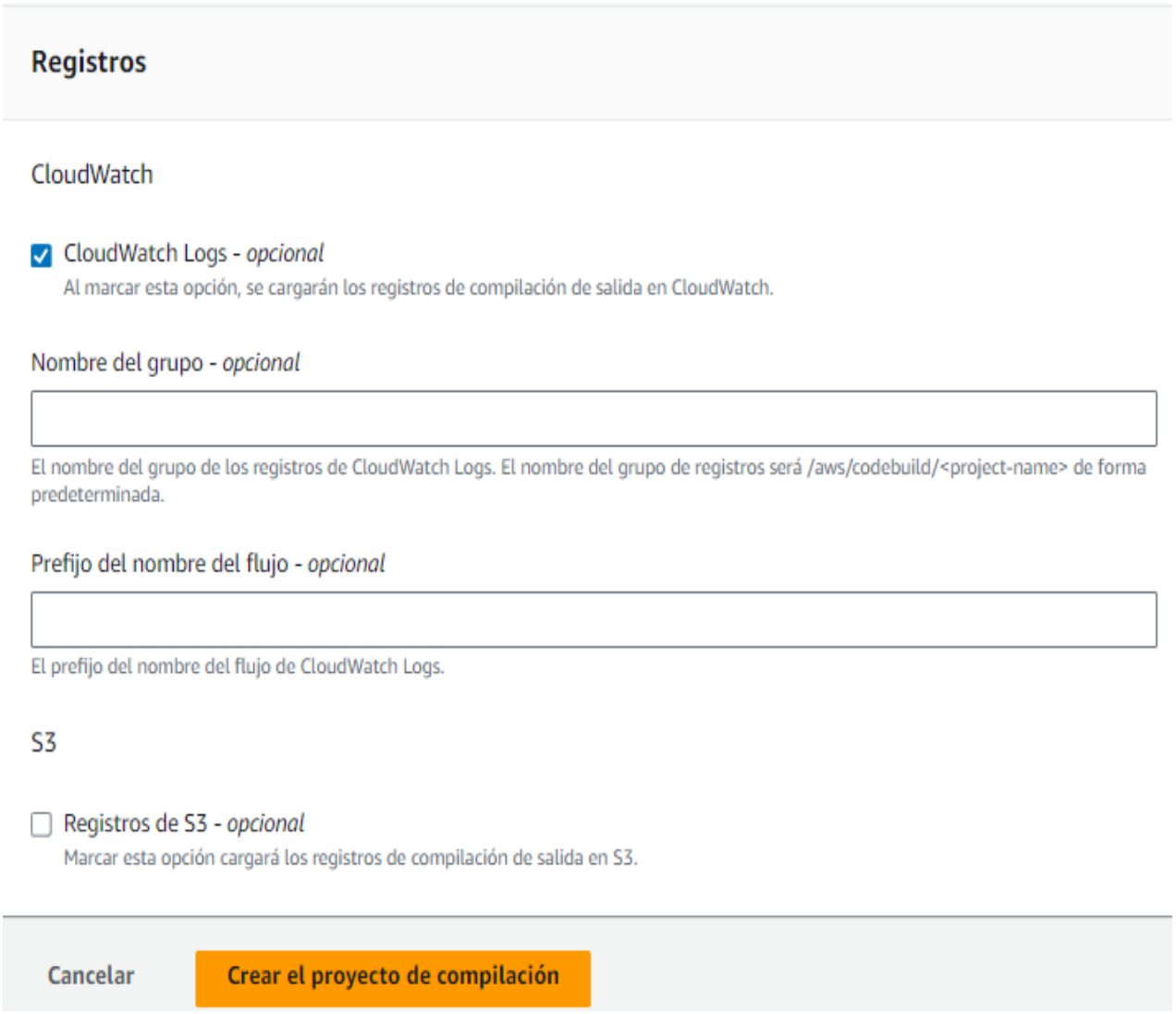
- Etapas de Compilación y Pruebas: Se definen las etapas de compilación y pruebas que se ejecutarán en el proyecto. Esto puede incluir la instalación de dependencias, la compilación de código, la ejecución de pruebas unitarias, pruebas de integración, análisis estático de código, entre otros.



- Gestión de Artefactos: Se configura la forma en que se gestionarán los artefactos generados durante el proceso de compilación. Esto puede incluir la especificación de la ubicación de almacenamiento de artefactos, el formato de salida (como archivos ZIP o imágenes de Docker), y la integración con otros servicios de AWS, como S3 o CodePipeline.



En nuestro entorno de AWS CodeBuild, es importante destacar que no se requiere el ingreso manual de artefactos durante el proceso de configuración



Nota: debemos configurar el buildspect.yaml para seguir con este paso

- Configuración del buildspect.yaml

Procederemos a configurar el formato de creación del nuevo proyecto que estará vinculado al servicio de CodeBuild. Es imperativo tener en cuenta que en nuestro proyecto debe estar presente el archivo buildspec.yaml para iniciar la configuración del servicio. Este archivo desempeña un papel crucial en el proceso, ya que es donde especificamos las directivas y comandos necesarios para la ejecución del flujo de trabajo de integración continua en el proyecto.

Ejemplo:

Cvu.Back / buildspec.yml Información — Editar

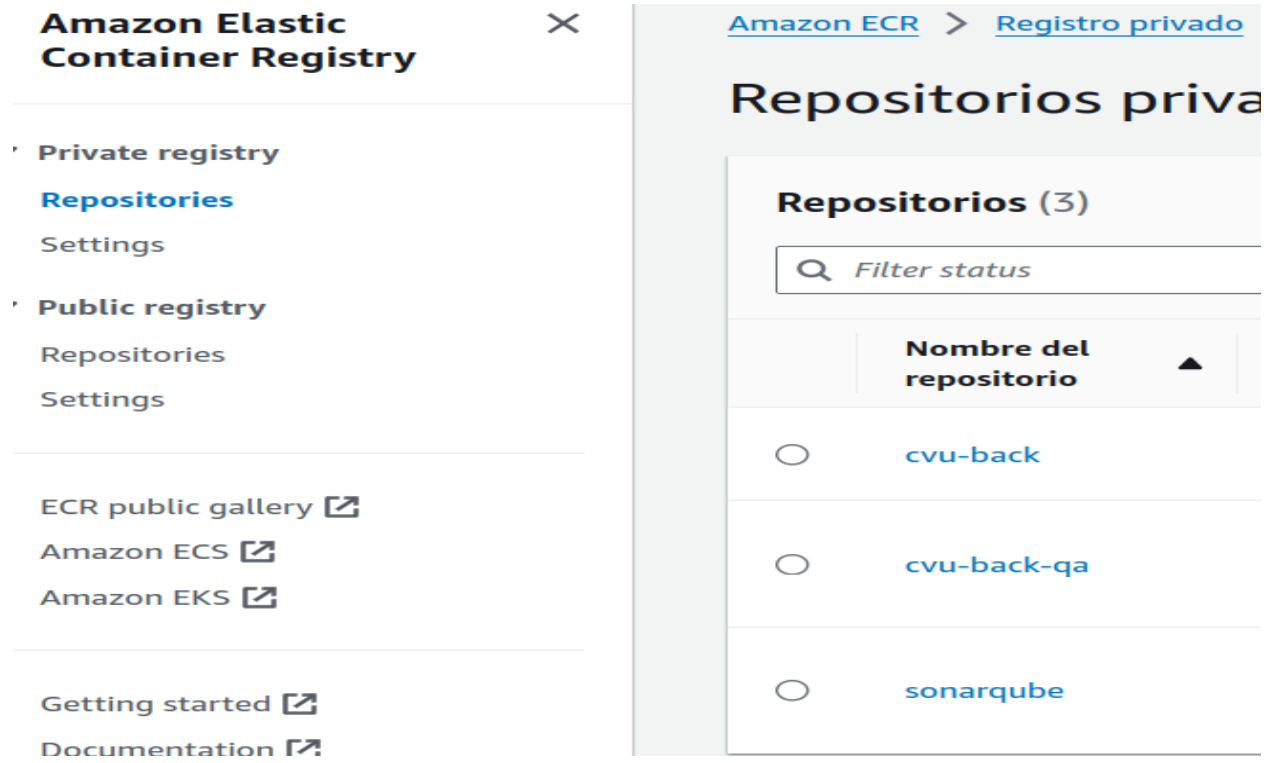
```yaml
version: 0.2

env:
  variables:
    AWS_ACCOUNT_ID: "573165592496"
    CLUSTER: "dev-cvu"
    SERVICE: "cvu-back"
    IMAGE_REPO_NAME: "cvu-back"
    IMAGE_TAG: "latest"

phases:
  pre_build:
    commands:
      - aws --version
      - echo Ingresado a Amazon ECR...
      - aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 573165592496.dkr.ecr.us-east-1.amazonaws.com

  build:
    commands:
      - echo Creando imagen de Docker...
      - docker build -f .docker/Dockerfile -t cvu-back .
      - docker tag cvu-back:latest 573165592496.dkr.ecr.us-east-1.amazonaws.com/cvu-back:latest
      - echo Copiando imagen de Docker a repositorio...
      - docker push 573165592496.dkr.ecr.us-east-1.amazonaws.com/cvu-back:latest
  post_build:
    commands:
      - echo Writing image definitions file...
      - printf '{"ImageURI":"%s"}' 573165592496.dkr.ecr.us-east-1.amazonaws.com/cvu-back:latest > imageDetail.json
      - aws ecs update-service --cluster $CLUSTER --service $SERVICE --force-new-deployment
artifacts:
  files:
    - imageDetail.json
```

Implementación en Amazon ECR

Después de que los artefactos se han generado exitosamente en CodeBuild, el siguiente paso en el flujo de trabajo de CI/CD es almacenar las imágenes de contenedor en Amazon Elastic Container Registry (ECR). Este servicio de administración de contenedores de Docker en la nube de AWS proporciona un repositorio seguro y escalable para almacenar, administrar y implementar imágenes de Docker. La integración entre CodeBuild y ECR permite la transferencia automatizada de imágenes de contenedor generadas durante el proceso de compilación a los repositorios de ECR correspondientes.

Configuración de ECR:

Se crea un repositorio en ECR para cada microservicio, proporcionando un lugar centralizado para almacenar y administrar las imágenes de contenedor.



Integración con CodeBuild:

En la configuración del proyecto de CodeBuild, se especifica la acción de construcción de Docker para generar, etiquetar y subir la imagen de contenedor a ecr.

```
  build:
    commands:
      - echo Creando imagen de Docker...
      - docker build -f .docker/Dockerfile -t cvu-back .
      - docker tag cvu-back:latest 573165592496.dkr.ecr.us-east-1.amazonaws.com/cvu-back:latest
      - echo Copiando imagen de Docker a repositorio...
      - docker push 573165592496.dkr.ecr.us-east-1.amazonaws.com/cvu-back:latest
  post_build:
    commands:
      - echo Writing image definitions file...
      - printf '{"ImageURI":"%s"}' 573165592496.dkr.ecr.us-east-1.amazonaws.com/cvu-back:latest > imageDetail.json
      - aws ecs update-service --cluster $CLUSTER --service $SERVICE --force-new-deployment
artifacts:
  files:
    - imageDetail.json
```

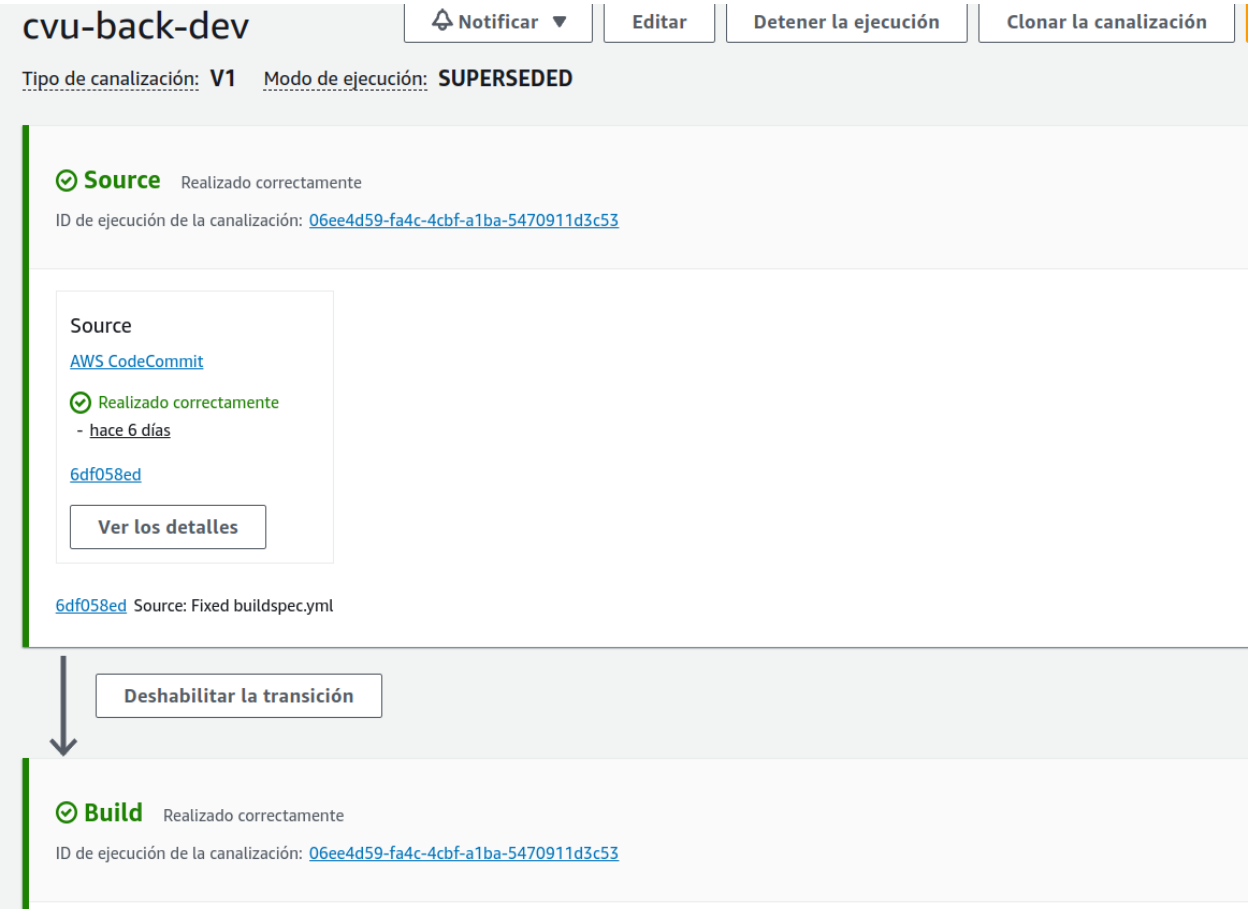
Transferencia Automatizada:

Después de que la imagen de contenedor se ha construido exitosamente en CodeBuild, se configura una acción adicional en el pipeline de CodePipeline para transferir la imagen a ECR. Esto garantiza que las imágenes de contenedor estén disponibles y listas para su implementación en el entorno de ECS Fargate.

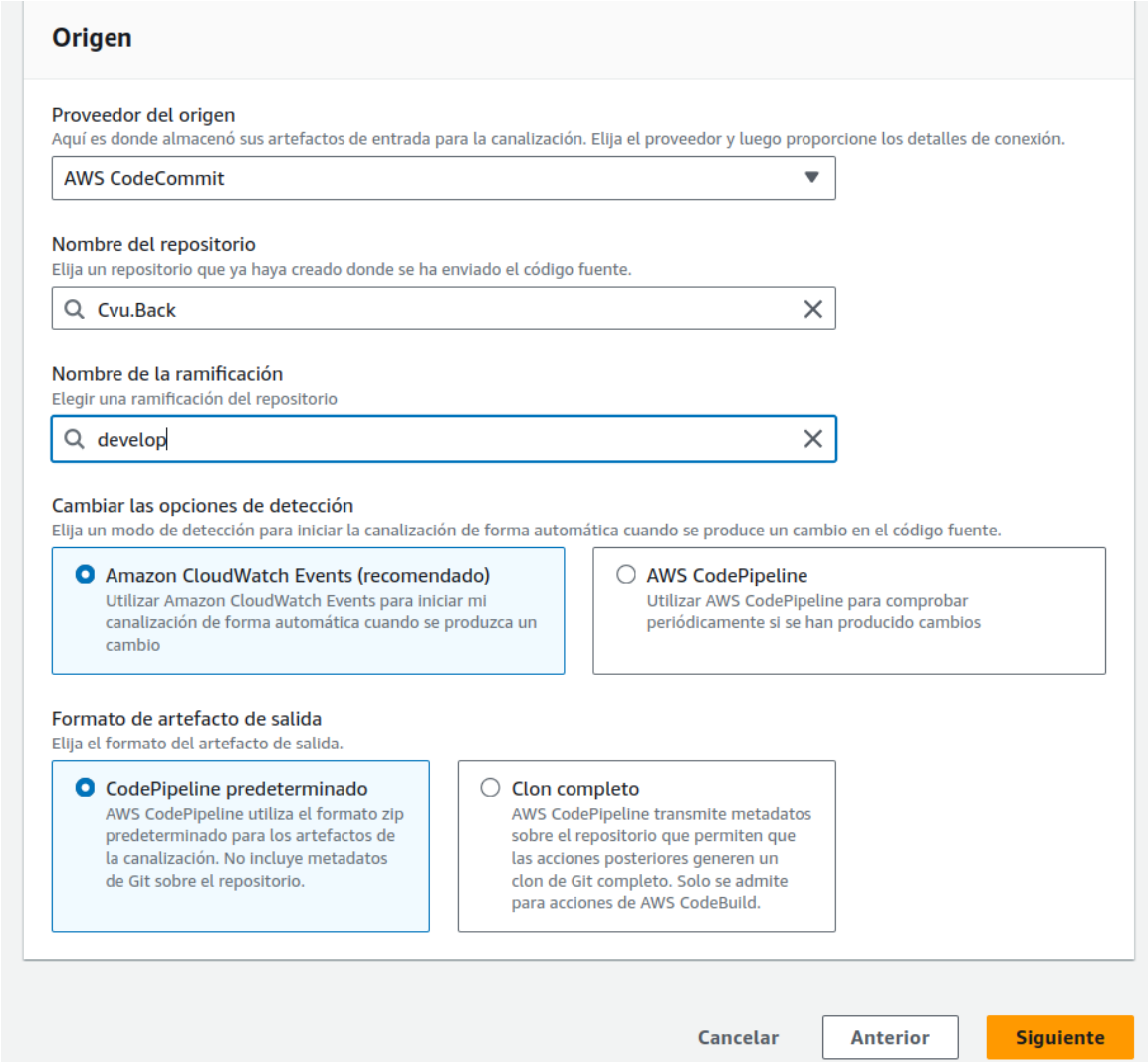
AWS CodePipeline

AWS CodePipeline es el servicio encargado de orquestar y automatizar el flujo de trabajo de CI/CD en CVU. A continuación, se detallan los componentes y pasos esenciales que conforman la implementación técnica de este servicio:

- Definición de Pipelines: Se configuran pipelines específicos para cada microservicio en el entorno de CVU. Cada pipeline está compuesto por una serie de etapas que representan las diferentes fases del proceso de entrega de software, como compilación, pruebas y despliegue.



- Integración con Otros Servicios: CodePipeline se integra estrechamente con otros servicios de AWS, como CodeBuild y CodeDeploy, así como con herramientas externas, para ejecutar acciones específicas en cada etapa del pipeline. Por ejemplo, en la etapa de compilación, CodePipeline puede llamar a CodeBuild para compilar y empaquetar el código fuente del microservicio.
-



- Monitoreo y Registro de Actividades: CodePipeline proporciona herramientas integradas para monitorear y registrar las actividades de entrega en tiempo real. Esto permite a los equipos de desarrollo y operaciones supervisar el progreso del pipeline, identificar posibles problemas y tomar medidas correctivas de manera proactiva.

Actualización del Servicio ECS

Después de que el pipeline haya construido y almacenado la nueva imagen en Amazon ECR, el siguiente paso es actualizar el servicio o tarea de ECS. En este caso, se prefiero actualizar el servicio ECS directamente. se Configuro una acción en el pipeline para llamar a la API de ECS y actualizar la definición del servicio con la nueva imagen. Esto garantiza una actualización gradual del servicio sin interrumpir el tráfico existente

Los pasos básicos para esta actualización son:
Identificar el servicio ECS correspondiente en tu clúster.

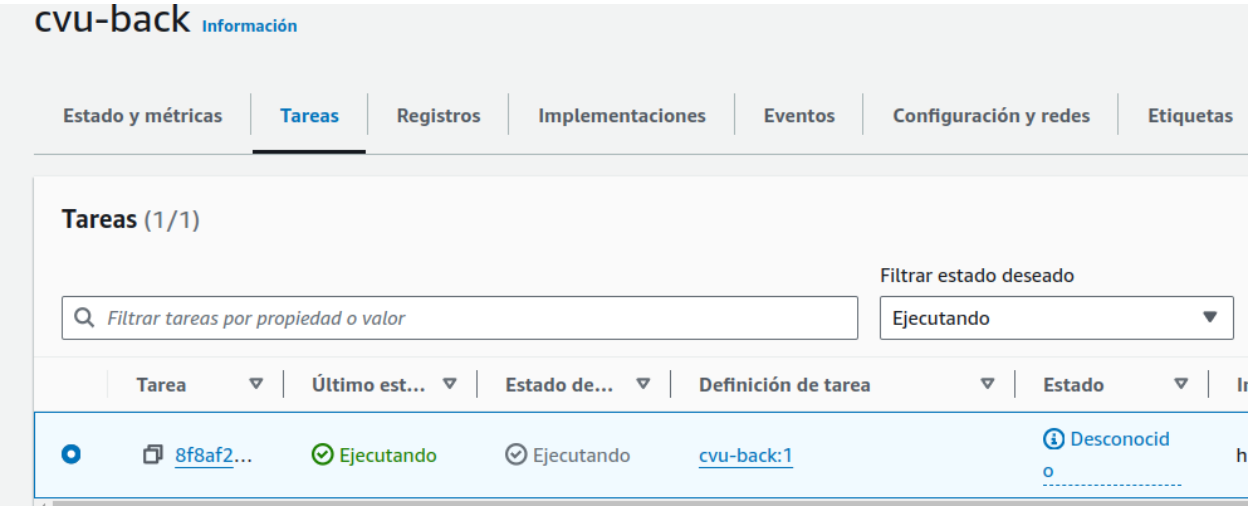
Cvu.Back / buildspec.yml Información

```yaml
version: 0.2

env:
  variables:
    AWS_ACCOUNT_ID: "573165592496"
    CLUSTER: "dev-cvu"
    SERVICE: "cvu-back"
    IMAGE_REPO_NAME: "cvu-back"
    IMAGE_TAG: "latest"

```

Actualizar la definición del servicio con la nueva versión de la imagen de contenedor.

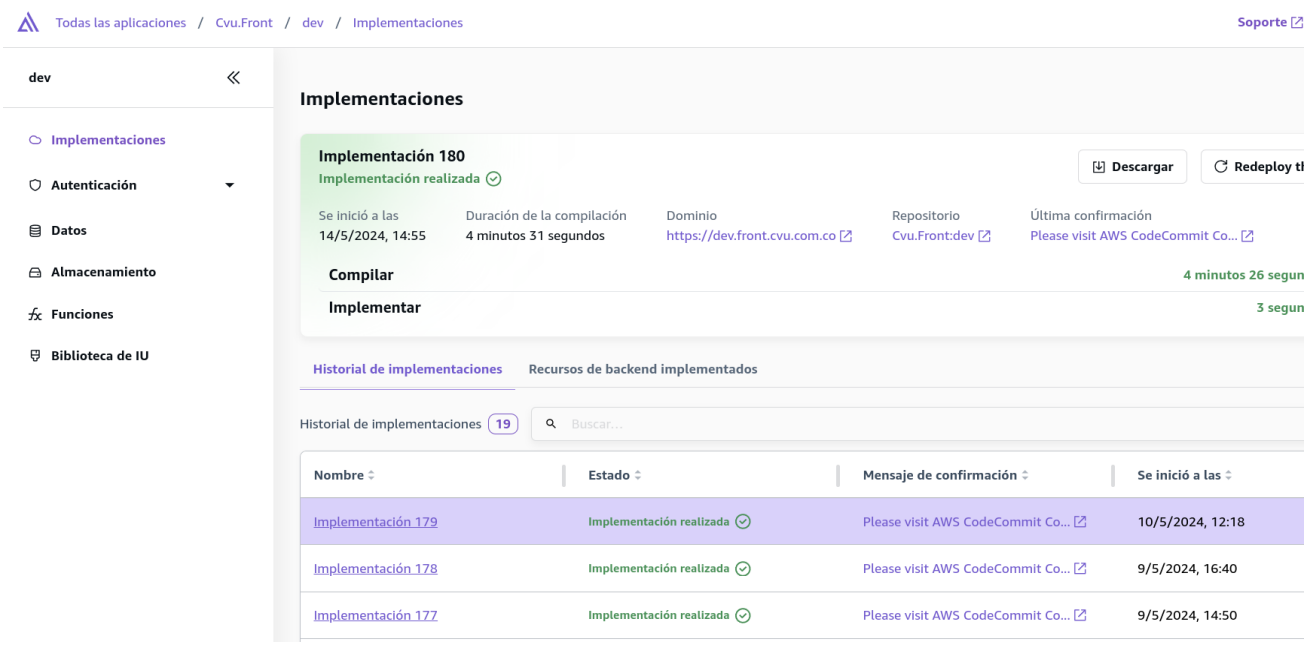
```
  post_build:
    commands:
      - echo Writing image definitions file...
      - printf '{"ImageURI":"%s"}' 573165592496.dkr.ecr.us-east-1.amazonaws.com/cvu-back:latest > imageDetail.json
      - aws ecs update-service --cluster $CLUSTER --service $SERVICE --force-new-deployment
artifacts:
  files:
    - imageDetail.json
```

Actualización de Servicio y Tarea:



Proceso de Automatización Amplify

Además de la implementación de microservicios en ECS Fargate, CVU utiliza AWS Amplify para la gestión y automatización de aplicaciones frontend. Amplify simplifica el proceso de desarrollo y despliegue de aplicaciones web y móviles ya que Cuando hacemos cambios en el código, Amplify los revisa automáticamente y, si todo está bien, actualiza nuestra aplicación en línea sin que tengamos que hacer mucho trabajo manual. Esto nos ayuda a lanzar nuevas características más rápido y a asegurarnos de que nuestra aplicación esté siempre actualizada.



Conclusión:

En conclusión, la implementación de Continuous Integration/Continuous Deployment (CI/CD) en CVU ha sido un cambio significativo para el proyecto. Esta automatización ha simplificado enormemente nuestro proceso de desarrollo y despliegue de aplicaciones en la nube. Ahora podemos lanzar nuevas funciones y correcciones de errores de manera más rápida y sin problemas, lo que nos permite adaptarnos mejor a las necesidades del proyecto. Además, hemos mejorado la confiabilidad de nuestras aplicaciones al estandarizar y automatizar nuestras prácticas de implementación.

DOUBLE V PARTNERS

INFORME TÉCNICO DE INFRAESTRUCTURA DE CVU

Elaborado por:

Equipo DVP

índice

Introducción.

Este informe presenta en detalle la infraestructura de microservicios implementada en la nube de AWS para el proyecto CVU. En él, se examina minuciosamente la configuración y los componentes esenciales empleados en la construcción y mantenimiento de esta plataforma. Su propósito radica en ofrecer una comprensión exhaustiva de la arquitectura subyacente que impulsa el proyecto, resaltando tanto su escalabilidad como su eficiencia operativa.

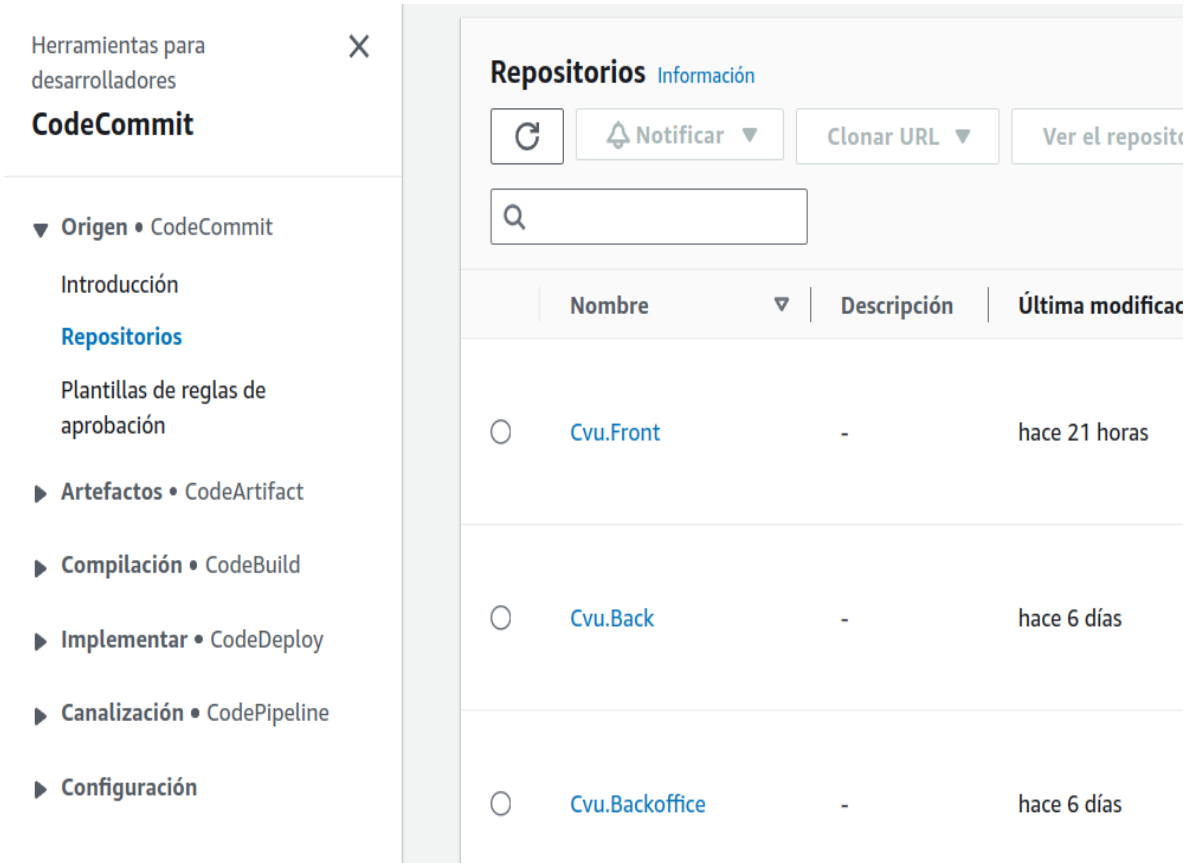
Explore a continuación los detalles clave que hacen posible este entorno altamente eficiente y fiable

Arquitectura de la Infraestructura.

La arquitectura del proyecto consta de los siguientes componentes:

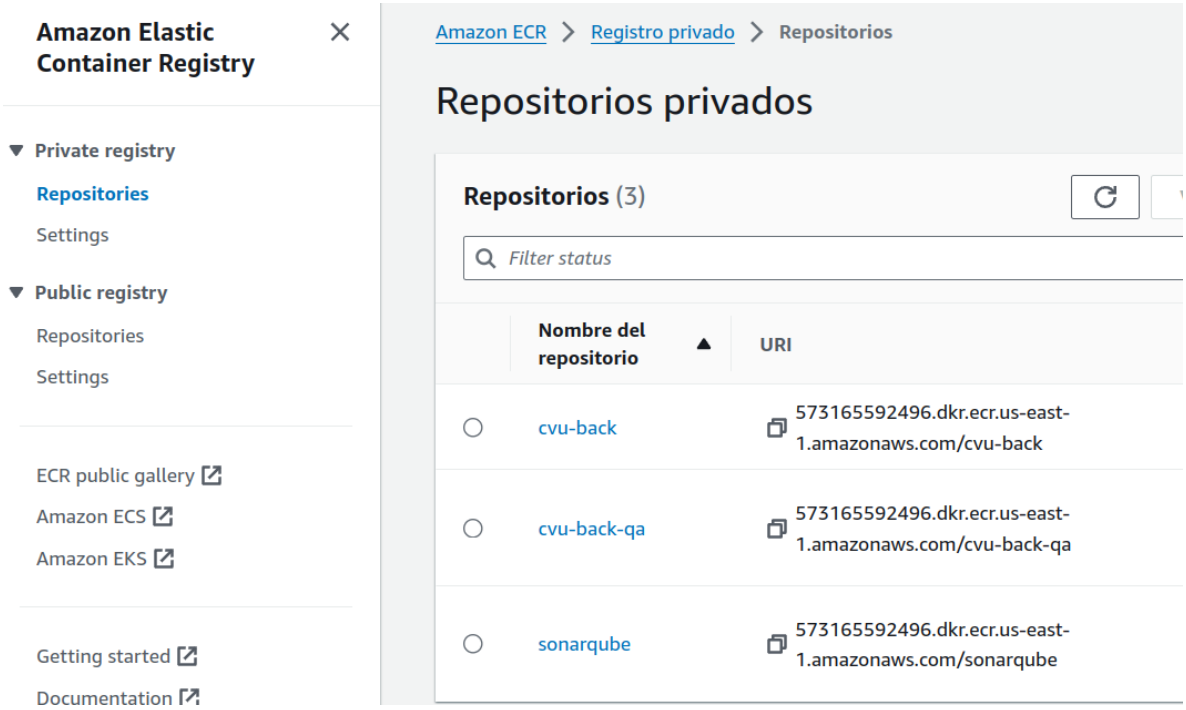
Amazon CodeCommit:

Se emplea como servicio para almacenar los repositorios del proyecto, permitiendo un control de versiones eficiente para gestionar el código fuente. Los desarrolladores pueden colaborar en el código de manera segura y realizar seguimientos de los cambios realizados.



Amazon Elastic Container Registry (ECR):

Este servicio se utiliza para almacenar y gestionar las imágenes de contenedores Docker utilizadas por la aplicación. Las imágenes se actualizan y versionan según las necesidades del proyecto, lo que facilita el despliegue de nuevas versiones de la aplicación de manera rápida y segura.

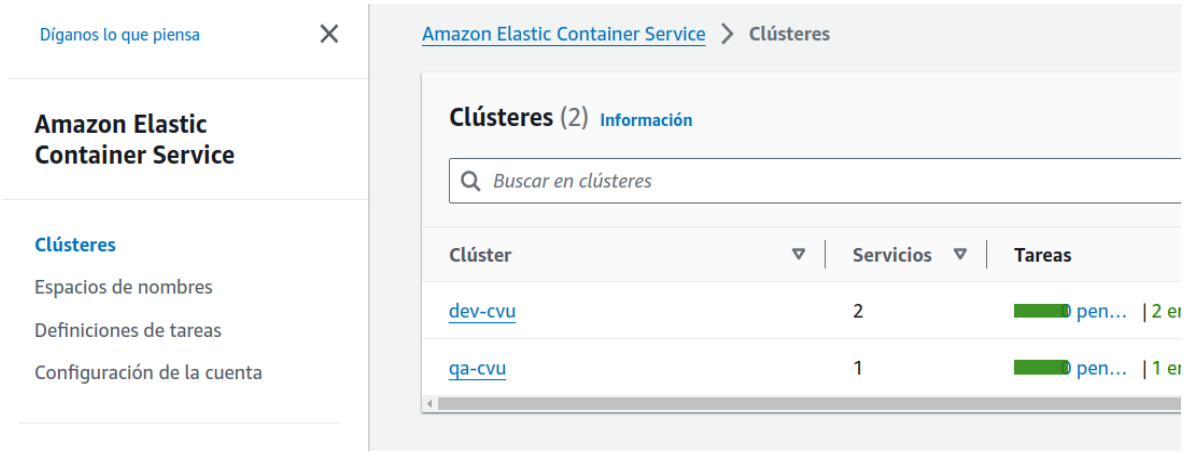


VPC (Virtual Private Cloud):

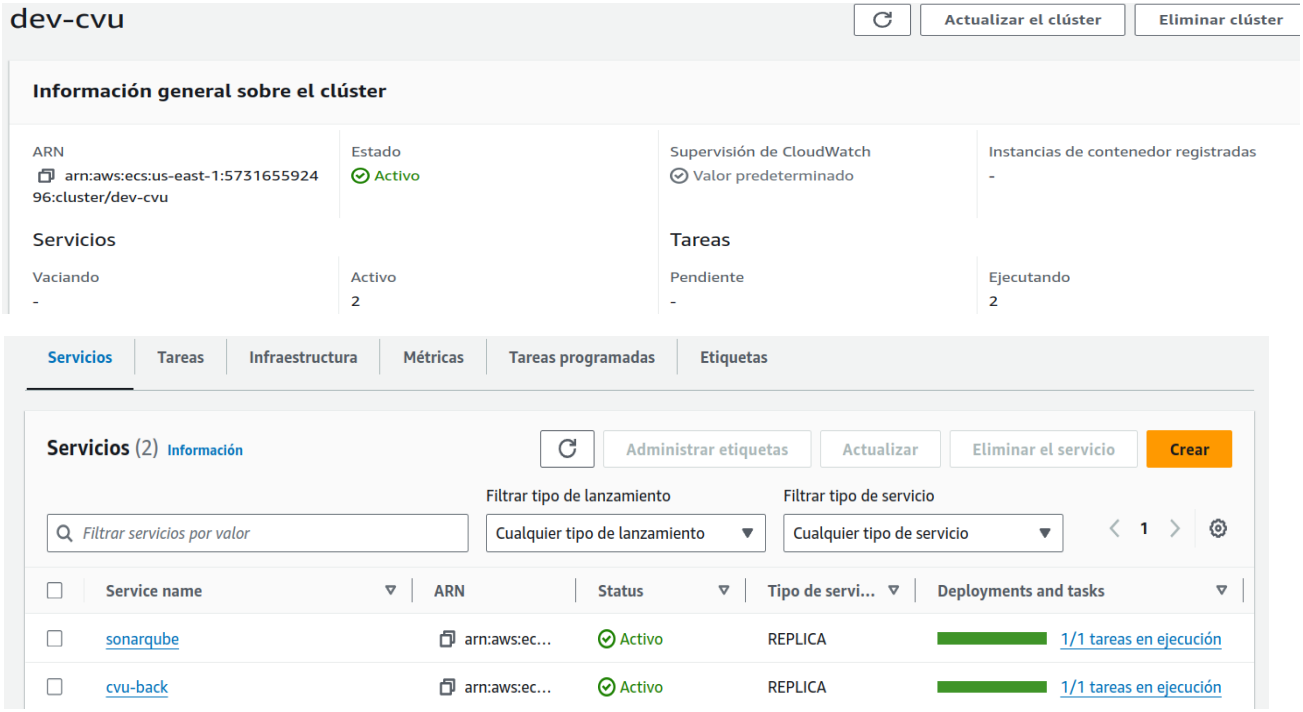
La VPC es un componente esencial de la infraestructura de CVU en AWS. Es una red virtual privada en la nube que nos proporciona un entorno de red aislado y seguro para implementar nuestros servicios. Con la VPC, podemos controlar el acceso a nuestros recursos, garantizar la seguridad y privacidad de nuestros datos, y escalar nuestra infraestructura de red según sea necesario

ECS Fargate:

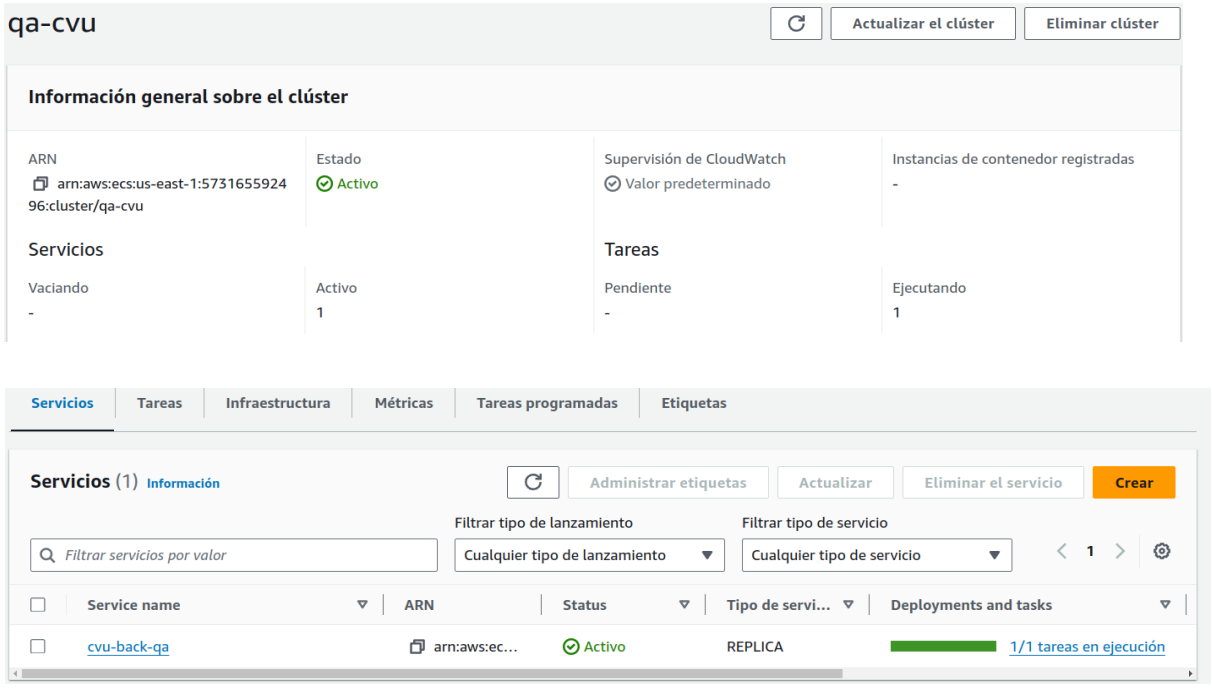
Se trata de un servicio de contenedores administrado por AWS que permite ejecutar contenedores en la nube sin la necesidad de administrar servidores. CVU utiliza ECS Fargate para ejecutar los contenedores que componen la aplicación, lo que proporciona una infraestructura flexible y escalable que se ajusta automáticamente a la carga de trabajo.



CVU-dev:

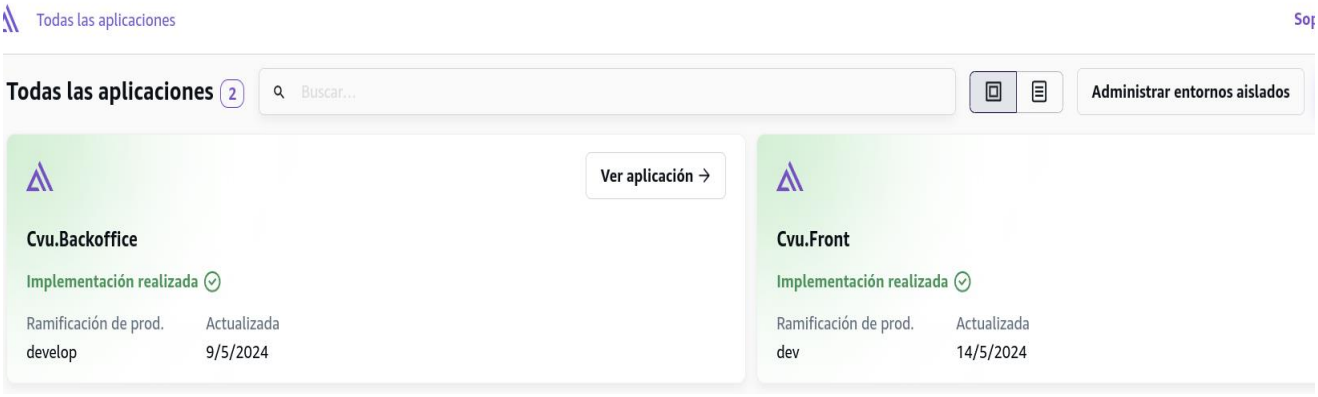


CVU-QA

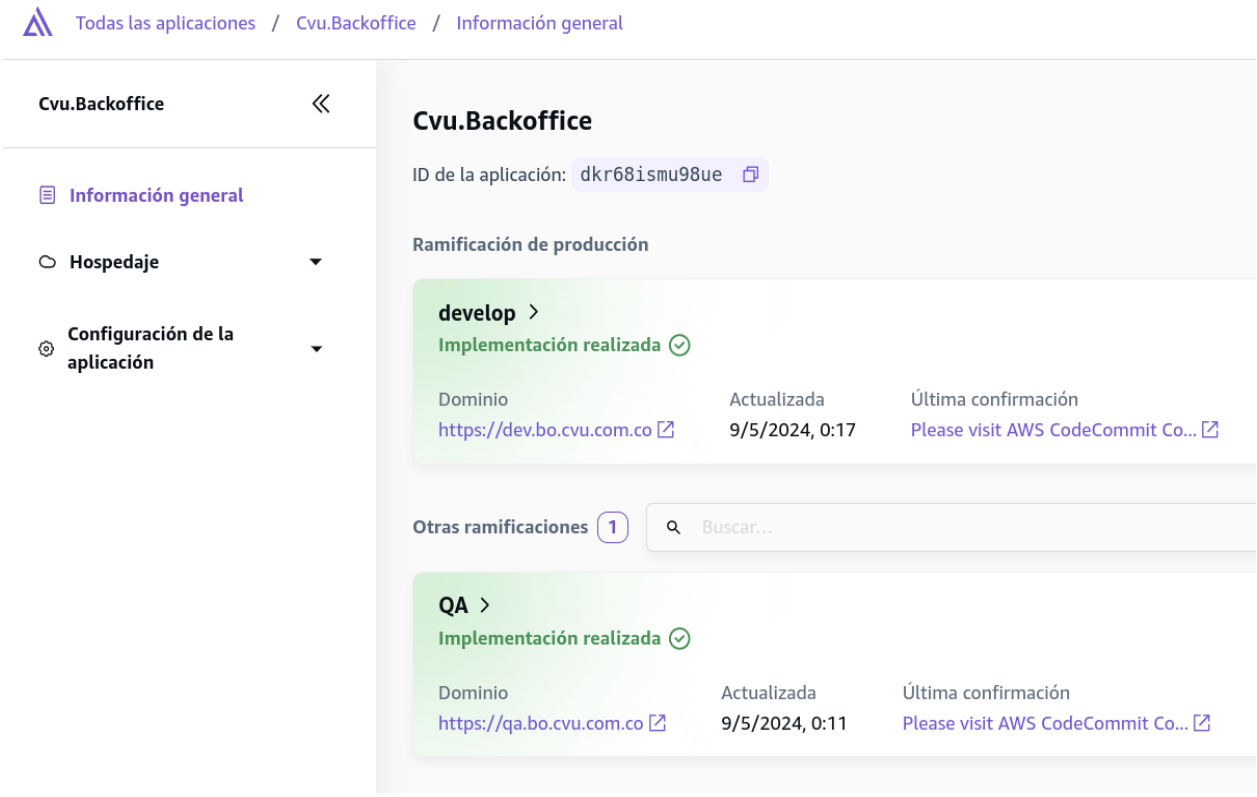


Amplify:

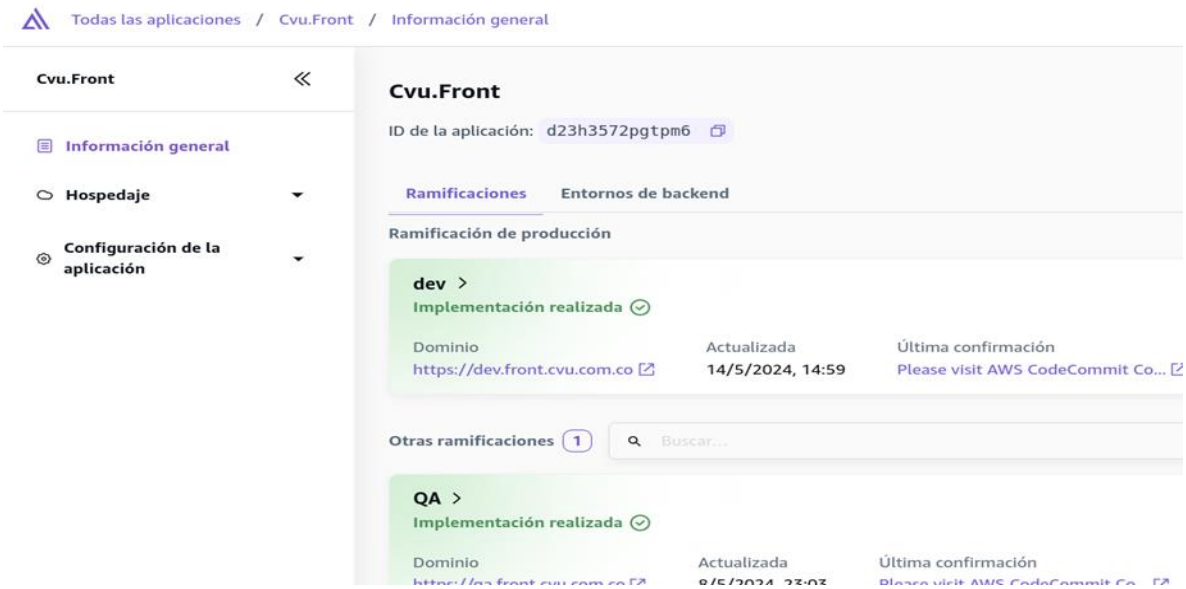
Amplify es un servicio de AWS que proporciona una forma rápida y sencilla de construir y desplegar aplicaciones web y móviles. CVU utiliza Amplify para la gestión de páginas webs estáticas y la integración con servicios en la nube, lo que permite ofrecer una experiencia de usuario moderna y personalizada.



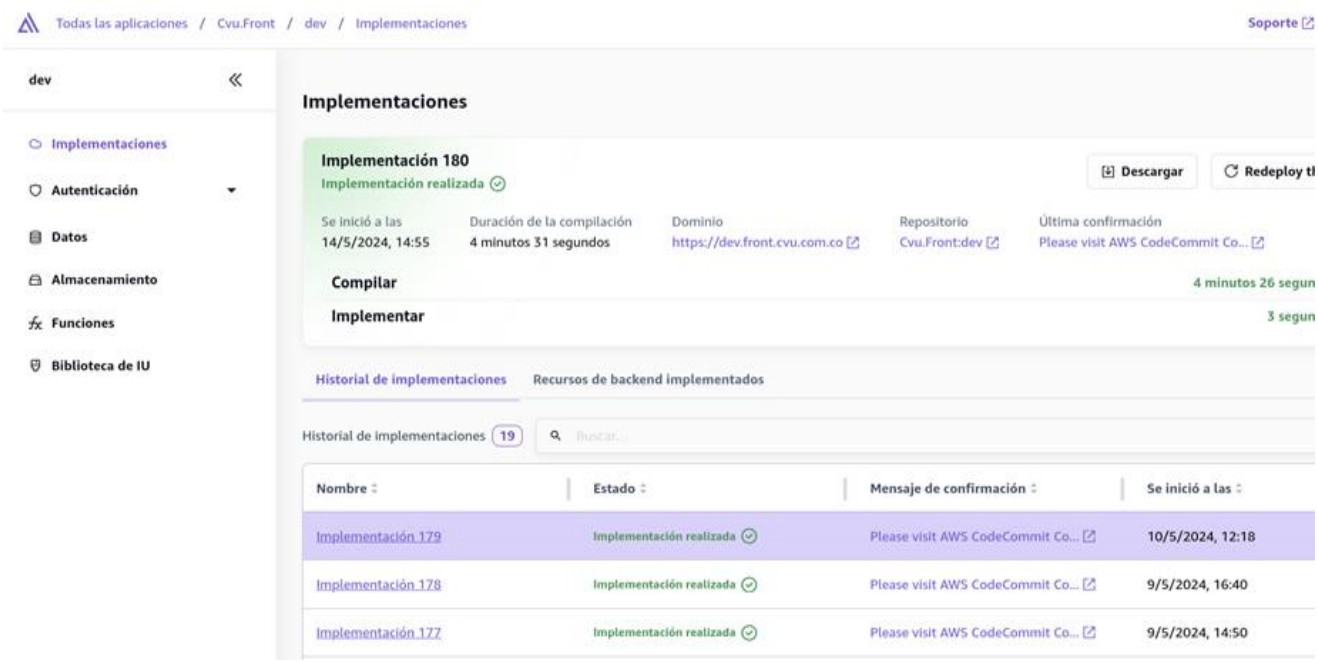
Cvu.Backoffice



Cvu.Front



Adicional con Amplify, podemos hacer que el proceso de enviar cambios y actualizar nuestra aplicación sea mucho más fácil. Cuando hacemos cambios en el código, Amplify los revisa automáticamente y, si todo está bien, actualiza nuestra aplicación en línea sin que tengamos que hacer mucho trabajo manual. Esto nos ayuda a lanzar nuevas características más rápido y a asegurarnos de que nuestra aplicación esté siempre actualizada.



SonarQube:

Se integra para evaluar y mejorar la calidad del código de la aplicación. SonarQube realiza análisis estáticos del código y proporciona informes detallados sobre problemas de calidad y deuda técnica adicional se pueden obtener métricas y análisis para evaluar la salud general del proyecto.
Algunas de las características y métricas que se pueden encontrar en el informe general de SonarQube incluyen:

- **Índice de calidad del código**: Proporciona una visión general de la calidad del código fuente, evaluando aspectos como la complejidad, las vulnerabilidades y los errores de codificación.
- **Cobertura de pruebas**: Muestra el porcentaje de código cubierto por pruebas unitarias, lo que ayuda a identificar áreas del código que no están siendo probadas adecuadamente.
- **Deuda técnica**: Indica la cantidad de trabajo necesario para remediar problemas de calidad del código, como duplicación, código duplicado y malas prácticas de programación.
- **Reglas de calidad**: Enumera las reglas específicas que se están aplicando para evaluar el código fuente y proporciona información detallada sobre los problemas encontrados en cada regla.

La implementación de SonarQube se realizó a través de un contenedor ECS Fargate en AWS (Amazon Web Services). Esta elección de arquitectura ofrece varias ventajas clave para asegurar un despliegue eficiente y escalable de SonarQube

Acceso a SonarQube en ECS Fargate

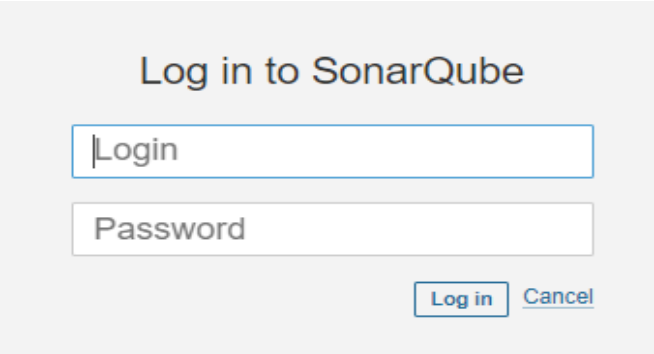
- Abre tu navegador web preferido.

Ingresa la URL del endpoint de SonarQube en la barra de direcciones del navegador (por ejemplo, http://54.147.237.78:9000 utilizando HTTP o HTTPS y el puerto asignado).

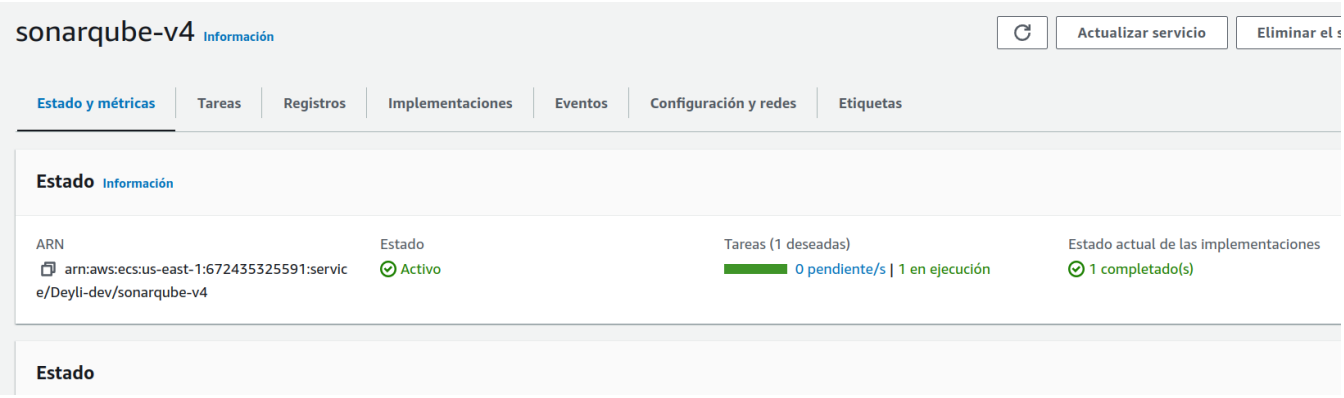
Presiona Enter para acceder a la interfaz web de SonarQube.

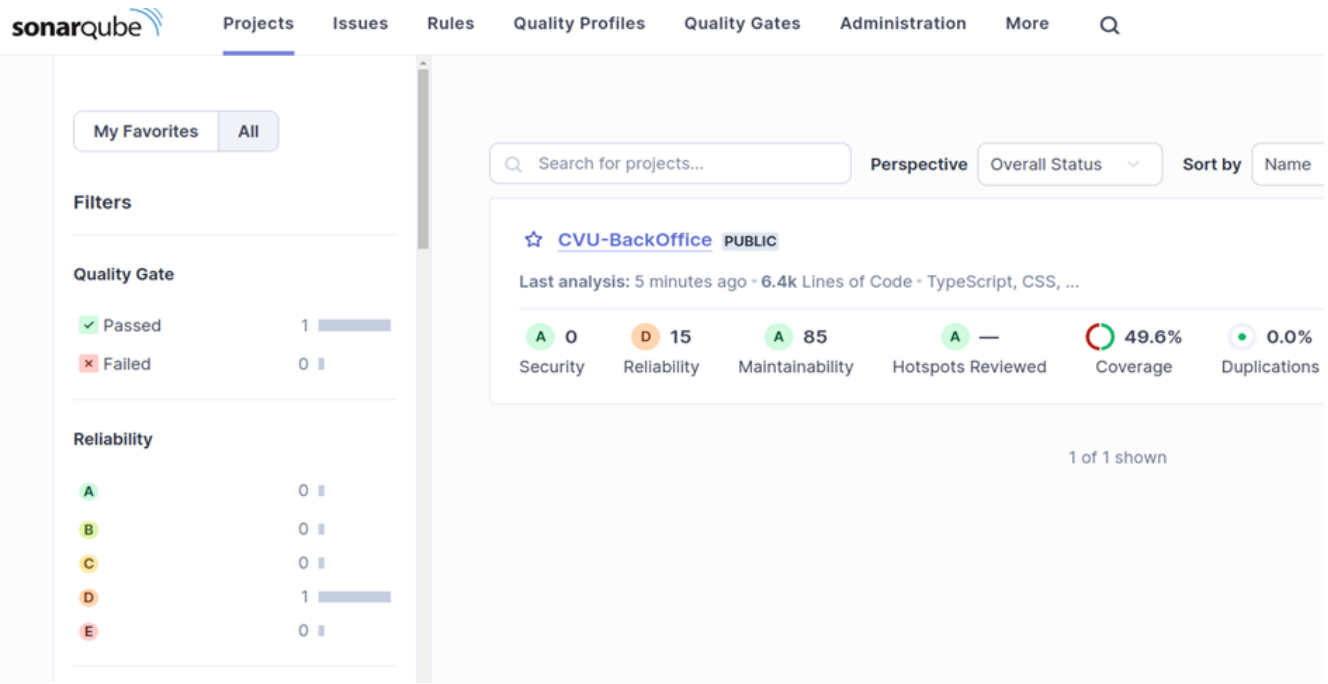
- Inicio de Sesión en SonarQube

Una vez que accedes a la interfaz web de SonarQube, es posible que se te solicite iniciar sesión. Si ya tienes credenciales, ingrésalas en los campos correspondientes (nombre de usuario y contraseña) y haz clic en "Iniciar Sesión".



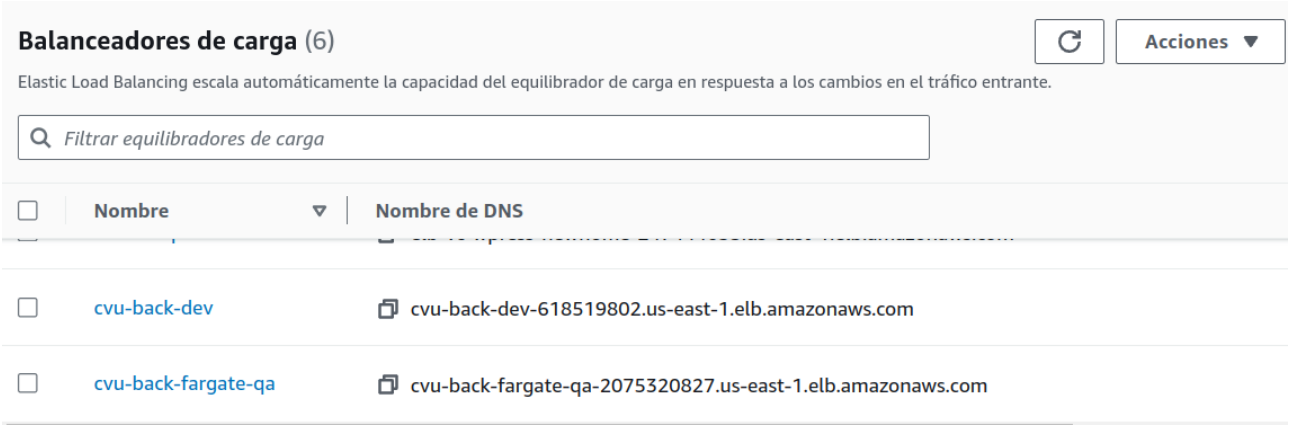
Si es la primera vez que accedes o necesitas crear una cuenta, sigue las instrucciones proporcionadas en la página de inicio de sesión para registrar una nueva cuenta de usuario.





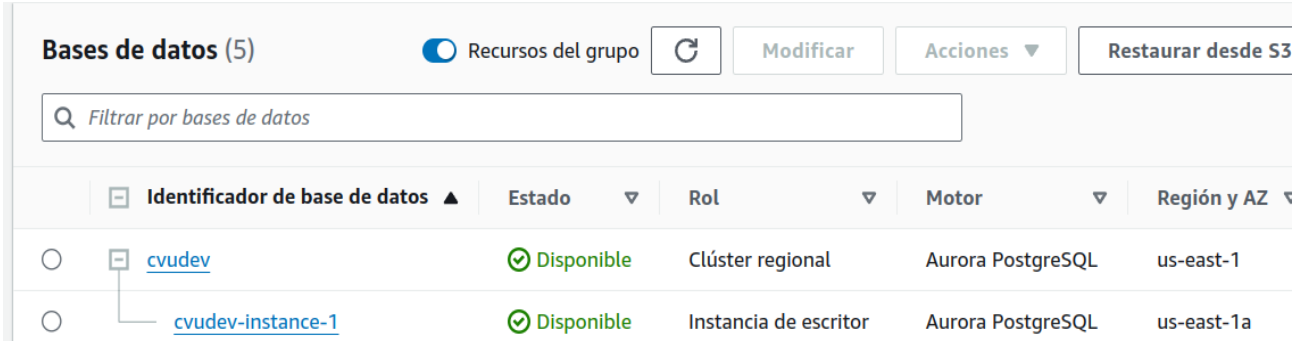
Amazon Load Balancer:

Se utiliza un balanceador de carga de AWS para distribuir el tráfico de red entrante entre los diferentes contenedores ejecutándose en ECS Fargate. Esto garantiza una alta disponibilidad y rendimiento de la aplicación, ya que el tráfico se distribuye de manera equitativa entre los diferentes recursos disponibles.



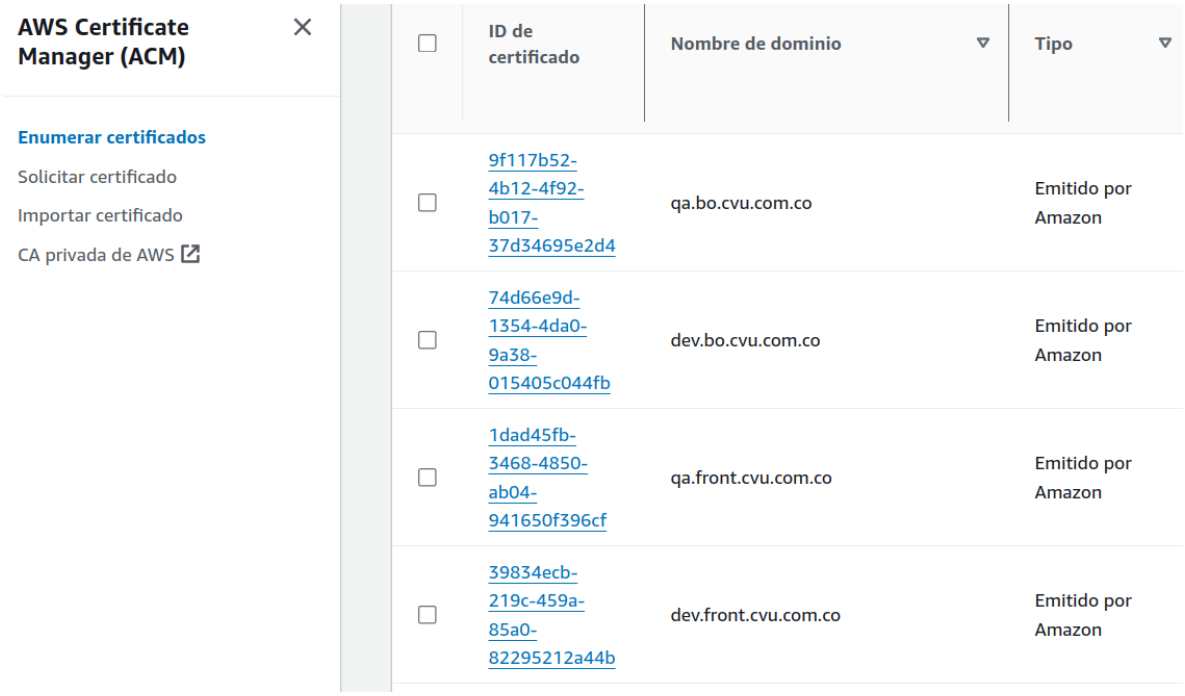
Amazon RDS PostgreSQL:

Se emplea una base de datos gestionada por AWS que utiliza el motor de base de datos PostgreSQL. Esta base de datos proporciona un almacenamiento seguro y escalable para los datos de la aplicación, garantizando la integridad y disponibilidad de los mismos. CVU utiliza RDS PostgreSQL para almacenar datos críticos como información de usuarios, registros de actividad y configuraciones de la aplicación.



Amazon Certificate Manager (ACM):

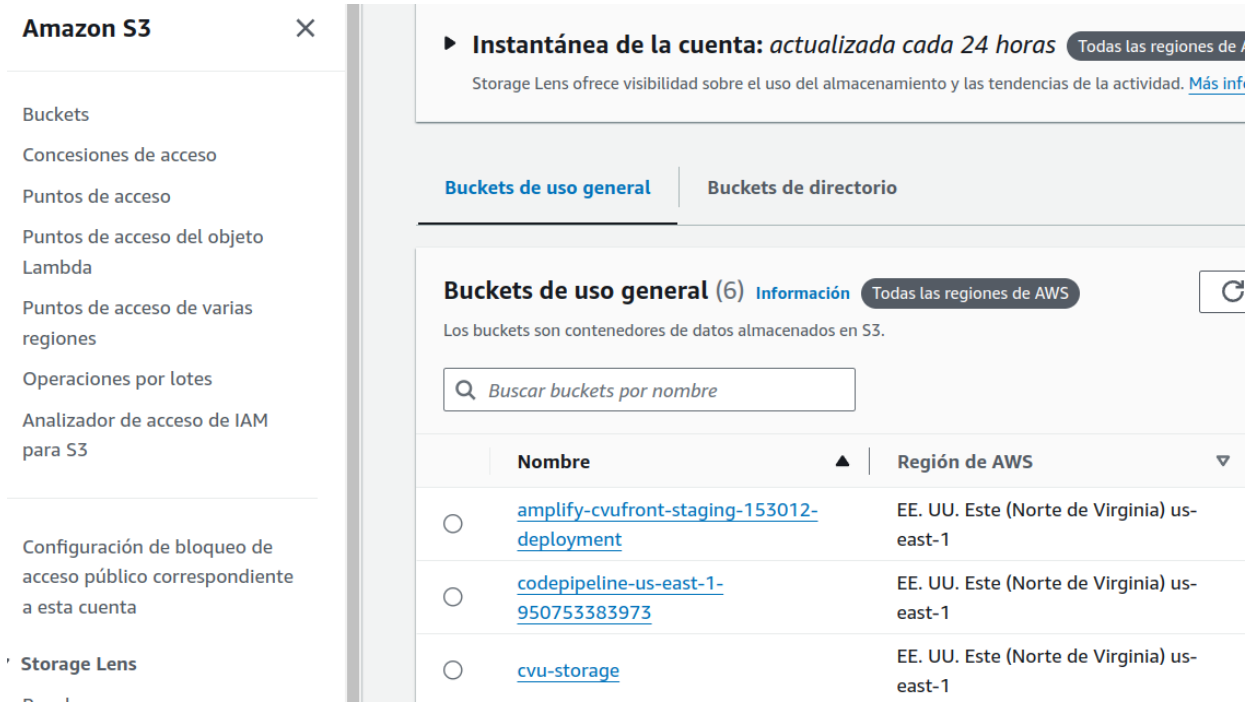
ACM se utiliza para gestionar y aprovisionar fácilmente certificados SSL/TLS para las aplicaciones web desplegadas en la infraestructura de AWS. Proporciona una forma sencilla de asegurar las comunicaciones entre los usuarios y la aplicación, garantizando la confidencialidad e integridad de los datos transmitidos.



Amazon S3 (Simple Storage Service):

Amazon S3 se utiliza como un servicio de almacenamiento en la nube altamente escalable y duradero. En CVU, S3 se emplea para almacenar recursos estáticos, como archivos multimedia y documentos. Esto proporciona un almacenamiento seguro y de alta disponibilidad para los recursos estáticos de la aplicación.

La integración con S3 permite a CVU aprovechar la escalabilidad y la durabilidad de este servicio para almacenar y servir contenido estático de manera eficiente, lo que contribuye a una experiencia de usuario fluida y rápida.



Conclusión:

En conclusión, hemos creado una infraestructura sólida y eficiente en la nube de AWS para respaldar el proyecto CVU. Desde el almacenamiento de nuestro código en AWS CodeCommit hasta el despliegue automatizado con AWS Amplify, hemos aprovechado una variedad de servicios de AWS para construir y mantener nuestra plataforma.

La arquitectura de microservicios en ECS Fargate nos ha proporcionado la flexibilidad y escalabilidad necesarias para ajustarnos a las demandas cambiantes del proyecto CVU. Además, mediante la integración de servicios como SonarQube, hemos asegurado altos estándares de calidad de código y seguridad. Esta sinergia entre tecnologías y prácticas nos ha permitido ofrecer una experiencia moderna y confiable.

DOUBLE V PARTNERS

Guía para Desplegar una Nueva Versión en el Ambiente de Producción

Elaborado por:

Equipo DVP

Versión:

1.0

Fecha de elaboración:

1 de Diciembre de 2024

Índice

Introducción

El despliegue de una nueva versión en producción es un proceso crítico dentro del ciclo de vida del software. Este documento proporciona una guía detallada para realizar esta tarea en el proyecto CVU, utilizando servicios de AWS como **Elastic Container Registry (ECR)** y **Elastic Container Service (ECS)**. Es importante destacar que los pasos aquí descritos son específicos para los microservicios del backend, ya que los servicios de frontend y backoffice cuentan con un proceso de CI/CD completamente automatizado a través de AWS Amplify, el cual gestiona la construcción, las pruebas y el despliegue de forma autónoma.

En este caso, el pipeline CI/CD implementado para el backend automatiza el proceso de construcción de la imagen Docker y su subida al repositorio ECR. Sin embargo, la actualización de la versión del servicio en ECS se realiza manualmente. Esta decisión fue tomada con el objetivo de mantener un mayor control sobre el proceso de despliegue en producción, asegurando que cualquier cambio sea cuidadosamente supervisado y ejecutado con precisión. Este documento asegura que todas las personas involucradas comprendan cada paso, lo ejecuten correctamente y mantengan la estabilidad del sistema durante todo el proceso.

Objetivo

Definir el procedimiento claro, paso a paso, para desplegar una nueva versión de los microservicios del proyecto CVU en el ambiente de producción, garantizando que:

- La nueva versión esté correctamente registrada en el clúster de ECS.
- Los servicios sean actualizados con interrupciones mínimas.
- Se realicen validaciones post-despliegue para confirmar el correcto funcionamiento.

Requisitos Previos

Antes de comenzar, asegúrate de cumplir con los siguientes puntos:

1. Permisos y Accesos

- Acceso a la consola de AWS con permisos para:
  - **ECS**: Gestión de clústeres, servicios y tareas.
  - **ECR**: Visualización y consulta de repositorios de imágenes.
  - **CloudWatch Logs**: Revisión de logs generados por los contenedores.
- Configuración del **AWS CLI** con un perfil que permita operaciones en producción.

2. Herramientas y Entorno

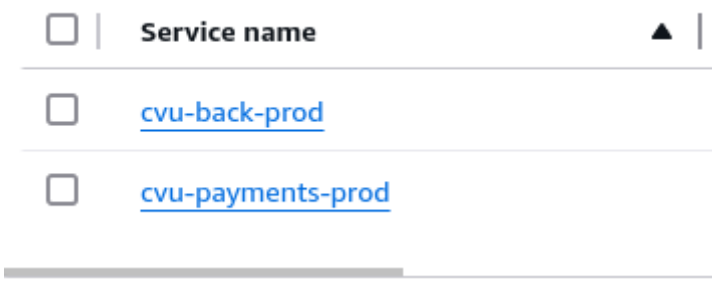
- Tener instalado y configurado:
  - **AWS CLI** en la máquina local.
  - **Docker** para pruebas locales, si es necesario.

3. Información Necesaria

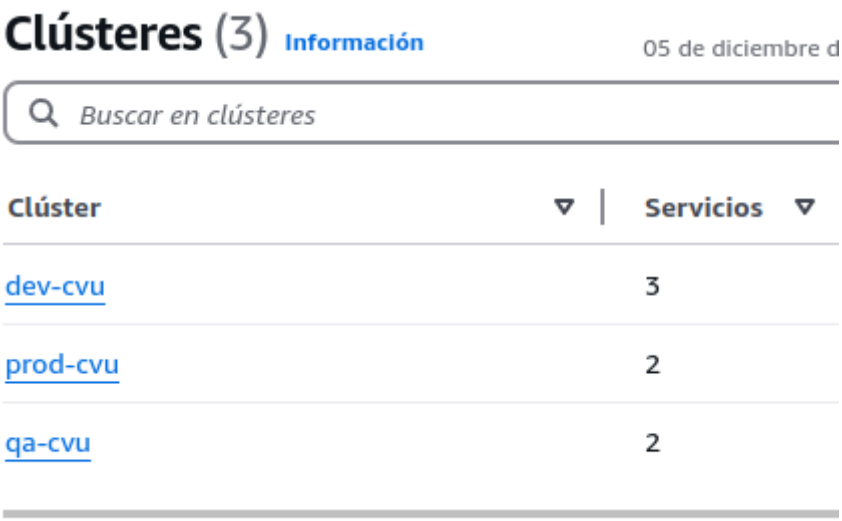
- **Tag de la nueva imagen**: Debes conocer el tag asignado a la imagen generada en ECR.

Nota: El proceso está configurado para el tag de la imagen sea siempre el último

- **Nombre del servicio ECS** que se actualizará.
- Ejemplo: cvu-back-service, cvu-payments



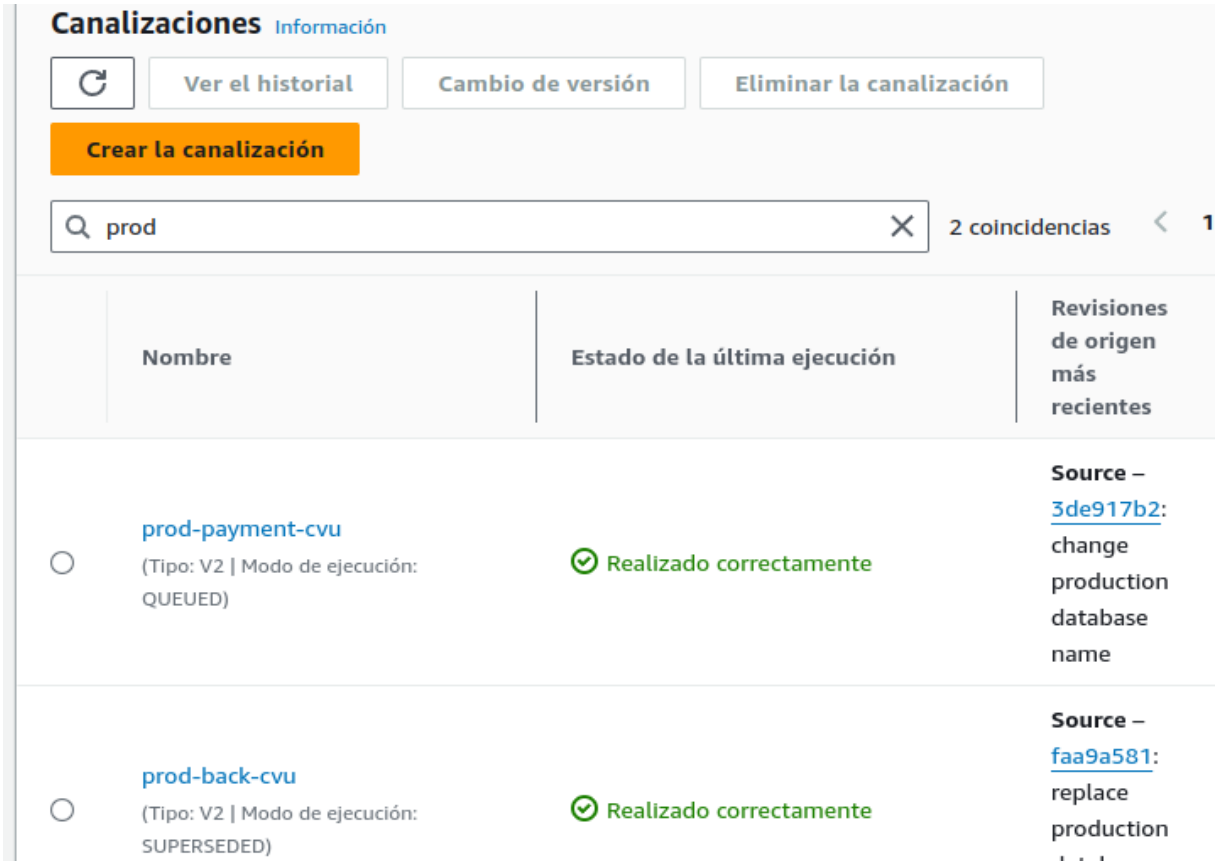
- **Clúster de ECS** donde está desplegado el servicio.
- Ejemplo: prod-cvu.



Proceso de Despliegue

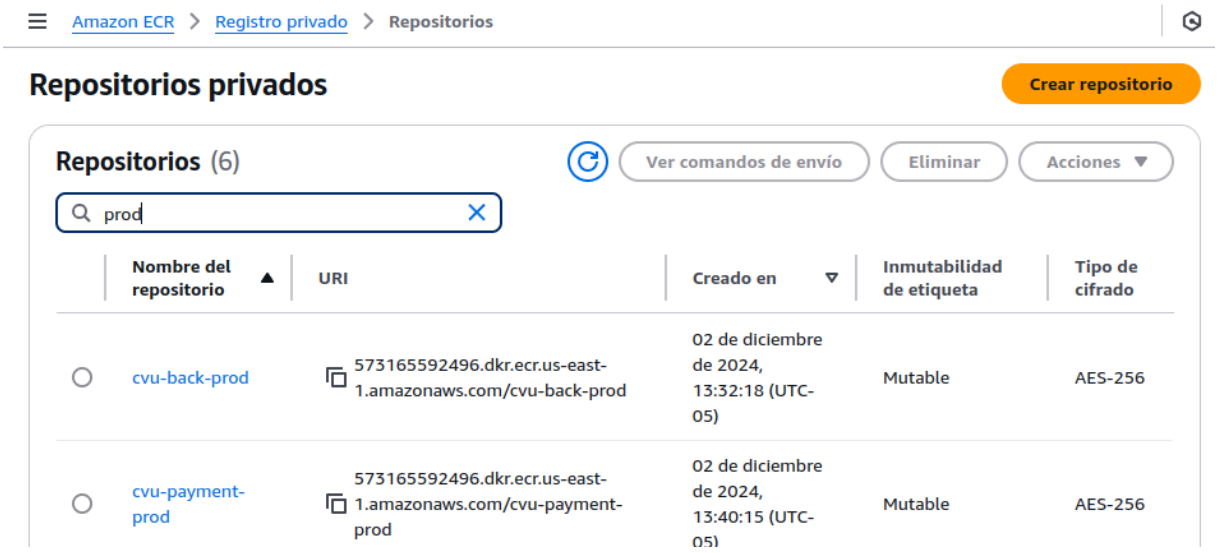
Paso 1: Validar la Generación y Subida de la Nueva Imagen a ECR

1. **Verificar el pipeline CI/CD:**
   - Asegúrate de que el pipeline de CodePipeline o CodeBuild haya finalizado exitosamente.

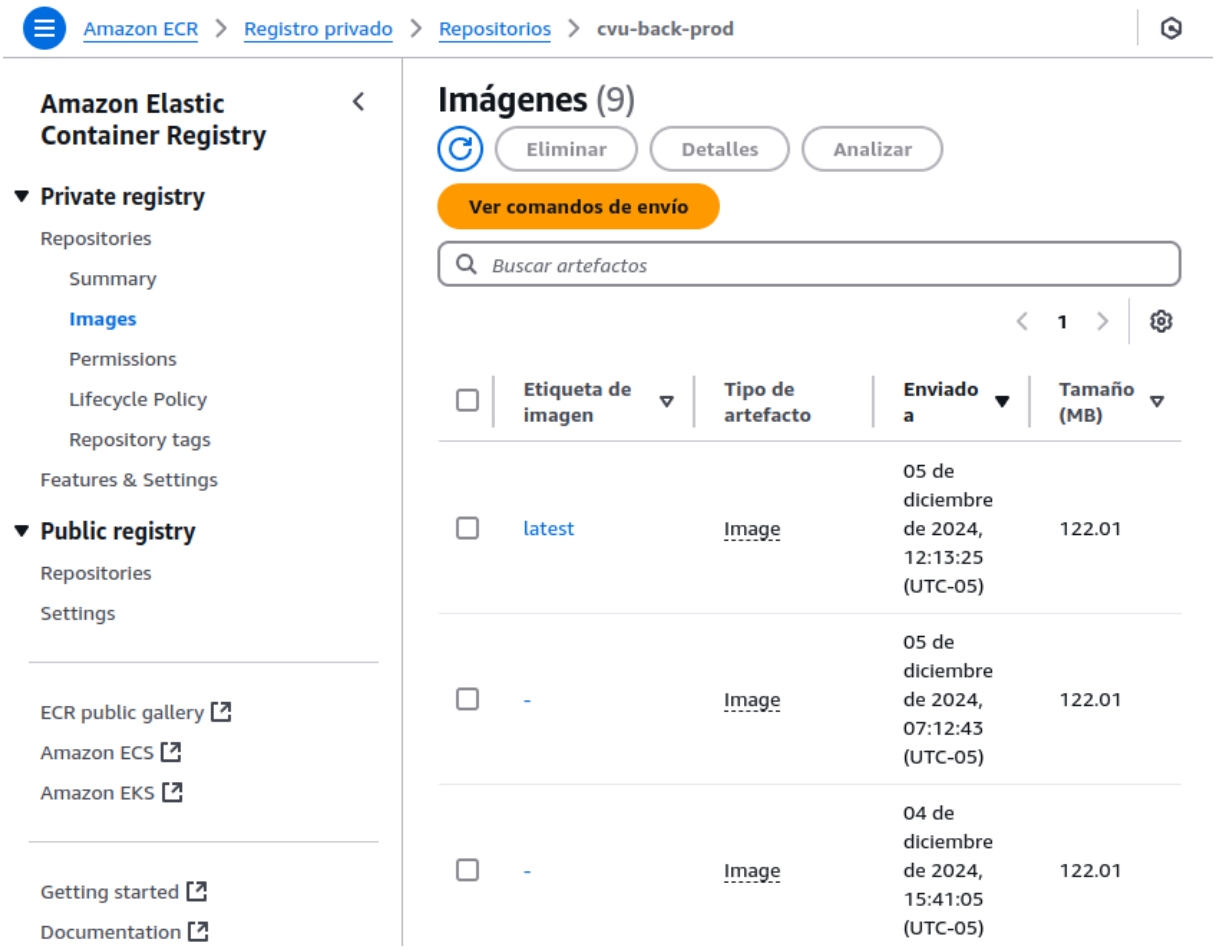


- El pipeline debe incluir los pasos de:
  - Construcción de la imagen Docker.
  - Subida de la imagen al repositorio ECR.

2. **Revisar la nueva imagen en ECR:**
   - Accede a la consola de AWS y dirígete a **Amazon ECR**.



- Selecciona el repositorio correspondiente al servicio (por ejemplo: cvu-back.prod).



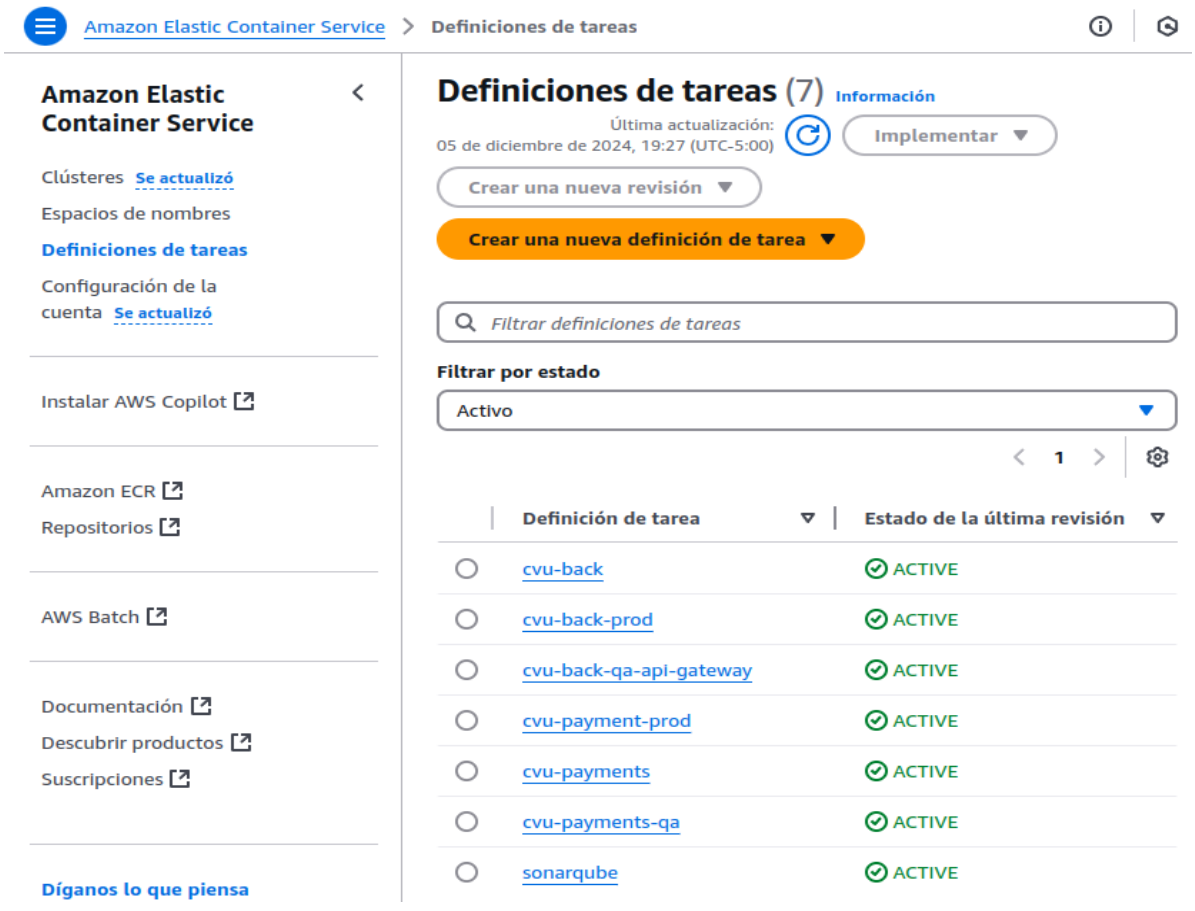
- Confirma que la imagen esté presente y etiquetada con el tag correcto, en este caso es el últimos (latest)

Paso 2: Actualizar la Definición de la Tarea para la Nueva Versión

Antes de actualizar el servicio ECS, es necesario modificar la definición de la tarea para que utilice la nueva versión de la imagen almacenada en Amazon ECR.

1. Acceder a Definiciones de Tareas en ECS

- En la consola de AWS, dirígete a **Elastic Container Service (ECS)**.
- En el menú lateral, selecciona **Task Definitions**.

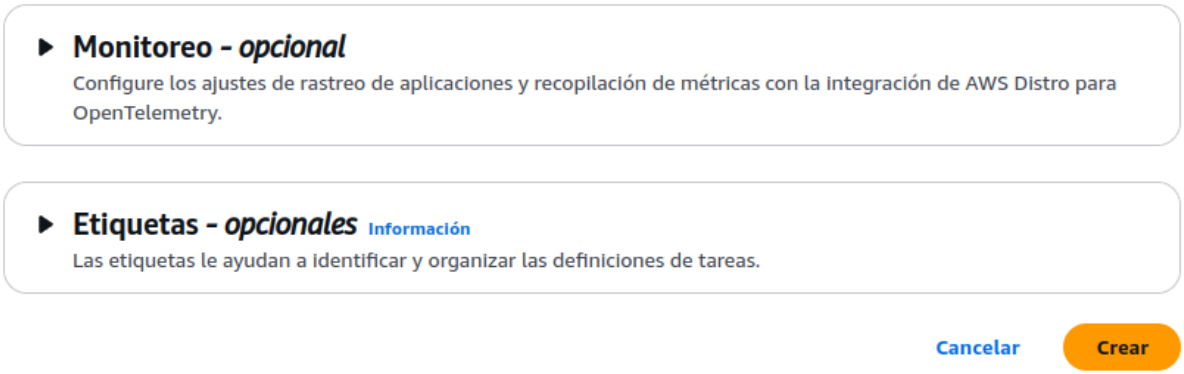


- Busca y selecciona la definición de tarea correspondiente al servicio (por ejemplo, `cvu-back-prod`).

2. Crear una Nueva Revisión de la Definición de Tarea

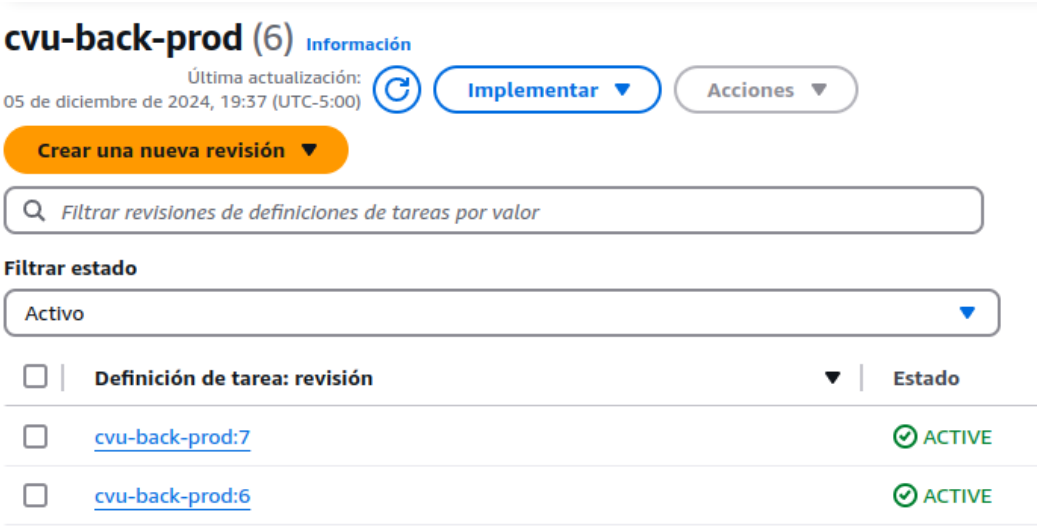
- Haz clic en **Create new revision** para crear una nueva versión de la definición existente.

Nota: Las definiciones de tarea para ambos microservicios ya están correctamente configuradas. Por lo tanto, no es necesario realizar ajustes adicionales en esta sección. Simplemente navega hasta el último ítem de la configuración y haz clic en Crear nueva versión para generar una nueva revisión de la definición de tarea.



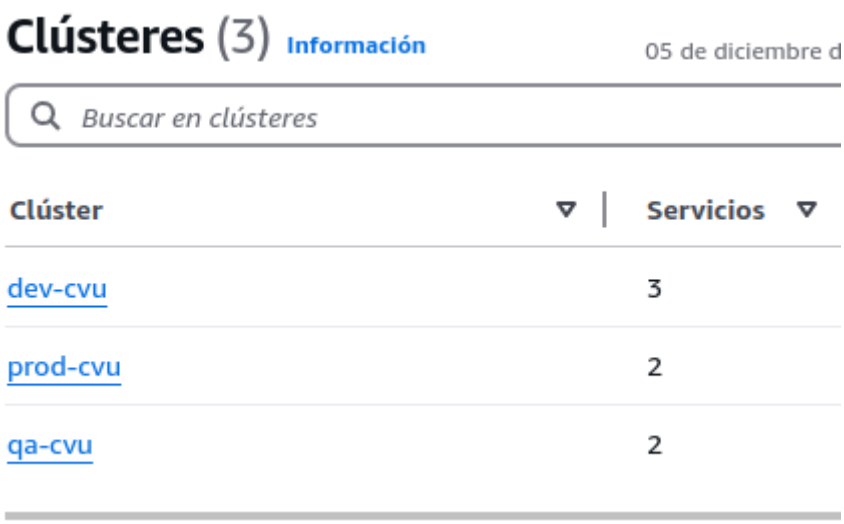
3. Confirmar la Nueva Revisión

- Asegúrate de que la revisión creada esté listada en la sección de definiciones de tareas, con el número de revisión incrementado (por ejemplo, `cvu-back-prod:7`).

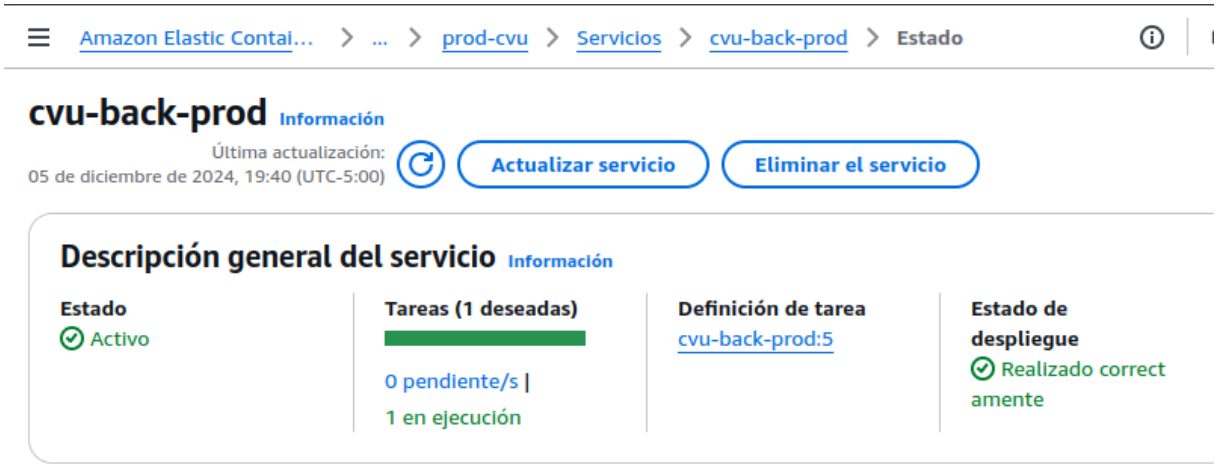


Paso 3: Actualizar Manualmente el Servicio en ECS

1. **Acceder al clúster en ECS:**
   - En la consola de AWS, navega a **Elastic Container Service (ECS)**.
   - Selecciona el clúster donde está desplegado el servicio (por ejemplo, prod-cvu).

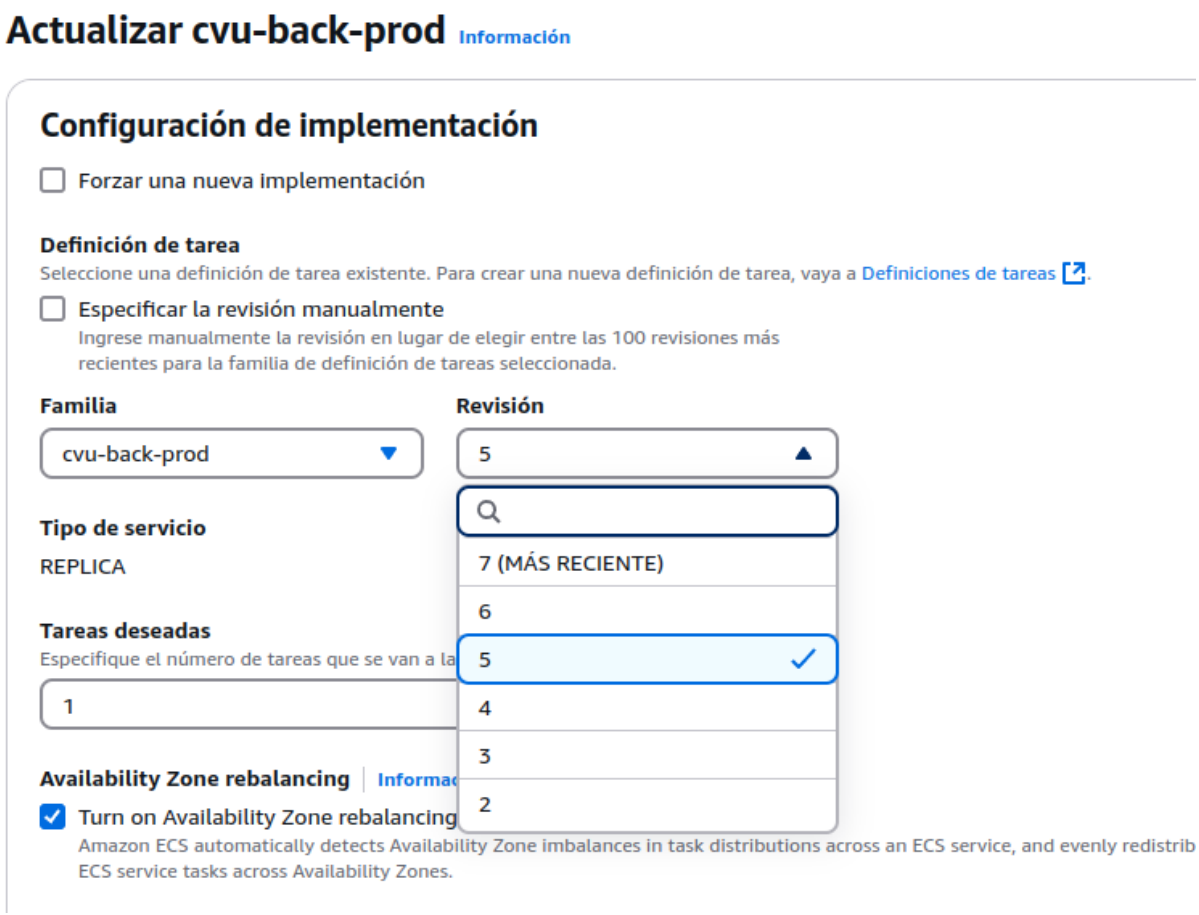


2. **Seleccionar el servicio a actualizar:**
   - Dentro del clúster, localiza el servicio que requiere la actualización, como cvu-back-prod.
   - Haz clic en el nombre del servicio para acceder a su configuración.



3. **Actualizar la definición de tarea:**
   - Haz clic en el botón **Update Service**.

- En la sección **Task Definition**, selecciona la última definición de tarea o crea una nueva si es necesario.

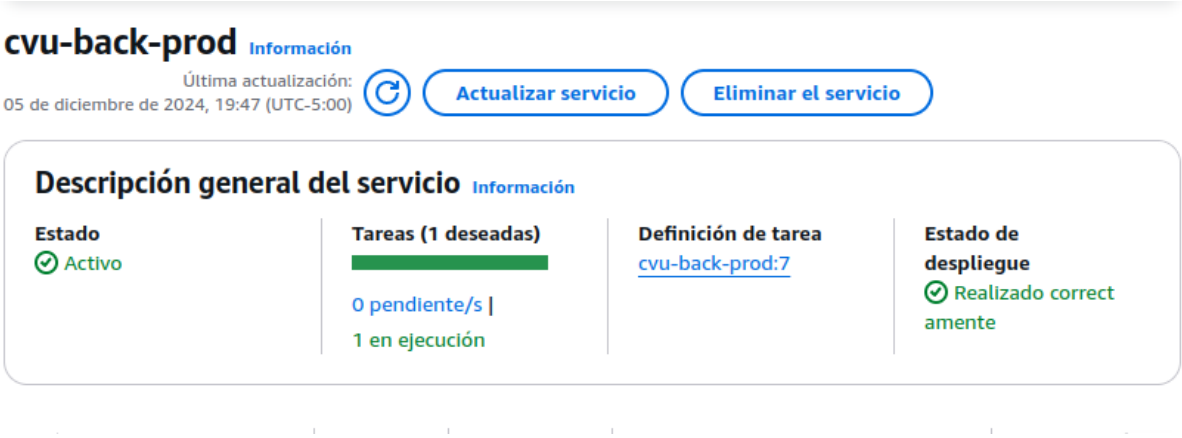


Asegúrate de que la nueva definición apunte a la imagen correcta en ECR. Por ejemplo: 573165592496.dkr.ecr.us-east-1.amazonaws.com/cvu-back-prod:latest

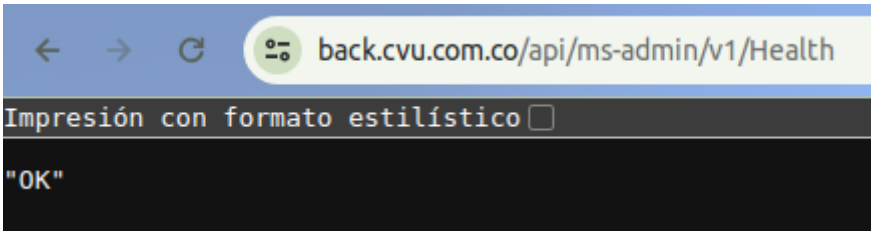
4. **Desplegar los cambios:**
   - Revisa los cambios propuestos y haz clic en **Deploy**.
   - AWS ECS iniciará nuevas tareas con la versión actualizada de la imagen y gradualmente detendrá las antiguas.

Paso 4: Validar el Despliegue

1. **Monitorear el progreso en ECS**:
    - En la consola de AWS ECS, revisa la pestaña **Events** del servicio para asegurarte de que las nuevas tareas se inicien correctamente.
    - Verifica que todas las tareas antiguas sean reemplazadas por las nuevas.



2. **Realizar pruebas de validación**:
    - Accede a la aplicación o API para confirmar que las funcionalidades principales funcionan correctamente.
    - Asegúrate de probar rutas críticas y endpoints relevantes.



3. **Revisar logs en CloudWatch**:
    - Ve a **CloudWatch Logs** y consulta los logs generados por las nuevas tareas para identificar posibles errores o comportamientos inusuales.

Manejo de Problemas

1. Rollback en caso de errores graves

- Si el despliegue presenta fallos críticos, puedes realizar un *rollback* actualizando nuevamente el servicio con la versión anterior de la imagen almacenada en ECR.

Mejores Prácticas

1. **Pruebas exhaustivas en QA**:
    - Siempre realiza pruebas completas en un entorno de QA antes de desplegar en producción.
2. **Monitorización proactiva**:
    - Mantén un monitoreo continuo del sistema para identificar y resolver problemas rápidamente.

Conclusión

El despliegue de una nueva versión en producción es un proceso crítico que debe realizarse de manera controlada para garantizar la estabilidad del sistema. Esta guía proporciona instrucciones claras y detalladas para completar el proceso con éxito. Para cualquier consulta o soporte adicional, contacta al equipo de infraestructura.

DOUBLE V PARTNERS

Guía Completa para Solucionar Errores y Configurar SonarQube

Elaborado por:

Equipo DVP

Versión:

1.0

Fecha de elaboración:

15 de Diciembre de 2024

Índice

Introducción

En el proyecto CVU, utilizamos SonarQube como una herramienta clave para garantizar la calidad del código y detectar problemas potenciales en el desarrollo de nuestros microservicios. Sin embargo, en ocasiones, SonarQube puede presentar errores que afectan su funcionamiento normal. Entre los problemas más comunes, se encuentran el cambio inesperado de la dirección IP del servicio y el restablecimiento de credenciales a los valores predeterminados.

Este documento tiene como objetivo guiar al equipo en la solución de estos problemas, desde acceder al servicio hasta configurar nuevos proyectos en SonarQube. Además, se incluyen pasos claros para generar un token de autenticación y vincularlo con un repositorio de código. La guía está diseñada para que cualquier miembro del equipo pueda seguir los pasos y garantizar que SonarQube esté operativo para los análisis necesarios.

***Nota**: sonarqube solo está en el ambiente de desarrollo.*

Problemas Comunes en SonarQube

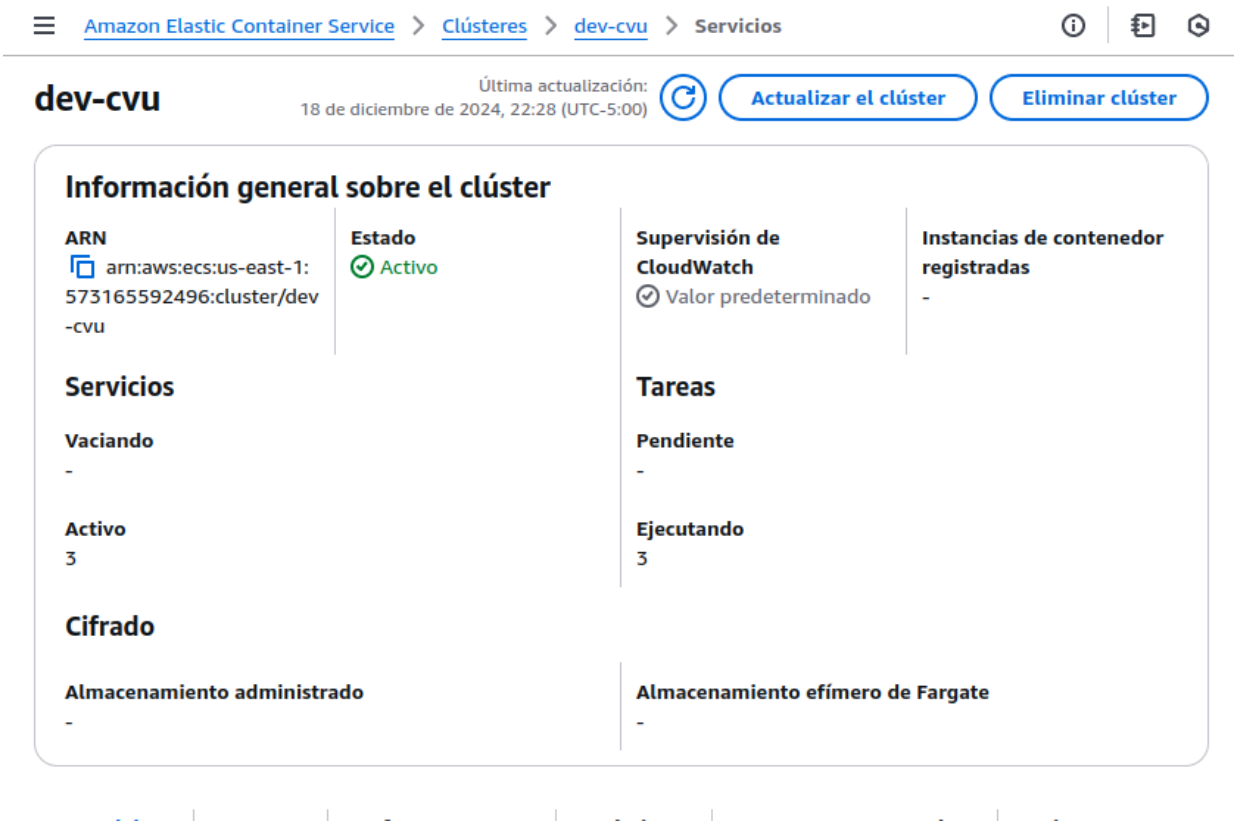
- **Cambio de IP del Servicio**
    - SonarQube puede cambiar su IP si no tiene un certificado SSL configurado. Esto sucede especialmente si el servicio está alojado en un entorno dinámico, como un clúster ECS.
- **Credenciales Restablecidas**
    - En algunos casos, las credenciales personalizadas para acceder a SonarQube se eliminan, y el sistema vuelve a usar el nombre de usuario y la contraseña predeterminados:
        - ***Usuario:** admin*
        - ***Contraseña:** admin*

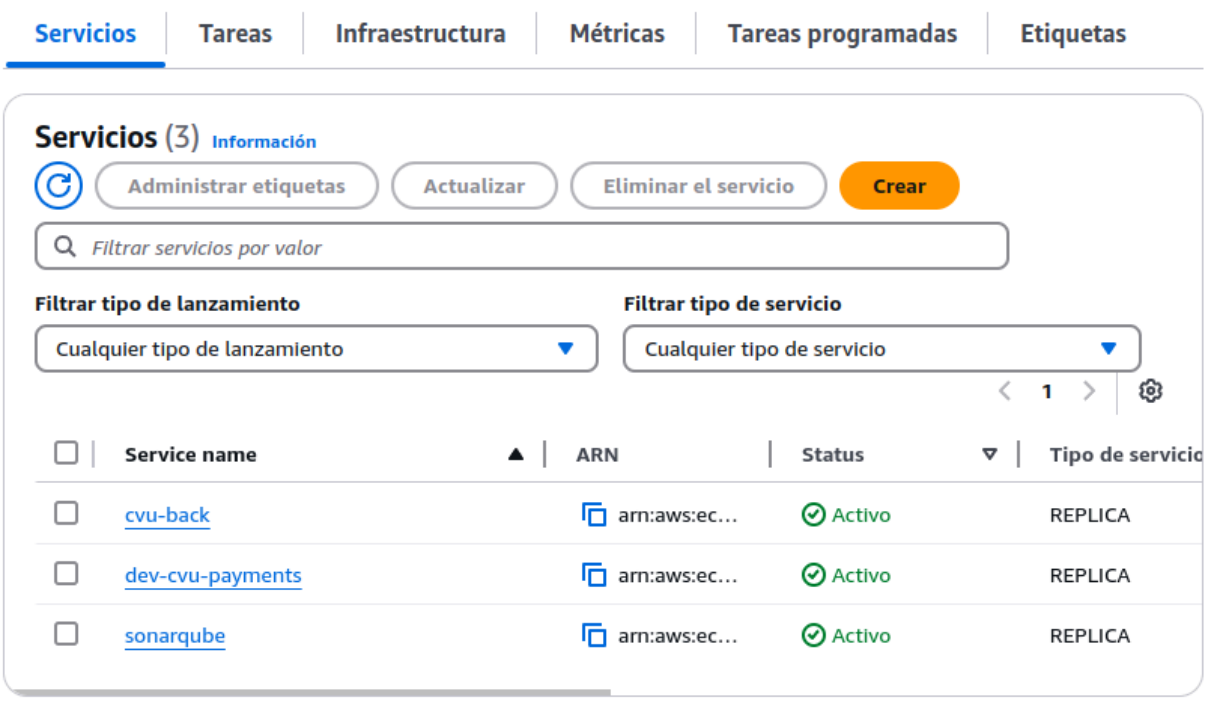
- **Acceso al Servicio**
    - La dirección para acceder a SonarQube incluye la IP del servicio seguida del puerto *9000*. Si no puedes acceder, es probable que la IP haya cambiado.

Solución de Problemas y Configuración

Paso 1: Identificar la Nueva IP del Servicio

1. **Accede a la Consola de AWS**:
    - Inicia sesión en la consola de **Amazon Web Services (AWS)**.
2. **Localiza el Servicio de SonarQube**:
    - Dirígete a **Elastic Container Service (ECS)**.
    - Busca el clúster donde está desplegado SonarQube y selecciona el servicio correspondiente.

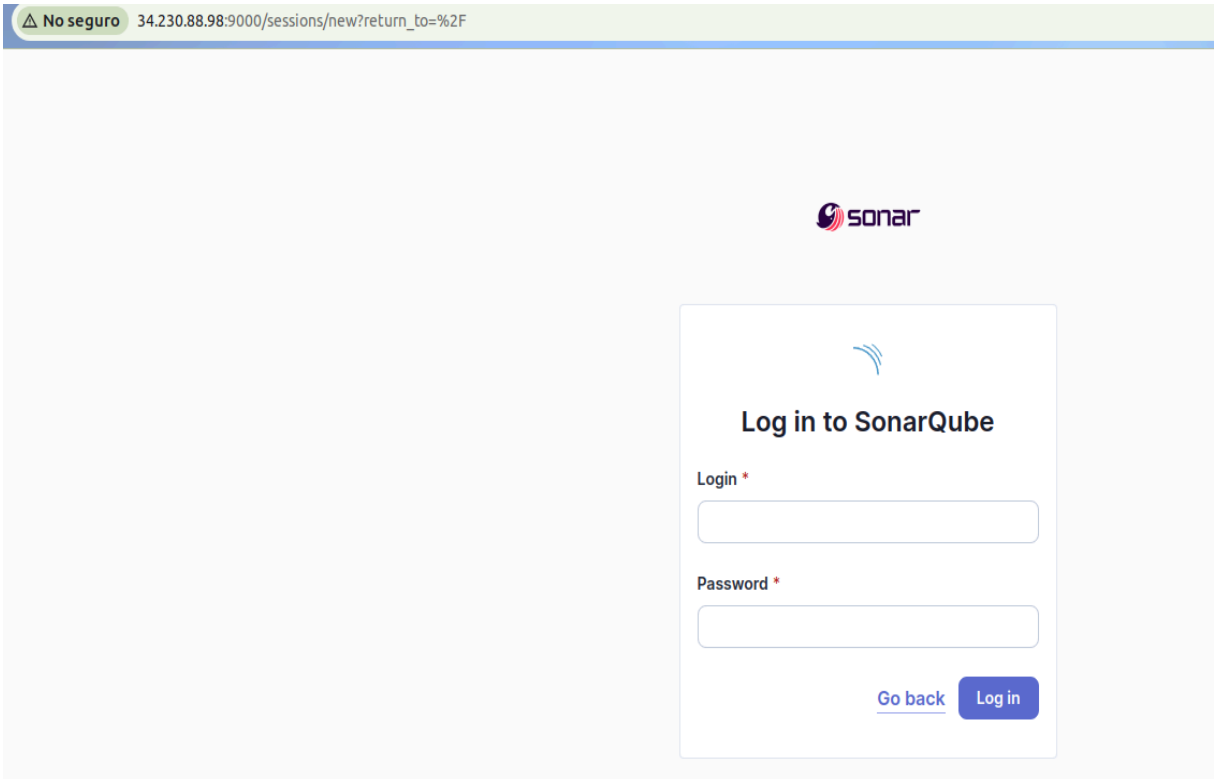




3. **Obtener la IP del Servicio**:
    - Revisa la pestaña **Tasks** dentro del servicio y localiza la dirección IP pública asociada a la tarea de SonarQube.

4. **Accede a la Interfaz de SonarQube**:

En tu navegador, escribe la dirección obtenida seguida del puerto 9000. Por ejemplo:
http://34.230.88.98:9000



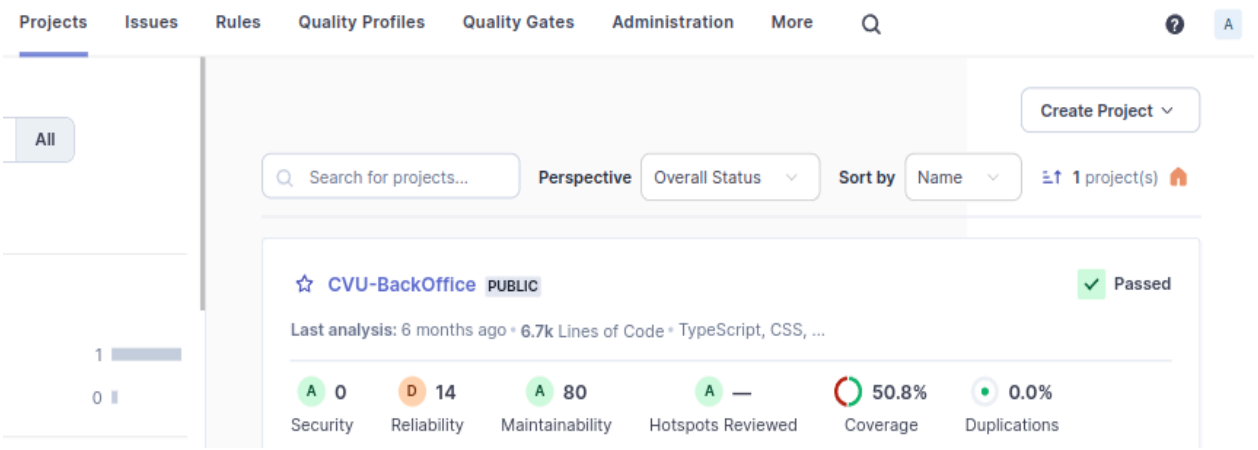
Paso 2: Acceder a SonarQube

1. **Ingresar las Credenciales**:
    - Si SonarQube ha restablecido las credenciales, usa las siguientes:
        - ***Usuario:*** *admin*
        - ***Contraseña:*** *admin*
    - Es importante tener en cuenta las credenciales que se enviaron por la documentación de accesos a Sonarqube.
2. **Resolver Problemas de Acceso**:

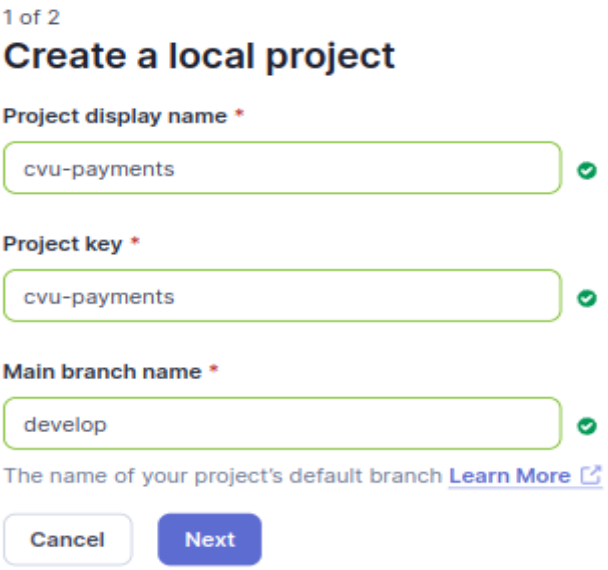
- Si no puedes acceder con las credenciales predeterminadas, prueba restablecerlas directamente desde el servicio o consulta con el administrador del sistema.

Paso 3: Crear un Nuevo Proyecto en SonarQube

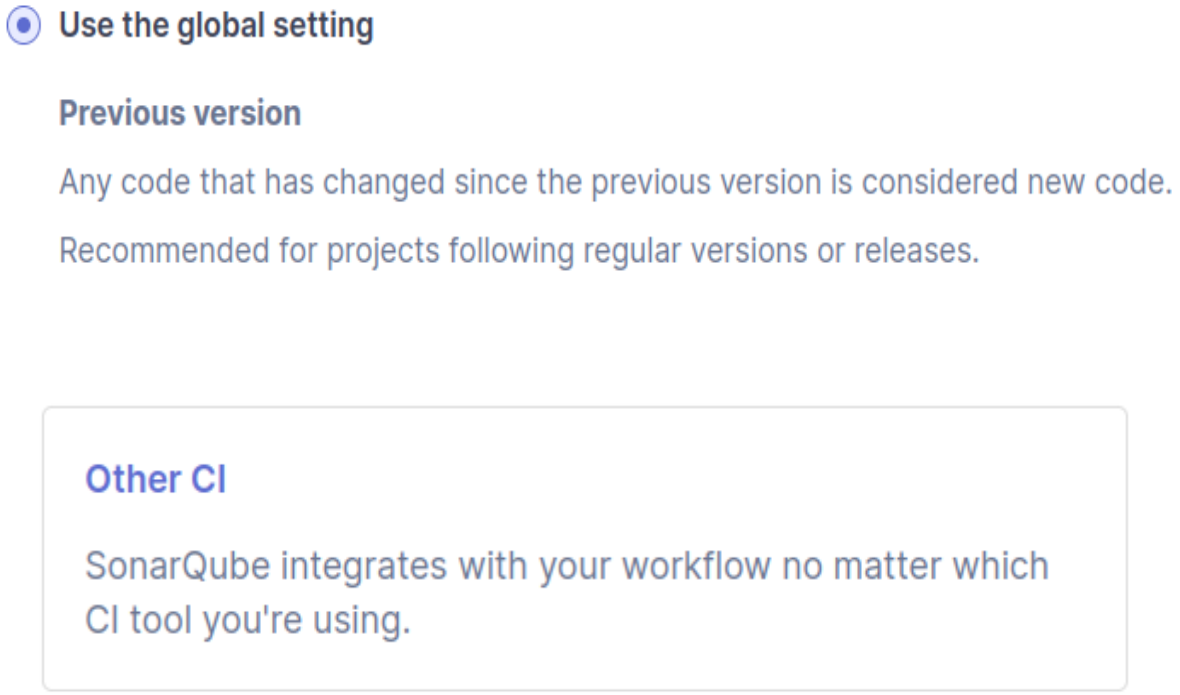
1. **Accede al Panel de Control**:
   - Una vez dentro de SonarQube, haz clic en **Projects** en el menú principal.



   - Selecciona **Create Project**.
2. **Configurar el Proyecto**:
   - Ingresa un nombre para tu proyecto (por ejemplo, cvu-payments).

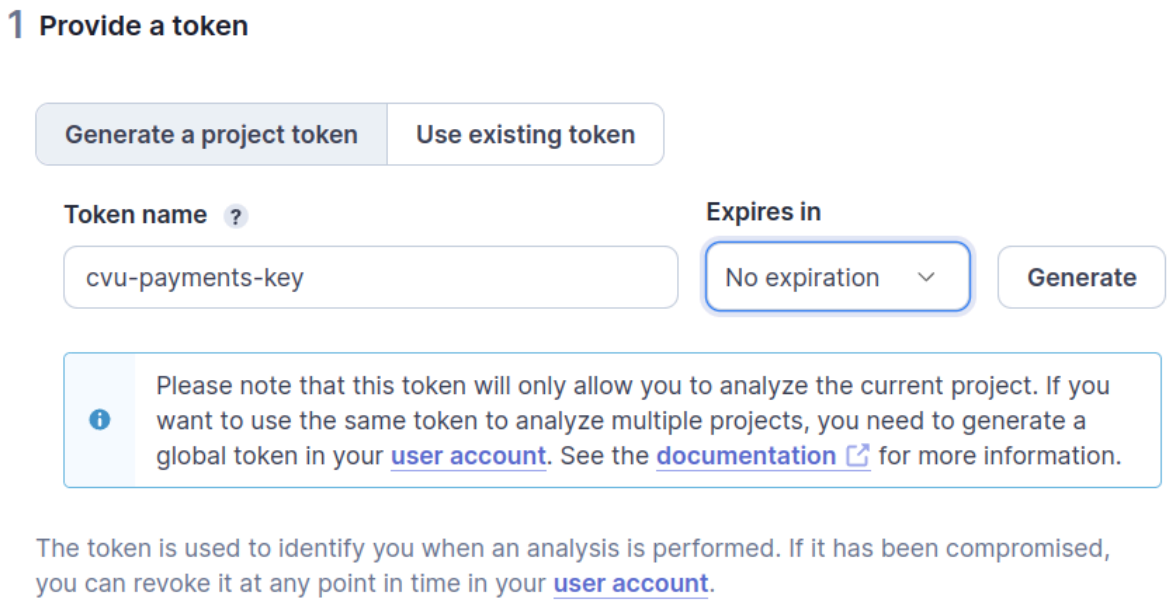


- Define un **Key** (clave única) para el proyecto (por ejemplo, cvu-payments).
- Haz clic en **Set Up** para continuar.

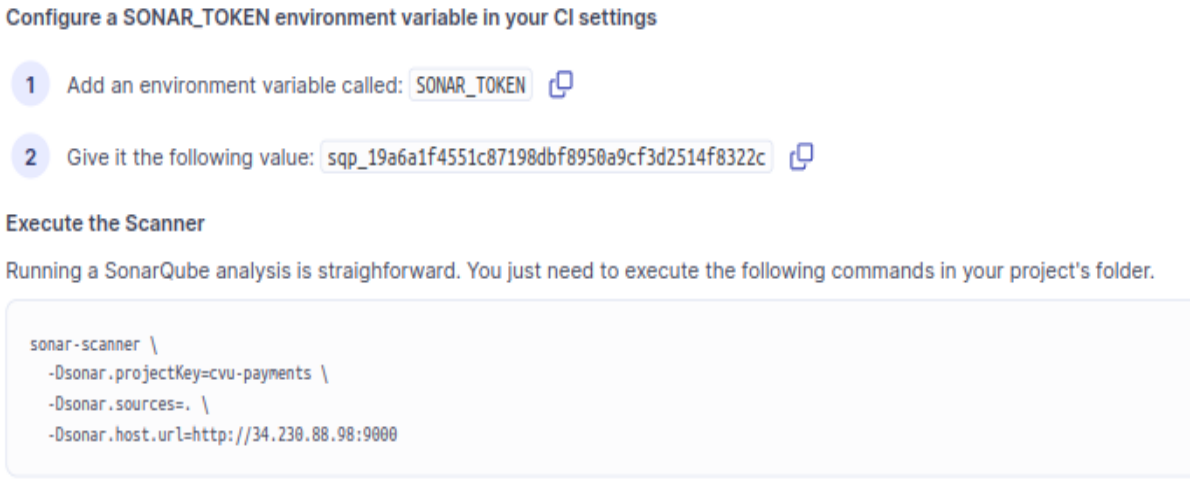


Paso 4: Generar un Token de Autenticación

1. **Generar el Token**:
   - Durante el proceso de configuración del proyecto, SonarQube pedirá que generes un **token de autenticación**.
   - Haz clic en **Generate** y copia el token que aparece en pantalla. Este token es necesario para integrar el proyecto con tu repositorio de código.



2. **Guardar el Token**:
    - Asegúrate de guardar este token en un lugar seguro, ya que no podrás recuperarlo nuevamente. Si lo pierdes, tendrás que generar uno nuevo.



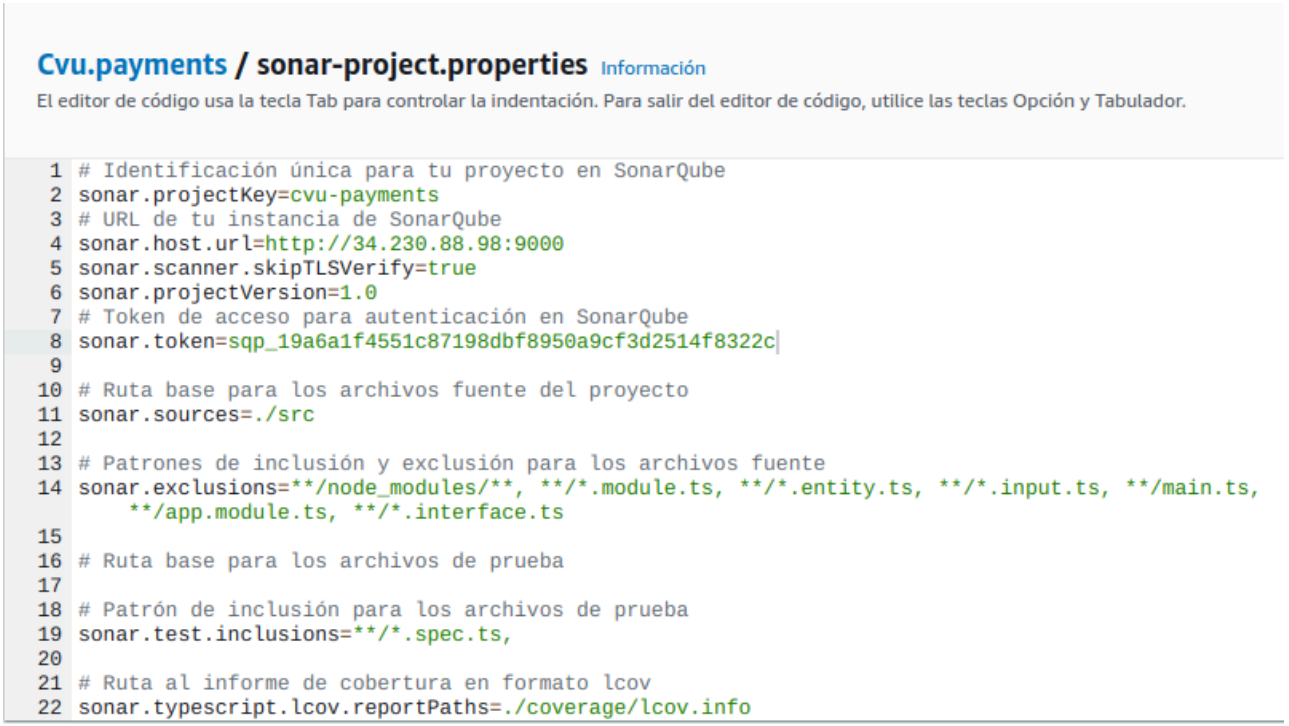
Paso 5: Configurar el Repositorio con el Token

1. **Accede al Repositorio de Código**:
    - Inicia sesión en AWS e ingresa a codecommit.

2. **Agregar el Token**:
    - Navega a la configuración del repositorio y busca la sección de **Variables de Entorno** o **Secrets**.
    - Crea una nueva variable con un nombre como *SONAR_TOKEN* y pega el token generado en SonarQube.

3. **Configurar el Archivo sonar-project.properties**:

En el directorio raíz de tu proyecto en el ambiente dev, edita el archivo sonar-project.properties con el siguiente contenido:



Cvu.payments / sonar-project.properties Información

El editor de código usa la tecla Tab para controlar la indentación. Para salir del editor de código, utilice las teclas Opción y Tabulador.

```
# Identificación única para tu proyecto en SonarQube
sonar.projectKey=cvu-payments
# URL de tu instancia de SonarQube
sonar.host.url=http://34.230.88.98:9000
sonar.scanner.skipTLSVerify=true
sonar.projectVersion=1.0
# Token de acceso para autenticación en SonarQube
sonar.token=sqp_19a6a1f4551c87198dbf8950a9cf3d2514f8322c

# Ruta base para los archivos fuente del proyecto
sonar.sources=./src

# Patrones de inclusión y exclusión para los archivos fuente
sonar.exclusions=**/node_modules/**, **/*.module.ts, **/*.entity.ts, **/*.input.ts, **/main.ts,
    **/app.module.ts, **/*.interface.ts

# Ruta base para los archivos de prueba

# Patrón de inclusión para los archivos de prueba
sonar.test.inclusions=**/*.spec.ts,

# Ruta al informe de cobertura en formato lcov
sonar.typescript.lcov.reportPaths=./coverage/lcov.info
```

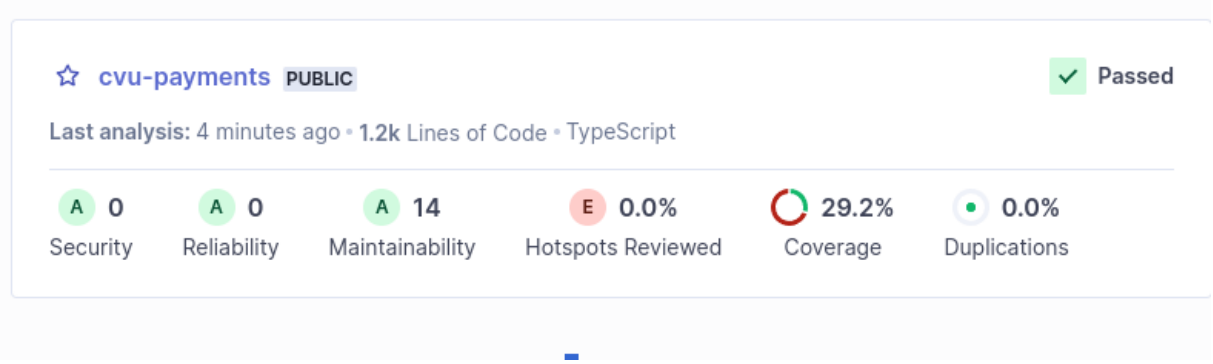
Paso 6: Validar la Configuración

1. **Ejecutar el Análisis**:

Cuando editas los valores del archivo la automatización que ya está implementada permite ejecutar los comandos necesarios para realizar el sonar-scanner.

- Asegúrate de que el análisis se complete sin errores.

2. **Revisar el Resultado en SonarQube**:
   - Vuelve a la interfaz de SonarQube y verifica que el proyecto haya sido analizado correctamente.



Conclusión

Esta guía te brinda un procedimiento claro para solucionar problemas comunes en SonarQube, acceder al sistema y configurar un proyecto nuevo. Aunque SonarQube puede presentar inconvenientes ocasionales, su integración en el ciclo de vida del desarrollo mejora significativamente la calidad del código y la seguridad de los proyectos.

Siguiendo estos pasos, tendrás un entorno de análisis funcional y bien configurado para tus necesidades. Si encuentras más problemas, no dudes en buscar soporte adicional o contactar al administrador del sistema.

API

Servicios actuales

/submitTransaction

http://ec2-52-203-214-127.compute-1.amazonaws.com/o/cvu-pay/submitTransaction

/findByReferenceCode

http://ec2-52-203-214-127.compute-1.amazonaws.com/o/cvu-pay/ findByReferenceCode

/findByTransactionId

http://ec2-52-203-214-127.compute-1.amazonaws.com/o/cvu-pay/ findByTransactionId

La documentación de cada uno de los anteriores servicios la encuentra en:

http://ec2-52-203-214-127.compute-1.amazonaws.com/pagos-api-doc

Ejemplo request y response

1. Ejemplo request json

```
{
  "username": "comercio1",                          //Usuario de comercio de prueba
  "password": "passwordComercio1",                  //Contraseña de comercio de prueba
  "referenceCode": "VC_PAYMENT_0009",               //Codigo de referencia, debe ser único
  "ipAddress": "127.0.0.1",                         //IP del comercio
  "redirectUrl": "http://vc.com.co/retorno_paycvu",  //Url a donde se direcciona el usuario una
                                                    vez finalizada la transacción
  "confirmUrl": "https://www.cvu.com.co",           //Url no utilizada actualmente
  "payment": {                                      //Valor y descripción del servicio a pagar
    "value": 10000,
    "description": "Compra de tiquetes CTG"
  },
  "user": {                                         //Datos del cliente que hace la compra
    "firstName": "USERTEST",
    "lastName": "PORTALPRB",
```

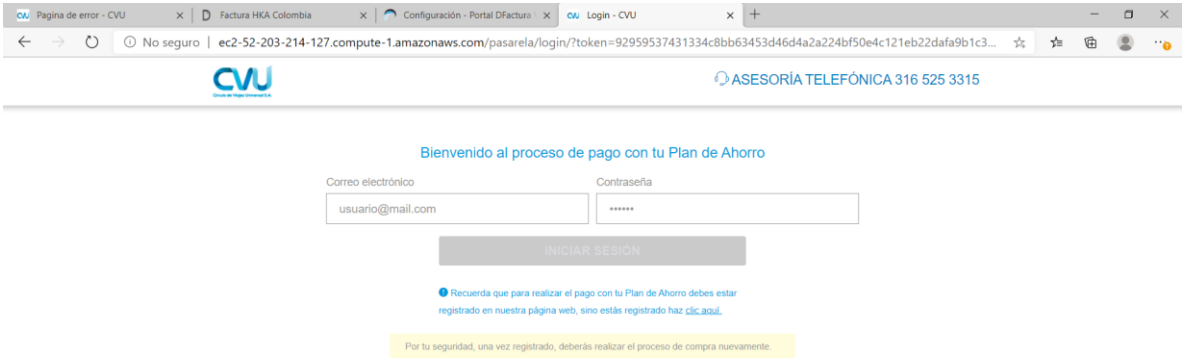
```
    "documentType": "CC",
    "documentNumber": "552111222",
    "email": "cvu.gti.qa@gmail.com",
    "cellPhone": "3152988562"
  },
  "travel": {                                   //Datos descriptivos del servicio a comprar
    "origin": "Bogota",
    "destination": "Cartagena",
    "type": "TKT"
  }
}
```


2. Ejemplo response


```
{
    "transactionId": "d26589c9-2e1f-40eb-9f04-6226bea2da21",
    "referenceCode": "VC_PAYMENT_0009",
    "url": "http://ec2-52-203-214-127.compute-
1.amazonaws.com/pasarela/login/?token=fe3110a18a27ac2640eadd58c7dc94f0cbeaa86100658f0
beef831b94b49c6e4",
    "hash": "fe3110a18a27ac2640eadd58c7dc94f0cbeaa86100658f0beef831b94b49c6e4",
    "status": "PENDING",
    "statusDetail": ""
}
```

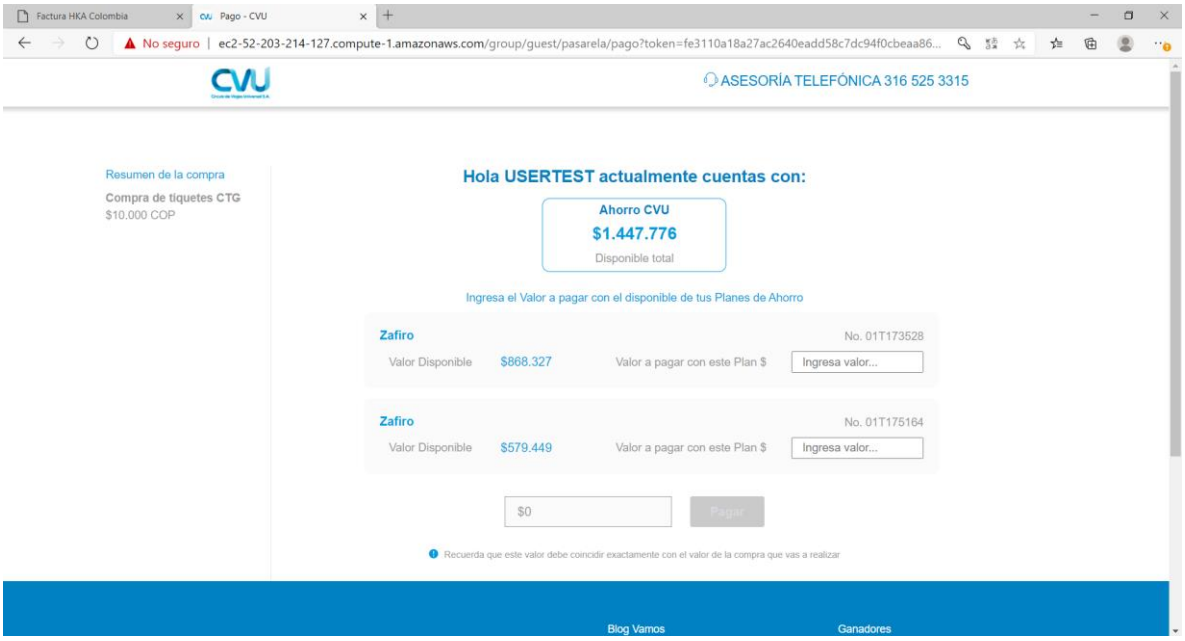
FLUJO PARA EL CLIENTE

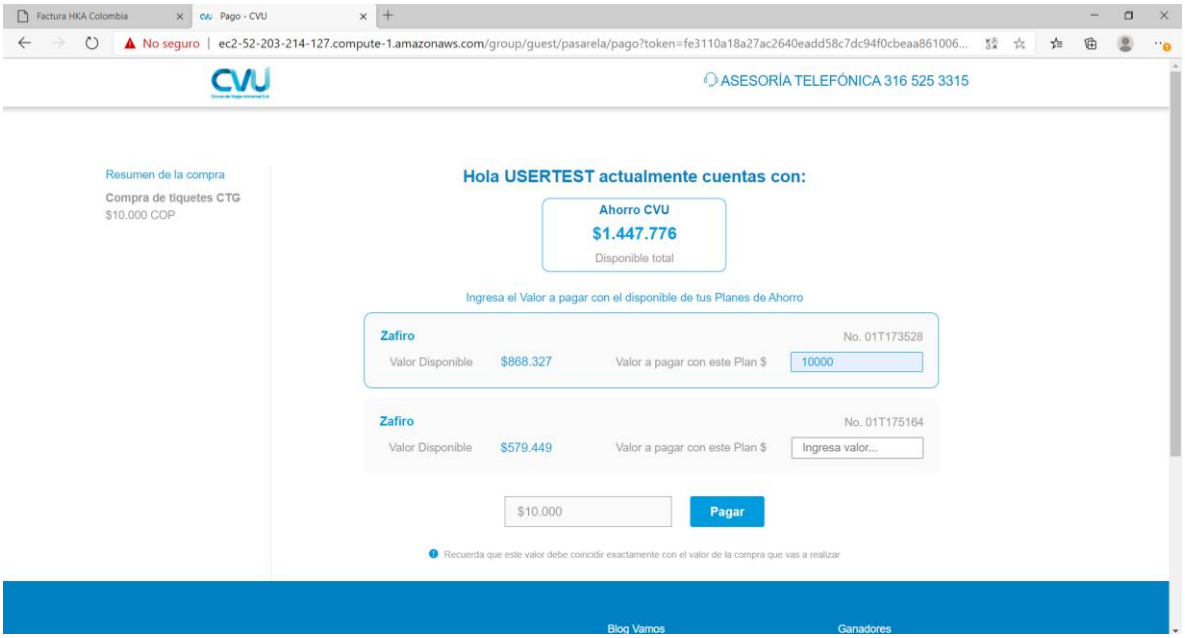
1. **Ingreso de credenciales de usuario:** Cuando el cliente seleccione como forma de pago, el ahorro con CVU, se debe hacer el consumo del servicio submitTransaction y direccionar al cliente a la url del response que lo llevara a una página como la que se muestra a continuación



//]]>

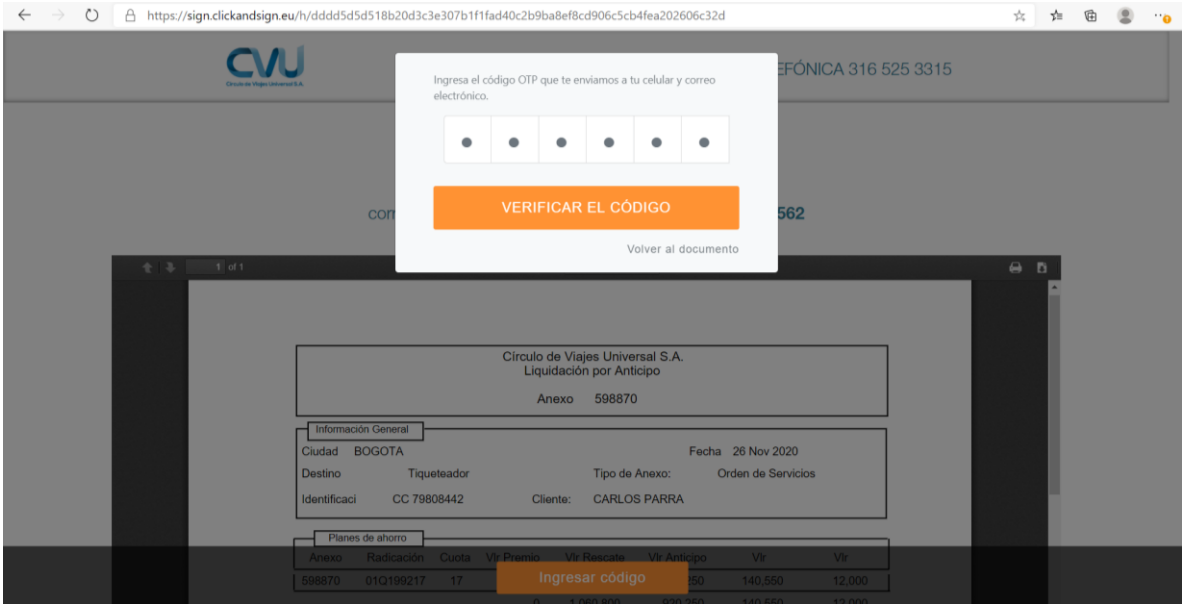
2. **Uso del ahorro:** Una vez pasados los dos filtros de seguridad, se despliega la siguiente página en donde se muestran los productos que tiene el cliente y allí el cliente debe definir de cuál producto hacer uso y dar clic en pagar.

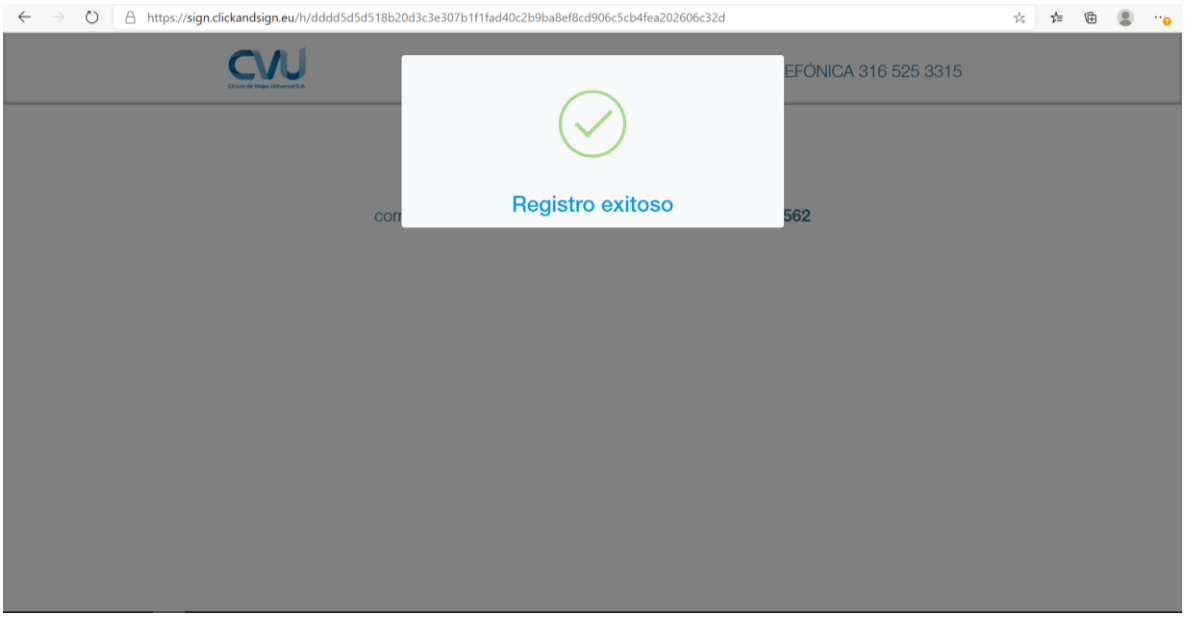




3. **Firma OTP:** Si la transacción es aprobada por nuestro sistema, se visualiza la página para que el cliente realice la firma electrónica del soporte de uso del ahorro a través de un código OTP que será enviado al correo y al celular.

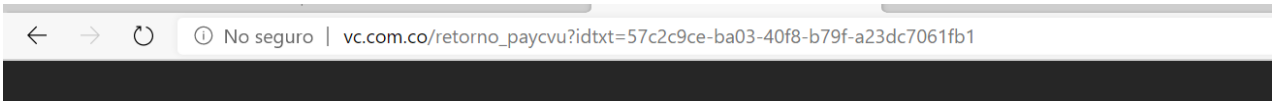
Dar clic en “Ingresar código”.


https://sign.clickandsign.eu/h/dddd5d5d518b20d3c3e307b1f1fad40c2b9ba8ef8cd906c5cb4fea202606c32d
Ingresa el código OTP que te enviamos a tu celular y correo electrónico.
VERIFICAR EL CÓDIGO
Volver al documento
Círculo de Viajes Universal S.A.
Liquidación por Anticipo
Anexo 598870
Información General
Ciudad BOGOTA
Fecha 26 Nov 2020
Destino Tiqueteador
Tipo de Anexo: Orden de Servicios
Identificaci CC 79808442
Cliente: CARLOS PARRA
Planes de ahorro
Anexo Radicación Cuota Vlr Premio Vlr Rescate Vlr Anticipo Vlr Vlr
598870 01Q199217 17 140,550 12,000
Ingresar código


https://sign.clickandsign.eu/h/dddd5d5d518b20d3c3e307b1f1fad40c2b9ba8ef8cd906c5cb4fea202606c32d
Registro exitoso

Al finalizar la transacción de forma exitosa, el cliente debe dar clic en “Volver al comercio” y será redireccionado a la url que se estableció en el atributo "redirectUrl" cuando se hizo consumo del servicio /submitTransaction, enviando como parámetro el id de la transacción.

Para este ejemplo, la url a donde será redireccionado el cliente es:



WEB SERVICES CVU

Botón de pago

Servicio: CPAV

URL: http://190.143.78.137:8484 /axis2/services/CPAV?wsdl

OPERACIONES	
Nombre	consultarMisCPAVsParaTxBotonPago
Descripción	Servicio encargado de consultar la información de los contratos de plan de ahorro que cumplen con las condiciones requeridas para la transacción a realizar, compra de servicios turísticos o compra de contrato de plan de ahorro.
Parámetros	Tipo de Dato: IdentificacionWS Nombre: idCliente Descripción: Objeto que contiene la identificación del cliente dueño de los contratos de plan de ahorro para viajes. Tipo de Dato: String Nombre: strReferenceCode Descripción: Referencia única, asignada a la transacción. Tipo de Dato: DatosValidacionWS Nombre: dtValidacionWS Descripción: Objeto que tiene la información que permite o no poder utilizar el servicio de consultarMisCPAVs.
Retorna	Dato de tipo DatosMisProductosWS que contiene la información resumen de los productos de un cliente y el listado de los productos. Sus atributos son • cantidadContratos • valorTotalAhorrado • valorTotalDisponible • DatosProductoWS infoProductos[]:Arreglo que contiene objetos de tipo DatosProductoWS con la información del contrato. El detalle de este objeto esta descrito en la operación ***consultarMiCPAV***
Datos para prueba	Identificación Código tipo:1 Descripción tipo: CC Numero:79808442 DatosValidacion Usuario:cvuws Password: cVu8gOoZ9oOzIWS

OPERACIONES	
Nombre	afectarCPAVPorReferencia
Descripción	Servicio encargado de realizar el débito del disponible de un CPAV cuando se realiza una compra de servicio turístico a través del botón de pago o compra de un contrato de plan de ahorro.
Parámetros	Arreglo de uno o varios objetos de tipo DatosAnexoWS Descripción: DatosAnexoWS es un objeto que contiene: valorTotalUtilizado: Valor usado del contrato para adquirir el nuevo contrato. dtTipoAnexoWS o Código o Descripción Si el tipo de anexo viene con valor 1, corresponde a una venta de un nuevo CPAV. Si el tipo de anexo viene con valor 3, corresponde a una compra de servicios turísticos. strPoliza: Objeto que contiene el identificador del contrato Tipo de Dato: String Nombre: strReferenceCode Descripción: Referencia única, asignada a la transacción. Tipo de Dato: DatosValidacionWS Nombre: dtValidacionWS Descripción: Objeto que tiene la información que permite o no poder utilizar el servicio.
Retorna	Si el tipo de anexo viene con valor 1, en el String de retorno, viene la descripción del anexo generado, por ejemplo “1-102030” Si el tipo de anexo viene con valor 3, en el String de retorno, contiene un arreglo de bytes con la información de un archivo con formato PDF.
Datos para prueba	DatosAnexoWS • valorTotalUtilizado= 5000 • dtTipoAnexoWS ▪ código=1 ▪ descripción = Pago alojamiento Cartagena • strPoliza = 01Q191633 strReferenceCode • REF_TESTCLEZ_XXX (XXX Valor entre 000 y 999) DatosValidacion • Usuario:cvuws • Password: cVu8gOoZ9oOzIWS

WEB SERVICES CVU

Servicio: Clientes

URL: http://190.143.78.137:8484/axis2/services/Clientes?wsdl

OPERACIONES	
Nombre	consultarCliente
Descripción	Servicio encargado de consultar la información de un cliente.
Parámetros	Tipo de Dato: String Nombre: strTipoId Descripción: Objeto que contiene el tipo de identificación de un cliente. Tipo de Dato: long Nombre: numeroId Descripción: Variable que contiene el número de identificación. Tipo de Dato: DatosValidacionWS Nombre: dtValidacionWS Descripción: Objeto que tiene la información que permite o no poder utilizar el servicio de consultarCliente.
Retorna	Objetos de tipo DatosClienteWS, que contiene toda la información relacionada con un cliente en particular.
Datos para prueba	Tipo de identificación Valor CC. Numero:79808442 DatosValidacion Usuario:cvuws Password: cVu8gOoZ9oOzIWS

OPERACIONES	
Nombre	generarCertificacionCliente
Descripción	Servicio encargado de generar un arreglo de bytes a partir de un archivo con formato PDF.
Parámetros	Tipo de Dato: String Nombre: strTipoId Descripción: Objeto que contiene el tipo de identificación de un cliente. Tipo de Dato: long Nombre: numeroId Descripción: Variable que contiene el número de identificación. Tipo de Dato: DatosValidacionWS Nombre: dtValidacionWS Descripción: Objeto que tiene la información que permite o no poder utilizar el servicio de generarCertificacionCliente.
Retorna	String que contiene un arreglo de bytes con la información de un archivo con formato PDF.
Datos para prueba	Tipo de identificación Valor CC.

	Numero:79808442 DatosValidacion Usuario:cvuws Password: cVu8gOoZ9oOzIWS

OPERACIONES	
Nombre	generarEstadoDeCuentaCliente
Descripción	Servicio encargado de generar un arreglo de bytes a partir de un archivo con formato PDF.
Parámetros	Tipo de Dato: String Nombre: strTipoId Descripción: Objeto que contiene el tipo de identificación de un cliente. Tipo de Dato: long Nombre: numeroId Descripción: Variable que contiene el número de identificación. Tipo de Dato: DatosValidacionWS Nombre: dtValidacionWS Descripción: Objeto que tiene la información que permite o no poder utilizar el servicio de generarEstadoDeCuentaCliente.
Retorna	String que contiene un arreglo de bytes con la información de un archivo con formato PDF.
Datos para prueba	Tipo de identificación Valor CC. Numero:79808442 DatosValidacion Usuario:cvuws Password: cVu8gOoZ9oOzIWS

OPERACIONES	
Nombre	generarCertificacionClienteEmbajada
Descripción	Servicio encargado de generar un arreglo de bytes a partir de un archivo con formato PDF.
Parámetros	Tipo de Dato: String Nombre: strTipoId Descripción: Objeto que contiene el tipo de identificación de un cliente. Tipo de Dato: long Nombre: numeroId Descripción: Variable que contiene el número de identificación. Tipo de Dato: DatosValidacionWS Nombre: dtValidacionWS Descripción: Objeto que tiene la información que permite o no poder utilizar el servicio de generarCertificacionClienteEmbajada.
Retorna	String que contiene un arreglo de bytes con la información de un archivo con formato PDF.

Datos para prueba	Tipo de identificación Valor CC. Numero:79808442 DatosValidacion Usuario:cvuws Password: cVu8gOoZ9oOzIWS

OPERACIONES	
Nombre	generarCertificacionClienteTributaria
Descripción	Servicio encargado de generar un arreglo de bytes a partir de un archivo con formato PDF.
Parámetros	Tipo de Dato: String Nombre: strTipoId Descripción: Objeto que contiene el tipo de identificación de un cliente. Tipo de Dato: long Nombre: numeroId Descripción: Variable que contiene el número de identificación. Tipo de Dato: DatosValidacionWS Nombre: dtValidacionWS Descripción: Objeto que tiene la información que permite o no poder utilizar el servicio de generarCertificacionClienteTributaria.
Retorna	String que contiene un arreglo de bytes con la información de un archivo con formato PDF.
Datos para prueba	Tipo de identificación Valor CC. Numero:79808442 DatosValidacion Usuario:cvuws Password: cVu8gOoZ9oOzIWS

OPERACIONES	
Nombre	generarFormularioReintegro
Descripción	Servicio encargado de generar un arreglo de bytes a partir de un archivo con formato PDF.
Parámetros	Tipo de Dato: String Nombre: strTipoId Descripción: Objeto que contiene el tipo de identificación de un cliente. Tipo de Dato: long Nombre: numeroId Descripción: Variable que contiene el número de identificación. Tipo de Dato: DatosValidacionWS Nombre: dtValidacionWS Descripción: Objeto que tiene la información que permite o no poder utilizar el servicio de generarFormularioReintegro.

Retorna	String que contiene un arreglo de bytes con la información de un archivo con formato PDF.
Datos para prueba	Tipo de identificación Valor CC. Numero:79808442 DatosValidacion Usuario:cvuws Password: cVu8gOoZ9oOzIWS

OPERACIONES	
Nombre	obtenerAccesoPseCliente
Descripción	Servicio encargado de generar una URL codificada en Base 64.
Parámetros	Tipo de Dato: String Nombre: strTipoId Descripción: Objeto que contiene el tipo de identificación de un cliente. Tipo de Dato: long Nombre: numeroId Descripción: Variable que contiene el número de identificación. Tipo de Dato: DatosValidacionWS Nombre: dtValidacionWS Descripción: Objeto que tiene la información que permite o no poder utilizar el servicio de obtenerAccesoPseCliente.
Retorna	Objeto de tipo DatosAccesoPSEWS que contiene: • Codigo hash generado a partir de los milisegundos de la fecha actual. • URL de Acceso. Esta URL lleva al cliente al portal https://pse.cvu.com.co/ y le permite realizar el aporte por PSE.
Datos para prueba	Tipo de identificación Valor CC. Numero:79808442 DatosValidacion Usuario:cvuws Password: cVu8gOoZ9oOzIWS

OPERACIONES	
Nombre	generarTokenBeneficiosCliente
Descripción	Servicio encargado de generar un token para poder tener acceso a la plataforma Tus Beneficios.
Parámetros	Tipo de Dato: String Nombre: strTipoId Descripción: Objeto que contiene el tipo de identificación de un cliente. Tipo de Dato: long Nombre: numeroId Descripción: Variable que contiene el número de identificación. Tipo de Dato: DatosValidacionWS Nombre: dtValidacionWS Descripción: Objeto que tiene la información que permite o no poder utilizar el servicio de generarTokenBeneficiosCliente.
Retorna	String que contiene el token que permite el acceso a la plataforma Tus Beneficios
Datos para prueba	Tipo de identificación Valor CC. Numero:79808442 DatosValidacion Usuario:cvuws Password: cVu8gOoZ9oOzIWS

OPERACIONES	
Nombre	generarUrlAccesoCVUPlatinum
Descripción	Servicio encargado de generar un token para poder tener acceso a la plataforma de CVU Platinum.
Parámetros	Tipo de Dato: String Nombre: strTipoId Descripción: Objeto que contiene el tipo de identificación de un cliente. Tipo de Dato: long Nombre: numeroId Descripción: Variable que contiene el número de identificación. Tipo de Dato: DatosValidacionWS Nombre: dtValidacionWS Descripción: Objeto que tiene la información que permite o no poder utilizar el servicio de generarUrlAccesoCVUPlatinum.
Retorna	String que contiene el token que permite el acceso a la plataforma CVU Platinum
Datos para prueba	Tipo de identificación Valor CC. Numero:79808442

	DatosValidacion Usuario:cvuws Password: cVu8gOoZ9oOzIWS

OPERACIONES	
Nombre	clienteEsCVUPlatinum
Descripción	Servicio encargado de validar si un cliente pertenece al programa CVU Platinum.
Parámetros	Tipo de Dato: String Nombre: strTipoId Descripción: Objeto que contiene el tipo de identificación de un cliente. Tipo de Dato: long Nombre: numeroId Descripción: Variable que contiene el número de identificación. Tipo de Dato: DatosValidacionWS Nombre: dtValidacionWS Descripción: Objeto que tiene la información que permite o no poder utilizar el servicio de generarUrlAccesoCVUPlatinum.
Retorna	Boolean que indica si el cliente es catalogado como CVU Platinum. Si es verdadero, entonces puede tener acceso a la plataforma CVU Platinum. De lo contrato no puede tener acceso.
Datos para prueba	Tipo de identificación Valor CC. Numero:79808442 DatosValidacion Usuario:cvuws Password: cVu8gOoZ9oOzIWS

OPERACIONES	
Nombre	clienteTieneBeneficios
Descripción	Servicio encargado de validar si un cliente pertenece a la campaña AHORRA MÁS – GANA MÁS.
Parámetros	Tipo de Dato: String Nombre: strTipoId Descripción: Objeto que contiene el tipo de identificación de un cliente. Tipo de Dato: long Nombre: numeroId Descripción: Variable que contiene el número de identificación.

	Tipo de Dato: DatosValidacionWS Nombre: dtValidacionWS Descripción: Objeto que tiene la información que permite o no poder utilizar el servicio de generarUrlAccesoCVUPlatinum.
Retorna	Boolean que indica si el cliente es pertenece a la campaña AHORRA MÁS – GANA MÁS. Si es verdadero, entonces puede tener acceso a la plataforma de beneficios De lo contrato no puede tener acceso.
Datos para prueba	Tipo de identificación Valor CC. Numero:79808442 DatosValidacion Usuario:cvuws Password: cVu8gOoZ9oOzIWS

Servicio: CPAV

URL: http://190.143.78.137:8484 /axis2/services/CPAV?wsdl

OPERACIONES	
Nombre	actualizarNombrePlan
Descripción	Servicio encargado de actualizar el nombre de uno de los planes de ahorro de un cliente.
Parámetros	Tipo de Dato: String Nombre: strPoliza Descripción: Objeto que contiene el identificador del contrato al cual se le va a realizar el aporte. Tipo de Dato: String Nombre: strNuevoNombre Descripción: Nombre que se le va asignar al contrato de plan de ahorro. Tipo de Dato: DatosValidacionWS Nombre: dtValidacionWS Descripción: Objeto que tiene la información que permite o no poder utilizar el servicio de registrarAporteAplicacion.
Retorna	Boolean que indica si se pudo registrar el nuevo nombre para el contrato de plan de ahorro.
Datos para prueba	strPoliza: 01Q191633 strNuevoNombre: Mundial 2026 DatosValidacion Usuario:cvuws Password: cVu8gOoZ9oOzIWS

OPERACIONES	
Nombre	afectarCPAV
Descripción	Servicio encargado de realizar el débito del disponible de un CPAV cuando se realiza la compra de un nuevo CPAV con la forma de pago aplicación.
Parámetros	Arreglo de uno o varios objetos de tipo DatosAnexoWS Descripción: DatosAnexoWS es un objeto que contiene: valorTotalUtilizado: Valor usado del contrato para adquirir el nuevo contrato. dtTipoAnexoWS o Código o Descripción Si el tipo de anexo viene con valor 1, corresponde a una venta de un nuevo CPAV. Si el tipo de anexo viene con valor 3, corresponde a una compra de servicios turísticos. strPoliza: Objeto que contiene el identificador del contrato Tipo de Dato: DatosValidacionWS Nombre: dtValidacionWS

	Descripción: Objeto que tiene la información que permite o no poder utilizar el servicio de afectarCPV.
Retorna	Si el tipo de anexo viene con valor 1, en el String de retorno, viene la descripción del anexo generado, por ejemplo “1-102030” Si el tipo de anexo viene con valor 3, en el String de retorno, contiene un arreglo de bytes con la información de un archivo con formato PDF.
Datos para prueba	DatosAnexoWS • valorTotalUtilizado= 5000 • dtTipoAnexoWS ▪ código=1 ▪ descripción = Venta por aplicacion. Desde la pagina WEB • strPoliza = 01Q191633 DatosValidacion Usuario:cvuws Password: cVu8gOoZ9oOzIWS

OPERACIONES	
Nombre	consultarMiCPAV
Descripción	Servicio encargado de consultar la información de un contrato de plan de ahorro.
Parámetros	Tipo de Dato: String Nombre: strPoliza Descripción: Objeto que contiene el identificador del contrato a consultar. Tipo de Dato: DatosValidacionWS Nombre: dtValidacionWS Descripción: Objeto que tiene la información que permite o no poder utilizar el servicio de consultarMiCPAV.
Retorna	Objeto de tipo DatosProductoWS que contiene toda la información relacionada a las características del contrato, historial de aportes e historial de anticipos. Los atributos del objeto son los siguientes: • tipoContrato • numeroContrato • radicacionCPAV • fechaUltimoAporte • valorAporte • valorAhorrado • valorDisponible • valorAnticipos • cantSorteosGanados • cantAportesRealizados • cantAportesPendientes • fechaInicio • fechaMetaInicial • fechaMetaEstimada

	• infoAportes[]: Arreglo que contiene objetos de tipo DatosAporteWS con la información del historial de aportes. Sus atributos son:
	  ○ numeroFactura
	  ○ numeroAporte
	  ○ fechaAporte
	  ○ valorAporte
	• DatosAnticipoWS infoAnticipos[] : Arreglo que contiene objetos de tipo DatosAnticipoWS con la información del historial de anticipos. Sus atributos son:
	  ○ numeroAporte
	  ○ fechaMovimiento
	  ○ valorUtilizado
	  ○ motivo

Datos para prueba	strPoliza: 01Q191633 DatosValidacion Usuario:cvuws Password: cVu8gOoZ9oOzIWS

OPERACIONES	
Nombre	consultarMisCPAVs
Descripción	Servicio encargado de consultar la información de los contratos de plan de ahorro de un cliente.
Parámetros	Tipo de Dato: IdentificacionWS Nombre: idCliente Descripción: Objeto que contiene la identificación del cliente dueño de los contratos de plan de ahorro para viajes. Tipo de Dato: DatosValidacionWS Nombre: dtValidacionWS Descripción: Objeto que tiene la información que permite o no poder utilizar el servicio de consultarMisCPAVs.
Retorna	Dato de tipo DatosMisProductosWS que contiene la información resumen de los productos de un cliente y el listado de los productos. Sus atributos son • cantidadContratos • valorTotalAhorrado • valorTotalDisponible • DatosProductoWS infoProductos[]:Arreglo que contiene objetos de tipo DatosProductoWS con la información del contrato. El detalle de este objeto esta descrito en la operación ***consultarMiCPAV***
Datos para prueba	Identificación Código tipo:1 Descripción tipo: CC Numero:79808442 DatosValidacion Usuario:cvuws Password: cVu8gOoZ9oOzIWS

OPERACIONES	
Nombre	consultarMisCPAVsParaVentaWeb
Descripción	Servicio encargado de consultar la información de los contratos de plan de ahorro de un cliente. En este listado vienen los contratos que pueden ser usados para adquirir un nuevo contrato con la forma de pago **Mi Ahorro CVU**.
Parámetros	Tipo de Dato: IdentificacionWS Nombre: idCliente Descripción: Objeto que contiene la identificación del cliente dueño de los contratos de plan de ahorro para viajes. Tipo de Dato: DatosValidacionWS Nombre: dtValidacionWS Descripción: Objeto que tiene la información que permite o no poder utilizar el servicio de consultarMisCPAVsParaVentaWeb.
Retorna	Dato de tipo DatosMisProductosWS que contiene la información resumen de los productos de un cliente y el listado de los productos. Sus atributos son • cantidadContratos • valorTotalAhorrado • valorTotalDisponible • DatosProductoWS infoProductos[]:Arreglo que contiene objetos de tipo DatosProductoWS con la información del contrato. El detalle de este objeto esta descrito en la operación ***consultarMiCPAV***
Datos para prueba	Identificación Código tipo:1 Descripción tipo: CC Numero:79808442 DatosValidacion Usuario:cvuws Password: cVu8gOoZ9oOzIWS

OPERACIONES	
Nombre	registrarAporteTarjetaCredito
Descripción	Servicio encargado de registrar un aporte realizado con una tarjeta de crédito.
Parámetros	Tipo de Dato: String Nombre: strPoliza Descripción: Objeto que contiene el identificador del contrato al cual se le va a realizar el aporte. Tipo de Dato: float Nombre: valorPagado Descripción: Valor del aporte. Tipo de Dato: String Nombre: identificadorTransaccion Descripción: Identificador de la transacción. Tipo de Dato: DatosValidacionWS

	Nombre: dtValidacionWS Descripción: Objeto que tiene la información que permite o no poder utilizar el servicio de registrarAporteTarjetaCredito.
Retorna	Boolean que indica si se pudo registrar el aporte con tarjeta de crédito.
Datos para prueba	strPoliza: 01Q191633 valorPagado: 158925 identificadorTransaccion: XXABC123 DatosValidacion Usuario:cvuws Password: cVu8gOoZ9oOzIWS

OPERACIONES	
Nombre	consultarMisCPAVsParaPagoAplicacion
Descripción	Servicio encargado de consultar la información de los contratos de plan de ahorro de un cliente. En este listado vienen los contratos que pueden ser usados para realizar un aporte con la forma de pago **Aplicacion**.
Parámetros	Tipo de Dato: IdentificacionWS Nombre: idCliente Descripción: Objeto que contiene la identificación del cliente dueño de los contratos de plan de ahorro para viajes. Tipo de Dato: String Nombre: strRadicacionAPagar Descripción: Objeto que contiene el identificador del contrato al cual se le va a realizar el aporte Tipo de Dato: DatosValidacionWS Nombre: dtValidacionWS Descripción: Objeto que tiene la información que permite o no poder utilizar el servicio de consultarMisCPAVsParaPagoAplicacion.
Retorna	Dato de tipo DatosMisProductosWS que contiene la información resumen de los productos de un cliente y el listado de los productos. Sus atributos son • cantidadContratos • valorTotalAhorrado • valorTotalDisponible • DatosProductoWS infoProductos[]:Arreglo que contiene objetos de tipo DatosProductoWS con la información del contrato. El detalle de este objeto esta descrito en la operación ***consultarMiCPAV***
Datos para prueba	Identificación Código tipo:1 Descripción tipo: CC Numero:79808442 DatosValidacion Usuario:cvuws Password: cVu8gOoZ9oOzIWS

OPERACIONES	
Nombre	registrarAporteAplicacion
Descripción	Servicio encargado de registrar un aporte realizado con la forma de pago Aplicación.
Parámetros	Tipo de Dato: String Nombre: strPoliza Descripción: Objeto que contiene el identificador del contrato al cual se le va a realizar el aporte. Tipo de Dato: float Nombre: valorPagado Descripción: Valor del aporte. Tipo de Dato: String Nombre: identificadorTransaccion Descripción: Identificador de la transacción. Tipo de Dato: DatosValidacionWS Nombre: dtValidacionWS Descripción: Objeto que tiene la información que permite o no poder utilizar el servicio de registrarAporteAplicacion.
Retorna	Boolean que indica si se pudo registrar el aporte con tarjeta de crédito.
Datos para prueba	strPoliza: 01Q191633 valorPagado: 158925 identificadorTransaccion: XXABC123 DatosValidacion Usuario:cvuws Password: cVu8gOoZ9oOzIWS

Servicio: VENTAS

URL: http://190.143.78.137:8484/axis2/services/Ventas?wsdl

OPERACIONES	
Nombre	registrarVentaCPAV
Descripción	Servicio encargado de grabar la información de la venta de un CPAV y crear el cliente en caso de ser el primer contrato.
Parámetros	*Tipo de Dato*: DatosVentaWS *Nombre*: dtVentaWS *Descripción*: Objeto que contiene la información de la venta; código del plan adquirido, expectativa de viaje, modalidad de recaudo y las formas de pago. *Tipo de Dato*: DatosClienteWS *Nombre*: dtClienteWS *Descripción*: Objeto que contiene la información de cliente que adquiere el CPAV. *Tipo de Dato*: DatosValidacionWS *Nombre*: dtValidacionWS *Descripción*: Objeto que tiene la información que permite o no poder utilizar el servicio de registrarVentaCPAV.
Retorna	Objeto de tipo DatosResultadoVentaWS que contiene el tipo de contrato, el numero de contrato y el id del producto creado.
Datos para prueba	DatosValidacion Usuario:cvuws Password: cVu8gOoZ9oOzIWS DatosVenta • Código de plan: Tener en cuenta la codificación que retorna la operación consultarPlanes del servicio UTIL. • Modalidad de recaudo: Tener en cuenta la codificación que retorna la operación consultarModalidadRecaudo del servicio UTIL. • Formas de pago: código 8 para tarjeta crédito y código 28 para tarjeta debito. DatosCliente Tener en cuenta para los siguientes campos: • Ciudad residencial y ciudad comercial: operación consultarCiudadesResCom del servicio UTIL. • Estado civil: operación consultarEstadosCiviles del servicio UTIL.

Servicio: UTIL

URL: http://190.143.78.137:8484/axis2/services/Util?wsdl

OPERACIONES	
Nombre	consultarPlanes
Descripción	Servicio encargado de consultar la información de los planes existentes (código, descripción, valores, tipo de producto: 18, 24,36).
Parámetros	Tipo de Dato: DatosValidacionWS Nombre: dtValidacionWS Descripción: Objeto que tiene la información que permite o no poder utilizar el servicio de consultarPlanes.
Retorna	Colección de objetos de tipo DatosPlanWS, que contiene toda la información relacionada con cada uno de los planes vigentes.
Datos para prueba	DatosValidacion Usuario:cvuws Password: cVu8gOoZ9oOzIWS

OPERACIONES	
Nombre	consultarCiudadesResCom
Descripción	Servicio encargado de consultar la información de las ciudades a utilizar en el momento del registro de una venta, para identificar la ciudad de residencia y la ciudad comercial (donde labora) del cliente.
Parámetros	Tipo de Dato: DatosValidacionWS Nombre: dtValidacionWS Descripción: Objeto que tiene la información que permite o no poder utilizar el servicio de consultarCiudadesResCom.
Retorna	Colección de objetos de tipo DatosCiudadWS, que contiene toda la información relacionada las ciudades Colombianas.
Datos para prueba	DatosValidacion Usuario:cvuws Password: cVu8gOoZ9oOzIWS

OPERACIONES	
Nombre	consultarModalidadRecaudo
Descripción	Servicio encargado de consultar la información de las modalidades de recaudo que actualmente están disponibles para recaudo no presencial.
Parámetros	Tipo de Dato: DatosValidacionWS Nombre: dtValidacionWS Descripción: Objeto que tiene la información que permite o no poder utilizar el servicio de consultarModalidadRecaudo.
Retorna	Colección de objetos de tipo DatosGeneralWS, que contiene toda la información relacionada las modalidades de recaudo no presencial.
Datos para prueba	DatosValidacion Usuario:cvuws Password: cVu8gOoZ9oOzIWS

OPERACIONES	
Nombre	consultarTiposParentesco
Descripción	Servicio encargado de consultar la información de los tipos de parentesco, los cuales se utilizan al registrar una venta para identificar cual es el parentesco del segundo suscriptor con respecto al primero.
Parámetros	Tipo de Dato: DatosValidacionWS Nombre: dtValidacionWS Descripción: Objeto que tiene la información que permite o no poder utilizar el servicio de consultarTiposParentesco.
Retorna	Colección de objetos de tipo DatosGeneralWS, que contiene toda la información relacionada con los tipos de parentesco, Ej.: Hermano/a, Padre, Madre.
Datos para prueba	DatosValidacion Usuario:cvuws Password: cVu8gOoZ9oOzIWS

OPERACION	
Nombre	consultarSexos
Descripción	Servicio encargado de consultar el listado de sexos registrados en la base de datos.
Parámetros	Tipo de Dato: DatosValidacionWS Nombre: dtValidacionWS Descripción: Objeto que tiene la información que permite o no poder utilizar el servicio de consultarSexos.
Retorna	Colección de objetos de tipo DatosGeneralWS, que contiene el listado de sexos.
Datos para prueba	DatosValidacion Usuario:cvuws Password: cVu8gOoZ9oOzIWS

OPERACION	
Nombre	consultarCiudades
Descripción	Servicio encargado de consultar el listado de ciudades registradas en la base de datos
Parámetros	Tipo de Dato: DatosValidacionWS Nombre: dtValidacionWS Descripción: Objeto que tiene la información que permite o no poder utilizar el servicio de consultarCiudades.
Retorna	Colección de objetos de tipo DatosCiudadWS, que contiene el listado de ciudades.
Datos para prueba	DatosValidacion Usuario:cvuws Password: cVu8gOoZ9oOzIWS

OPERACIÓN	
Nombre	consultarEstadosCiviles
Descripción	Servicio encargado de consultar el listado de estados civiles registrados en la base de datos.
Parámetros	Tipo de Dato: DatosValidacionWS Nombre: dtValidacionWS Descripción: Objeto que tiene la información que permite o no poder utilizar el servicio de consultarEstadosCiviles.
Retorna	Colección de objetos de tipo DatosGeneralWS, que contiene el listado de estados civiles.
Datos para prueba	DatosValidacion Usuario:cvuws Password: cVu8gOoZ9oOzIWS

OPERACIÓN	
Nombre	consultarTiposIdentificacion
Descripción	Servicio encargado de consultar el listado de tipos de identificación registrados en la base de datos.
Parámetros	Tipo de Dato: DatosValidacionWS Nombre: dtValidacionWS Descripción: Objeto que tiene la información que permite o no poder utilizar el servicio de consultarTiposIdentificacion.
Retorna	Colección de objetos de tipo DatosGeneralWS, que contiene el listado de tipos de identificación.
Datos para prueba	DatosValidacion Usuario:cvuws Password: cVu8gOoZ9oOzIWS

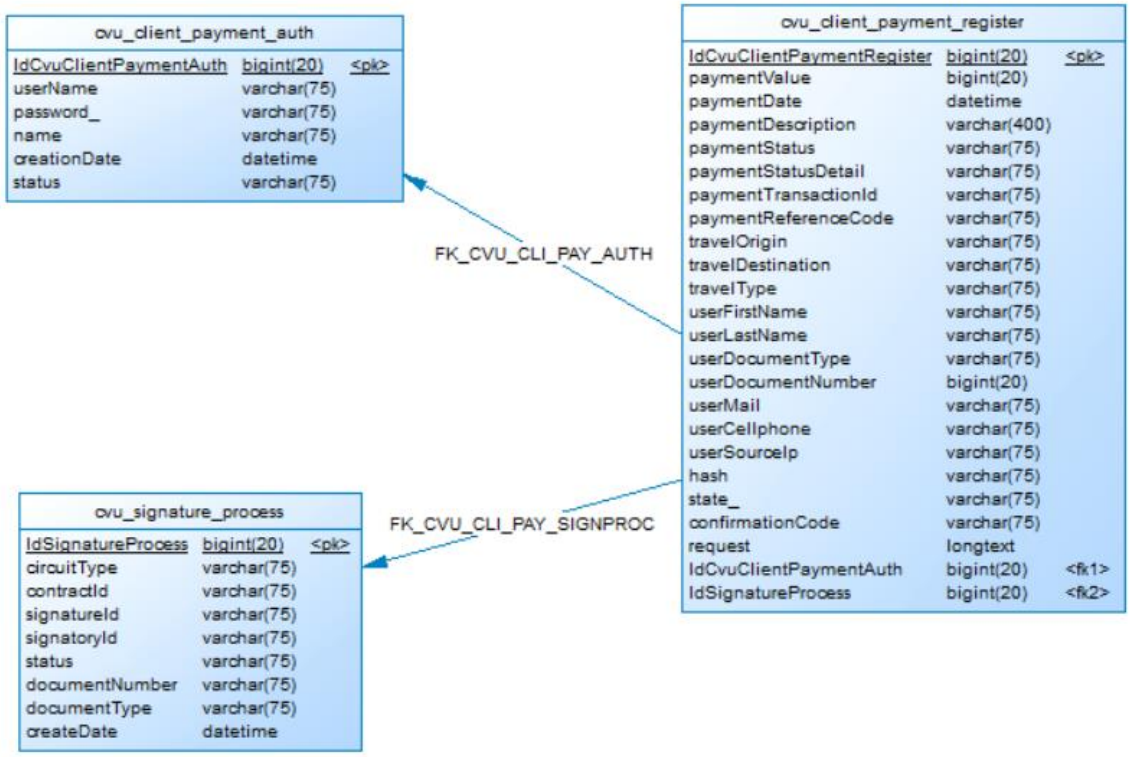
OPERACIÓN	
Nombre	consultarMaximaCantidadDeAportesEnVenta
Descripción	Servicio encargado de consultar el parámetro que indica cual esla cantidad máxima de aportes que se puede hacer en el momento de adquirir un nuevo contrato.
Parámetros	Tipo de Dato: DatosValidacionWS Nombre: dtValidacionWS Descripción: Objeto que tiene la información que permite o no poder utilizar el servicio de consultarMaximaCantidadDeAportesEnVenta.
Retorna	Colección de objetos de tipo DatosGeneralWS, que contiene el listado de tipos de identificación.
Datos para prueba	DatosValidacion Usuario:cvuws Password: cVu8gOoZ9oOzIWS

OPERACIÓN	
Nombre	consultarMaximaCantidadDeAportesEnRecaudo
Descripción	Servicio encargado de consultar el parámetro que indica cual es la cantidad máxima de aportes que se puede hacer en el momento de hacer un aporte con la forma de pago Aplicación.
Parámetros	Tipo de Dato: DatosValidacionWS Nombre: dtValidacionWS Descripción: Objeto que tiene la información que permite o no poder utilizar el servicio de consultarMaximaCantidadDeAportesEnRecaudo.
Retorna	Colección de objetos de tipo DatosGeneralWS, que contiene el listado de tipos de identificación.
Datos para prueba	DatosValidacion Usuario:cvuws Password: cVu8gOoZ9oOzIWS

BOTON DE PAGO CVU

ESTRUCTURA BD

Diagrama



Script

```
/*==============================================================*/
/* Table: cvu_client_payment_auth                               */
/*==============================================================*/
create table cvu_client_payment_auth
(
  IdCvuClientPaymentAuth bigint(20) not null,
  userName           varchar(75),
  password_          varchar(75),
  name               varchar(75),
  creationDate       datetime,
  status             varchar(75),
  primary key (IdCvuClientPaymentAuth)
);

/*==============================================================*/
```

```
/* Table: cvu_signature_process                    */
/*==============================================================*/
create table cvu_signature_process
(
  IdSignatureProcess   bigint(20) not null,
  circuitType          varchar(75),
  contractId           varchar(75),
  signatureId          varchar(75),
  signatoryId          varchar(75),
  status               varchar(75),
  documentNumber       varchar(75),
  documentType         varchar(75),
  createDate           datetime,
  primary key (IdSignatureProcess)
);

/*==============================================================*/
/* Table: cvu_client_payment_register                    */
/*==============================================================*/
create table cvu_client_payment_register
(
  IdCvuClientPaymentRegister bigint(20) not null,
  paymentValue         bigint(20) not null,
  paymentDate          datetime,
  paymentDescription   varchar(400),
  paymentStatus        varchar(75),
  paymentStatusDetail  varchar(75),
  paymentTransactionId varchar(75),
  paymentReferenceCode varchar(75),
  travelOrigin         varchar(75),
  travelDestination    varchar(75),
  travelType           varchar(75),
  userFirstName        varchar(75),
  userLastName         varchar(75),
  userDocumentType     varchar(75),
  userDocumentNumber   bigint(20),
  userMail             varchar(75),
  userCellphone        varchar(75),
  userSourceIp         varchar(75),
  hash                 varchar(75),
  state_               varchar(75),
  confirmationCode     varchar(75),
```

```
  request            longtext,
  IdCvuClientPaymentAuth bigint(20),
  IdSignatureProcess   bigint(20),
  primary key (IdCvuClientPaymentRegister)
);

alter table cvu_client_payment_register add constraint FK_CVU_CLI_PAY_AUTH foreign key
(IdCvuClientPaymentAuth)
    references cvu_client_payment_auth (IdCvuClientPaymentAuth) on delete restrict on update
restrict;

alter table cvu_client_payment_register add constraint FK_CVU_CLI_PAY_SIGNPROC foreign key
(IdSignatureProcess)
    references cvu_signature_process (IdSignatureProcess) on delete restrict on update restrict;
```