Actividad Extra: Despliegue de un Loosley Coupled Monolith con VSA

Este proyecto plantea la creación de un **Loosely Coupled Monolith** utilizando la **Vertical Slice Architecture (VSA)** para resolver las necesidades de digitalización de **Autoescuelas Eco**, una empresa real que busca gestionar su flota vehicular de manera más eficiente. La solución propuesta es un **SaaS** (Software as a Service) desplegado en la infraestructura cloud de AWS, lo que permite aprovechar la escalabilidad y flexibilidad de la nube.

Arquitectura y Diseño del Dominio

Para estructurar el monolito de manera adecuada, hemos aplicado el enfoque **Domain-Driven Design (DDD)**, lo que permite separar claramente los **Bounded Contexts**, **Subdominios**, **Agregados**, **Entidades**, y **Value Objects**. La solución incluye varias capas que se comunican de manera eficiente a través de un **EventBus** en memoria, lo que garantiza un bajo acoplamiento entre los distintos módulos. Este diseño permite que en el futuro los módulos puedan ser desplegados de manera independiente mediante servicios de mensajería como **SQS**, **RabbitMQ** o **Kafka**.

Además, el uso del **patrón repositorio** asegura la independencia de la base de datos, permitiendo que cada uno de los **Slices** (Autoescuelas, Profesores y Vehículos) pueda tener su propio almacenamiento independiente.

Como parte de la exposición de dicha aplicación se ha usado el framework SpringBoot. El sistema se ha expuesto mediante una API, que, si bien no cumple completamente el standard, se podría decir que es Full REST.

Componentes del Modelo

Bounded Context: Flotas Vehiculares

- ***Subdominios***:
    1. **Autoescuelas**:
        - ***Dominio***:
            - ***Agregado Autoescuela***: *Contiene la información principal de una autoescuela, como su* ***Nombre***, ***CIF (Id)***, *y listas de* ***Secciones***, ***Coches***, *y* ***Profesores***.
            - ***Entidades***:
                - ***Coche***: *Identificado por su* ***Matrícula (Id)***.
                - ***Profesor***: *Identificado por su* ***DNI (Id)***.
                - ***Sección***: *Identificada por un* ***UUID*** *y su* ***Localización***.
        - ***Aplicación***:
            - ***Casos de Uso***:
                - ***Comandos***: *Incluyen acciones como Dar de alta autoescuela(), Abrir sección(), y Cambiar datos fiscales().*
                - ***Queries***: *Consultas como Lista Autoescuelas().*
            - ***Puertos***:
                - ***IN***: *Repositorio de autoescuelas.*
                - ***OUT***: *Salida en formato JSON.*
        - ***Infraestructura***: *Un repositorio en memoria (In Memory Repo Autoescuela).*
    2. **Coches**:
        - ***Dominio***:
            - ***Agregado Coche***: *Contiene atributos como la* ***Autoescuela (Id)***, ***Matrícula (Id)***, ***Profesor (Id)***, ***Modelo***, ***Marca***, ***ITV***, *y* ***Combustible***.
        - ***Aplicación***:
            - ***Casos de Uso***:
                - ***Comandos***: *Comprar coche(), Asociar profesor().*

        - ***Queries**: Listar coches().*
    - ***Puertos**:*
        - ***IN**: Repositorio de coches.*
        - ***OUT**: Salida en formato JSON.*
- ***Infraestructura**: Repositorio en memoria (In Memory Repo Coche).*

3. **Profesores**:
    - ***Dominio**:*
        - ***Agregado Profesor**: Contiene información relacionada con la **Autoescuela (Id)**, **Matrículas de coches (list)**, **DNI (Id)**, **Nombre**, **Apellidos**, **Edad**, y **Sexo**.*
    - ***Aplicación**:*
        - ***Casos de Uso**:*
            - ***Comandos**: Contratar profesor(), Despedir profesor().*
            - ***Queries**: Listar profesores().*
        - ***Puertos**:*
            - ***IN**: Repositorio de profesores.*
            - ***OUT**: Salida en formato JSON.*
    - ***Infraestructura**: Repositorio en memoria (In Memory Repo Profesor).*

Shared kernel

- ***Dominio**:*
    - *Componentes compartidos como **entidad**, **aggregate root**, y **domain event**.*
- ***Aplicación**:*
    - *Interfaces como el **event bus**, **event handler**, **base repository**, **base command**, **base query**, y un **id generator**.*
- ***Infraestructura**:*
    - *Elementos pendientes de definir.*

Despliegue

Para favorecer el despliegue de esta aplicación se ha dockerizado. Hemos creado un Dockerfile que contiene lo siguiente:

```dockerfile
# Usar una imagen base de Maven con JDK 17
FROM maven:3.8.4-openjdk-17 AS build

# Establecer el directorio de trabajo dentro del contenedor
WORKDIR /app

# Copiar el archivo pom.xml y descargar las dependencias
COPY pom.xml .
RUN mvn dependency:go-offline -B

# Copiar el resto del proyecto al contenedor
COPY src ./src

# Compilar la aplicación con Maven
RUN mvn clean package -DskipTests

# Usar una imagen base de OpenJDK 17 para ejecutar la aplicación
FROM openjdk:17-jdk-slim

# Establecer el directorio de trabajo para la aplicación
WORKDIR /app

# Copiar el archivo JAR generado desde la fase anterior
COPY --from=build /app/target/*.jar app.jar

# Exponer el puerto 8080 para que la aplicación esté disponible
EXPOSE 8080

# Comando para ejecutar la aplicación Spring Boot
ENTRYPOINT ["java", "-jar", "app.jar"]
```

A continuación, hemos creado una instancia de EC2 configurada exactamente que las máquinas anteriores. Hemos establecido una conexión SSH con dicha máquina y hemos instalado git, en orden de clonar el proyecto, y Docker, en orden de desplegarlo.

```
[ec2-user@ip-172-31-16-124 FleetManager]$ sudo dnf install docker
Last metadata expiration check: 0:05:56 ago on Sun Sep 15 23:54:49 2024.
Dependencies resolved.
==========================================================================================================
 Package                    Architecture       Version                       Repository            Size
==========================================================================================================
Installing:
 docker                     x86_64             25.0.6-1.amzn2023.0.2         amazonlinux           44 M
Installing dependencies:
 containerd                 x86_64             1.7.20-1.amzn2023.0.1         amazonlinux           35 M
 iptables-libs              x86_64             1.8.8-3.amzn2023.0.2          amazonlinux          401 k
 iptables-nft               x86_64             1.8.8-3.amzn2023.0.2          amazonlinux          183 k
 libcgroup                  x86_64             3.0-1.amzn2023.0.1            amazonlinux           75 k
 libnetfilter_conntrack     x86_64             1.0.8-2.amzn2023.0.2          amazonlinux           58 k
 libnfnetlink               x86_64             1.0.1-19.amzn2023.0.2         amazonlinux           30 k
 libnftnl                   x86_64             1.2.2-2.amzn2023.0.2          amazonlinux           84 k
 pigz                       x86_64             2.5-1.amzn2023.0.3            amazonlinux           83 k
 runc                       x86_64             1.1.13-1.amzn2023.0.1         amazonlinux          3.2 M

Transaction Summary
==========================================================================================================
Install  10 Packages

Total download size: 84 M
Installed size: 317 M
```

```
[ec2-user@ip-172-31-16-124 ~]$ sudo dnf install git
Last metadata expiration check: 0:00:30 ago on Sun Sep 15 23:54:49 2024.
Dependencies resolved.
================================================================================
 Package            Architecture   Version                      Repository    Size
================================================================================
Installing:
 git                x86_64         2.40.1-1.amzn2023.0.3        amazonlinux   54 k
Installing dependencies:
 git-core           x86_64         2.40.1-1.amzn2023.0.3        amazonlinux  4.3 M
 git-core-doc       noarch         2.40.1-1.amzn2023.0.3        amazonlinux  2.6 M
 perl-Error         noarch         1:0.17029-5.amzn2023.0.2     amazonlinux   41 k
 perl-File-Find     noarch         1.37-477.amzn2023.0.6        amazonlinux   26 k
 perl-Git           noarch         2.40.1-1.amzn2023.0.3        amazonlinux   42 k
 perl-TermReadKey   x86_64         2.38-9.amzn2023.0.2          amazonlinux   36 k
 perl-lib           x86_64         0.65-477.amzn2023.0.6        amazonlinux   15 k

Transaction Summary
================================================================================
Install  8 Packages
```

Finalmente hemos compilado la imagen de Docker utilizando el comando:

```
docker build -t springboot-app .
docker run -p 8080:8080 springboot-app
```

Para garantizar el acceso a la api hay que habilitar el puerto 8080 de nuestro servidor igual que hicimos anteriormente.

Filter rules

Security group rule ID	Port range	Protocol	Source	Security groups	De
sgr-0eeac5bb53df2514f	8080	TCP	0.0.0.0/0	launch-wizard-2	–
sgr-043720c3dbcfab23a	22	TCP	0.0.0.0/0	launch-wizard-2	–

Finalmente observaremos como nuestro servidor esta ejecutándose sin ningún error.

No es seguro 54.226.252.141:8080/v1/driving_schools/

Dar formato al texto


```
{
  "value": [],
  "success": true,
  "errorMessage": null
}
```

Endpoints expuestos por el servicio web

DrivingSchoolsController (Gestión de las Autoescuelas)

1. **GET /v1/driving_schools/**
    - ***Descripción**: Obtiene una lista de todas las autoescuelas registradas en el sistema.*
    - ***Respuesta**: Devuelve un listado con los detalles de cada autoescuela.*
2. **POST /v1/driving_schools/**
    - ***Descripción**: Añade una nueva autoescuela al sistema.*
    - ***Cuerpo (RequestBody)**: Un objeto AddDrivingSchoolDTO que contiene los detalles de la nueva autoescuela.*
    - ***Respuesta**: Devuelve los detalles de la autoescuela recién creada.*
3. **POST /v1/driving_schools/{id}/sections**
    - ***Descripción**: Abre una nueva sección dentro de una autoescuela existente.*
    - ***Cuerpo (RequestBody)**: Un objeto OpenSectionDTO con la información de la nueva sección.*
    - ***Respuesta**: Devuelve los detalles de la sección creada.*

TeachersController (Gestión de Profesores)

1. **GET /v1/teachers/**
    - ***Descripción**: Obtiene una lista de todos los profesores contratados.*
    - ***Respuesta**: Devuelve un listado con los detalles de cada profesor.*
2. **POST /v1/teachers/**
    - ***Descripción**: Contrata a un nuevo profesor para una autoescuela.*
    - ***Cuerpo (RequestBody)**: Un objeto HireTeacherDTO con la información del nuevo profesor.*
    - ***Respuesta**: Devuelve los detalles del profesor contratado.*
3. **POST /v1/teachers/fire**
    - ***Descripción**: Despide a un profesor.*
    - ***Cuerpo (RequestBody)**: Un objeto FireTeacherDTO con los detalles del profesor a despedir.*
    - ***Respuesta**: Devuelve una confirmación de la operación.*

VehiclesController (Gestión de Vehículos)

1. **GET /v1/vehicles/**
    - ***Descripción**: Obtiene una lista de todos los vehículos disponibles en la flota de las autoescuelas.*
    - ***Respuesta**: Devuelve un listado con los detalles de cada vehículo.*
2. **POST /v1/vehicles/associate/**
    - ***Descripción**: Asocia un vehículo a un profesor.*
    - ***Cuerpo (RequestBody)**: Un objeto AssociateVehicleDTO con los detalles de la asociación entre el vehículo y el profesor.*
    - ***Respuesta**: Devuelve una confirmación de la operación.*
3. **POST /v1/vehicles/**
    - ***Descripción**: Compra un nuevo vehículo para la flota.*
    - ***Cuerpo (RequestBody)**: Un objeto BuyVehicleDTO con los detalles del vehículo a adquirir.*
    - ***Respuesta**: Devuelve los detalles del vehículo recién adquirido.*

Posibles mejoras

*En el futuro, se podrían implementar mejoras como una cadena CI/CD completa con **Jenkins** para automatizar el build, test y despliegue de la aplicación, y utilizar **Terraform** para gestionar la infraestructura como código, lo que facilita la escalabilidad y replicabilidad. Además, se puede integrar monitoreo con **Prometheus** y logging centralizado con **ELK Stack** o **CloudWatch** para mejorar la observabilidad, y fortalecer la seguridad automatizando escaneos de vulnerabilidades y optimizando reglas de acceso en AWS mediante **Security Groups** y **Network ACLs**.*