# Universidad La Salle
# CodeGuardGT

---

# Documento SAD
# Versión 2.2

# Documento de Arquitectura de Software

## 1. Introducción

### 1.1 Propósito

Este documento proporciona una visión general arquitectónica del sistema **CodeGuardGT**, diseñado para la detección de plagio en código SQL. El propósito es presentar una serie de vistas arquitectónicas que abordan diferentes aspectos del sistema. Esto permitirá capturar y establecer las decisiones arquitectónicas significativas tomadas durante el desarrollo, garantizando que el sistema cumpla con sus requisitos funcionales y no funcionales.

### 1.2 Alcance

El alcance de este documento incluye:

- La presentación de la arquitectura del sistema **CodeGuardGT**.
- La descripción de las decisiones arquitectónicas clave.
- La explicación de las vistas arquitectónicas que ilustran cómo el sistema está estructurado y cómo interactúan sus componentes.

### 1.3 Definiciones, Acrónimos, y Abreviaturas

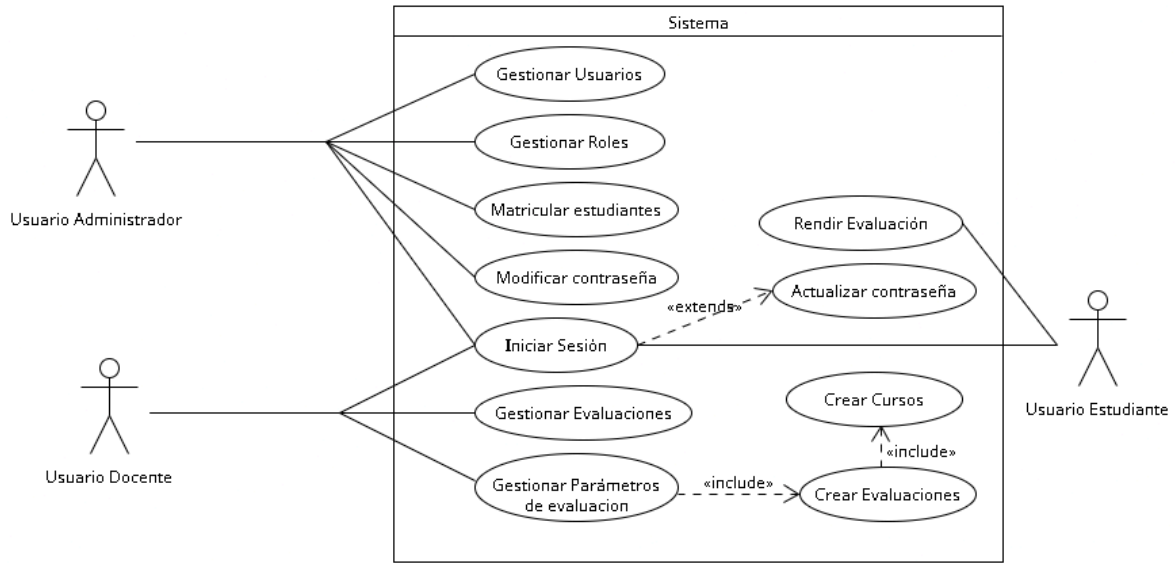
- **CodeGuardGT**: Sistema de detección de plagio en código SQL.
- **SQL**: Structured Query Language (Lenguaje de Consulta Estructurado).
- **SQLFiddle**: Plataforma online para ejecutar y comparar código SQL.
- **GDPR**: General Data Protection Regulation (Reglamento General de Protección de Datos).
- **SSL**: Secure Sockets Layer (Capa de Conexión Segura).

### 1.4 Visión general

El documento presenta la arquitectura del sistema a través de diversas vistas que son descritas en las siguientes secciones. Estas vistas proporcionan una visión integral de cómo está estructurado **CodeGuardGT** y cómo se espera que funcione en el entorno de producción.

# 2. Vista de escenarios

## 2.1 Diagrama

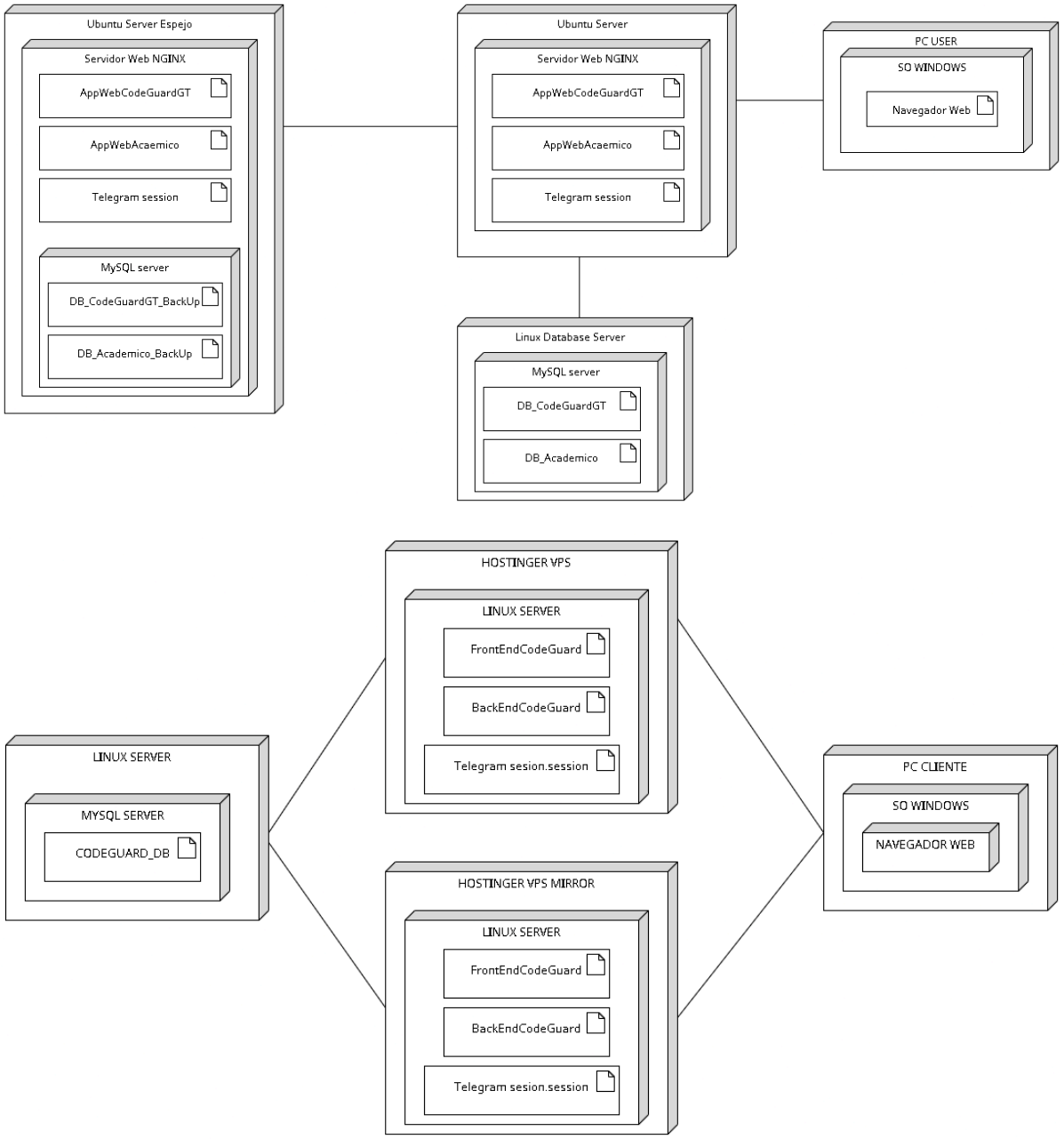



## 2.2 Descripción de la Vista:

- **Actores**: Los actores son representados por figuras de palitos. Estos pueden ser usuarios o sistemas externos que interactúan con el sistema. En tu diagrama, veo varios actores, como “Usuario”, “Administrador”, y “Sistema Externo”.
- **Casos de Uso**: Los casos de uso están representados por óvalos y describen las funcionalidades que el sistema proporciona a los actores. Algunos ejemplos en tu diagrama incluyen “Iniciar Sesión”, “Registrar Usuario”, y “Generar Reporte”.
- **Relaciones**: Las líneas que conectan los actores con los casos de uso indican quién puede realizar qué acciones. Por ejemplo, el “Usuario” está conectado a “Iniciar Sesión” y “Registrar Usuario”, lo que significa que el usuario puede realizar estas acciones.
- **Sistema**: El rectángulo que engloba los casos de uso representa el sistema en sí. Todo lo que está dentro del rectángulo es parte del sistema que se está modelando.
- **Extensiones y Inclusiones**: A veces, los casos de uso pueden tener relaciones especiales como “extiende” o “incluye”. Estas relaciones se utilizan para mostrar que un caso de uso puede ser parte de otro o que puede extender su funcionalidad. En tu diagrama, veo algunas de estas relaciones, como “Iniciar Sesión” que incluye “Validar Usuario”.

# 3. Vista física

## 3.1 Diagrama




## 3.2 Descripción de la Vista

En esta vista física se observa una infraestructura de red compuesta por varios servidores y un usuario conectado desde una PC con sistema operativo Windows. El usuario accede a las aplicaciones web mediante un navegador, el cual se comunica con un servidor principal basado en Ubuntu.

El servidor Ubuntu tiene configurado un servidor web NGINX que hospeda varias aplicaciones: AppWebCodeGuardGT, AppWebAcademico y una sesión activa de Telegram. Estas aplicaciones están vinculadas a una base de datos MySQL, la cual maneja respaldos de dos bases de datos principales: DB_CodeGuardGT_BackUp y DB_Academico_BackUp.

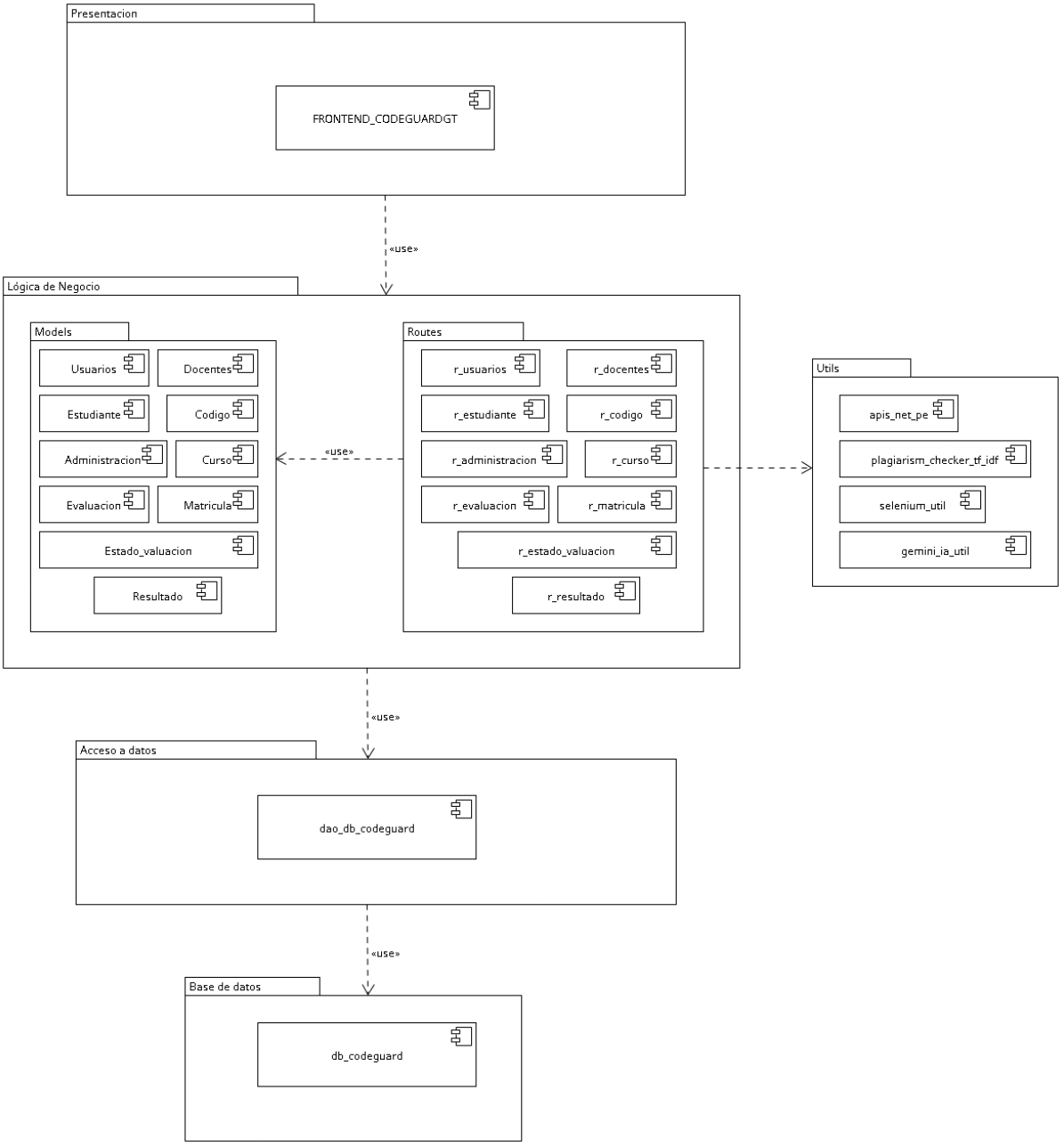
Junto al servidor principal de Ubuntu, existe un servidor espejo, también basado en Ubuntu, que replica las mismas aplicaciones web y el servidor de MySQL, asegurando la

disponibilidad de las bases de datos de respaldo DB_CodeGuardGT_BackUp y DB_Academico_BackUp.

Por otro lado, un servidor dedicado exclusivamente a las bases de datos en Linux maneja las versiones primarias de DB_CodeGuardGT y DB_Academico. Todas las aplicaciones web en los servidores Ubuntu, tanto el principal como el espejo, interactúan con este servidor de base de datos, garantizando el acceso a la información de forma centralizada.

# 4. Vista de desarrollo

## 4.1 Diagrama




## 4.2 Descripción de la Vista

Esta vista de desarrollo representa la arquitectura de una CODEGUARDGT. La estructura se divide en varias capas, desde la interfaz de presentación hasta la base de datos, organizando cada sección en módulos específicos:

- **Presentación (Frontend)**

    - Contiene el módulo FRONTEND_CODEGUARD que representa la interfaz de usuario de la aplicación. Aquí es donde los usuarios interactúan visualmente con la aplicación a través de navegadores u otros clientes.
- **Lógica de Negocio**
    - **Models**: Incluye varios modelos que representan las entidades clave de la aplicación:
    - Usuarios y Docentes: Representan los perfiles de usuario.
    - Estudiante, Administración, Evaluación: Modelos que probablemente gestionan el proceso de evaluación.
    - Código: Allmacena los códigos fuente a evaluar.
    - Curso y Matrícula: Representan los cursos y el registro de estudiantes en ellos.
    - Estado_valuacion y Resultado: Para el estado y los resultados de las evaluaciones.
    - **Routes**: Define las rutas de la aplicación, correspondientes a cada modelo:
    - r_usuarios, r_docentes, r_estudiante, etc., son rutas para manejar las operaciones CRUD (crear, leer, actualiza**Utils (Utilidades)**r, eliminar) en cada entidad.
    - Estas rutas se utilizan para organizar y controlar el flujo de datos entre el frontend y los modelos en el backend.
- **Utilitarios**
    - Incluye varios módulos de utilidad que añaden funcionalidades específicas:
    - apis_net_pe: Indica el API que se utiliza para verificar los DNI peruanos.
    - plagiarism_checker_tfidf: Un módulo para la detección de plagio usando TF-IDF.
    - selenium_util: Incluye funciones para automatizar navegadores mediante Selenium.
    - gemini_ia_util: Es una inteligencia artificial para funciones avanzadas.
    - classroom: Es un servicio de google para tareas
- **Acceso a Datos**
    - dao_db_codeguard: Este módulo se encarga del acceso y manipulación de datos en la base de datos, sirviendo como una capa entre la lógica de negocio y la base de datos real.
- **Base de Datos**
    - db_codeguard: Es el componente que representa la base de datos física donde se almacenan los datos de la aplicación, interactuando a través del módulo dao_db_codeguard para el almacenamiento y recuperación de la información.