ASIGNATURA

Programación 3

TEMA

Arquitectura del proyecto

PROFESOR

Willis Ezequiel Polanco

PARTICIPANTE

Pedro Alexander Medina (2019-7970)

Abiel Nicolas Sandoval Martínez (2019-7806)

Joel Federico Santana Reyes (2019-8786)

ENTREGA

15/06/2021

Arquitectura del Proyecto

Descripción del Proyecto

Objetivos

El objetivo general del proyecto es desarrollar un sistema que provea una plataforma para que las personas puedan encontrar o publicar un empleo.

Los objetivos de manera general pudieran listarse como:

- Desarrollar un sistema tecnológico y de vanguardia para la empleomanía.
- Ayudar a las comunidad a poder encontrar un empleo mediante una plataforma. ·
- Disminuir el costo de transporte o de impresión de los documentos(clase media y media baja).
- Proveer a los empleadores las informaciones de los postulantes al puesto (CV).

Alcance

El proyecto tiene como alcance desarrollar:

- Una **App Web** para desempleados y empleadores que permita agregar puestos de trabajos y buscar un trabajo para postularse.

- Un **Servicio Web / API** que conecte la aplicación con la base de datos del sistema.

Perspectiva del Producto

La comunidad de desempleados en la República Dominicana y Latinoamérica es bastante elevada en estos tiempos debido a la pandemia por la cual estamos pasando(Covid-19).

La crisis económica asociada a la emergencia sanitaria del covid-19 también ha “infectado” las actividades económicas en toda la región de Latinoamérica, empujando el cierre temporal y, en algunos casos definitivo, de miles de empresas.

En donde vimos la necesidad de crear una aplicación para que los desempleados puedan postularse a puestos de trabajo sin la necesidad de ir de manera presencial a la empresa. Y una vía para que los empleadores puedan publicar vacantes.

Características del Usuario

El sitio web tiene tres tipos de usuarios:

• **Administrador:** Propietario del Sitio Web.

• **User:** Visita la página web para buscar un puesto de trabajo y se postula para uno.

• **Poster:** Visita la página web para enviar/ofrecer un puesto de trabajo.

Debajo, planteamos las tablas de los usuarios como actores de casos de uso.

TABLA 2 CONJUNTO DE FICHAS DE ACTORES

Actor Specification	UCA-1
Actor Name: Administrador	**Abstract:** No
Descripción: Es el propietario del Sitio Web.	

Actor Specification	UCA-2
Actor Name: User	**Abstract:** No
Descripción: Es el visita la página web para buscar un puesto de trabajo y se postula para uno.	

Actor Specification	UCA-2
Actor Name: Poster	**Abstract:** No
Descripción: Visita la página web para enviar/ofrecer un puesto de trabajo	

Asunciones y Dependencias

- El usuario necesita de conexión a internet para poder visualizar y postularse a un puesto de trabajo.
- El poster necesita de conexión a internet para poder enviar/ofrecer un puesto de trabajo.

Requerimientos Específicos

Interfaces Externas

Funcionalidades

.

Aplicación Web:

- La aplicación debe permitir que el administrador crear una cuenta de usuario.
- La aplicación debe ser capaz de mostrar las vacantes a los usuarios.
- La aplicación debe permitir al administrador decidir las vacantes que serán enviadas a los usuarios.

Servicio Web / API:

El API debe ser el puente directo entre la base de datos y cualquier agente externo a ella. El API debe manejar las peticiones y respuestas de las aplicaciones externas, proveyéndoles la data correspondiente a su petición.

Atributos de Calidad

Aplicación Web

Seguridad

Un usuario no autorizado intenta acceder 4 veces seguidas a la aplicación, fallando la contraseña o usuario en cada intento; esta le niega el acceso, envía un mensaje al correo del usuario real indicando que hubo un intento de acceso no autorizado.

Disponibilidad

Un usuario entra a la aplicación web, la aplicación tiene vacantes nuevas, le notifica a cada usuario cada vez que se publica una vacante nueva.

Tácticas

En la siguiente tabla presentamos las medidas que tomamos para asegurar los atributos de calidad.

Atributo de Calidad	¿Cómo lograrlo?	Táctica/Patrón
Disponibilidad	Evaluar si hay una respuesta del API y verificar si la respuesta vuelve en el formato esperado.	Exceptions
Seguridad	Permitiendo la autenticación de usuarios podemos otorgar seguridad a la aplicación.	Authenticate User
Modificabilidad	· Manteniendo una semántica similar a través del sistema se puede mantener coherencia entre las dependencias de los módulos. · Al encapsular los módulos por capas (que comparten responsabilidades similares) se facilita el proceso de modificación.	Semantic Coherence Encapsulamiento De Capas

TABLA 3 TÁCTICAS DE CALIDAD

Documentación de Arquitectura

Vista Funcional

En esta sección veremos un conjunto de diagramas y tablas que representan la vista funcional del sistema.

Diagrama de Casos de Uso - Usuario

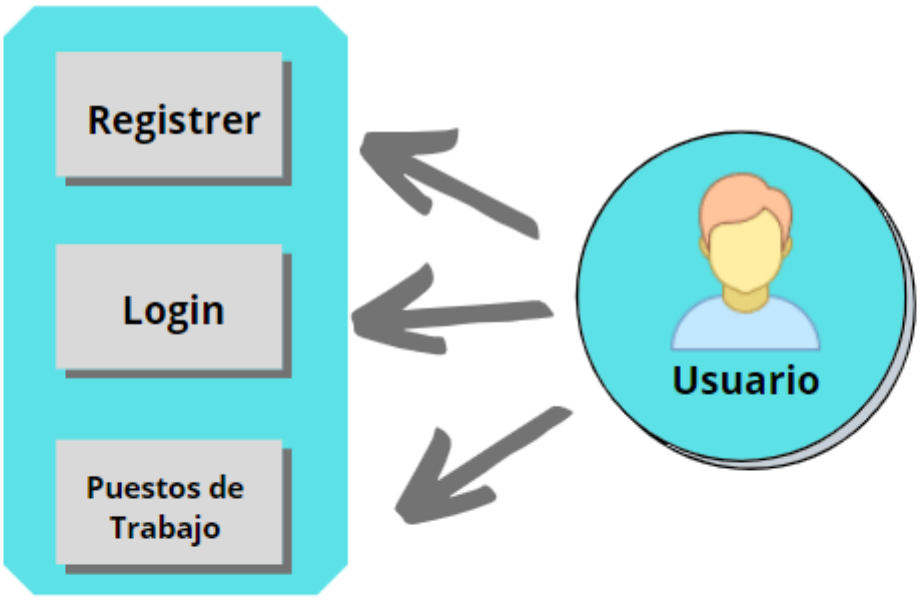

FIGURA 3 DIAGRAMA DE CASO DE USOS - Usuario

Diagrama de Casos de Uso - Poster

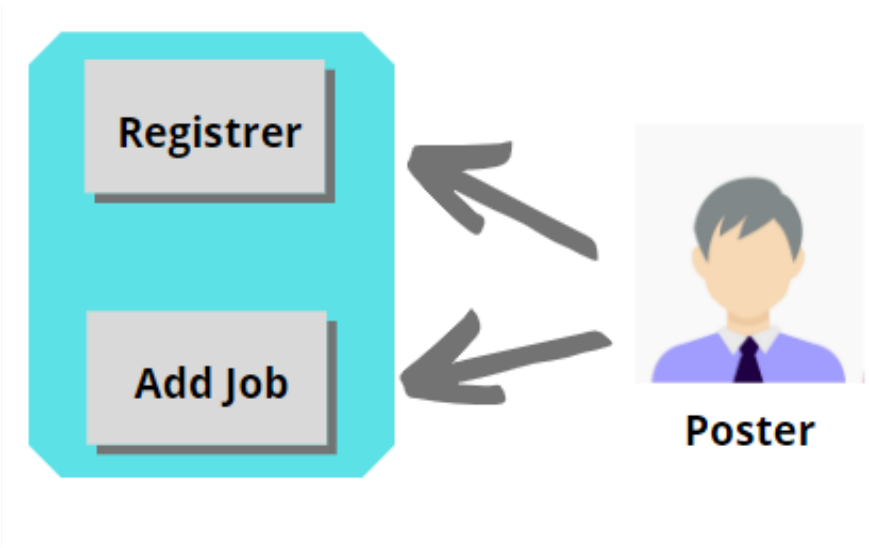

FIGURA 4 DIAGRAMA CASOS DE USO - Poster

Diagrama de Casos de Uso - Administrador

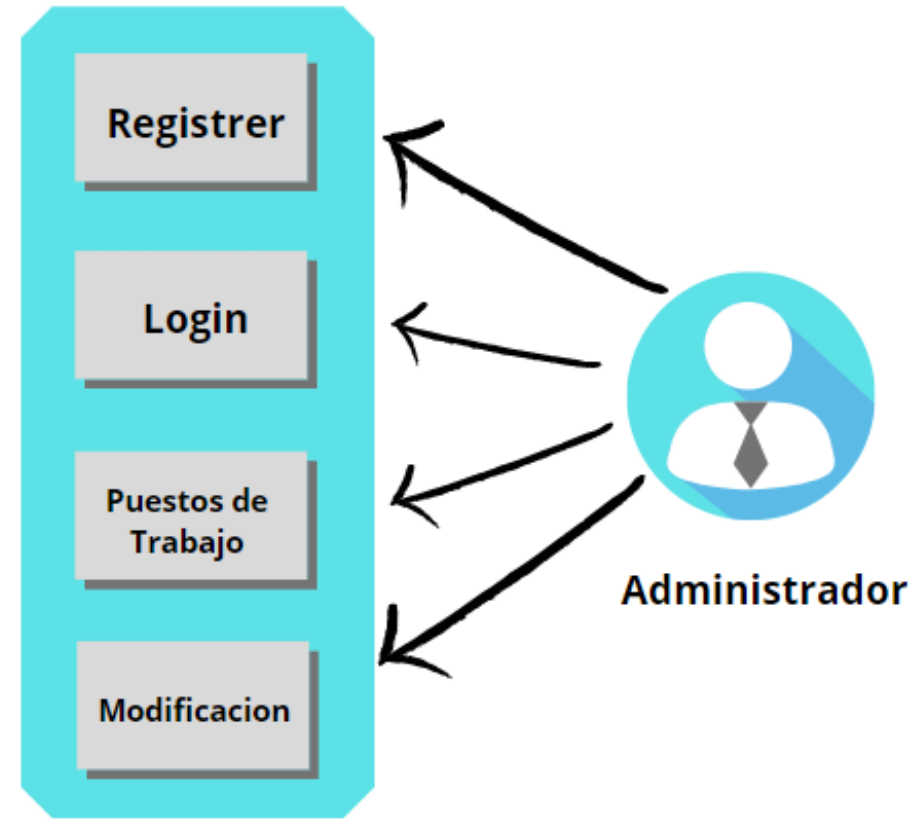

FIGURA 5 DIAGRAMA CASOS DE USO - Administrador

Fichas de Casos de Uso

En esta sección listamos en tablas los casos de uso del sistema completo con su descripción.

TABLA 4 CONJUNTO DE FICHAS DE CASOS DE USO

Use Case Name	Registrarse
Use Case ID	c-1
Primary Actor	Administrador
Secondary Actor	User
Precondición	N/A
Descripción	Ingresar los datos personales en la app web para crear una cuenta de usuario.

Use Case Name	Login
Use Case ID	c-2
Primary Actor	Administrador
Secondary Actor	User
Precondición	· **c-1:** Estar registrado en el sistema.
Descripción	El usuario ingresa sus credenciales de acceso para acceder al sistema.

Use Case Name	Ver puestos de trabajo
Use Case ID	c-3
Primary Actor	Administrador
Secondary Actor	User
Precondición	· **c-2:** Haber entrado al sistema.
Descripción	El usuario visualiza las vacantes disponibles.

Use CaseName	Agregar vacantes
Use Case ID	c-4
Primary Actor	Poster
Secondary Actor	N/A
Precondición	**c-2:** Haber entrado al sistema.
Descripción	El poster entra a la pagina para enviar o ofrecer puestos de trabajos.

Vista Estática

Decomposition Style – App Web

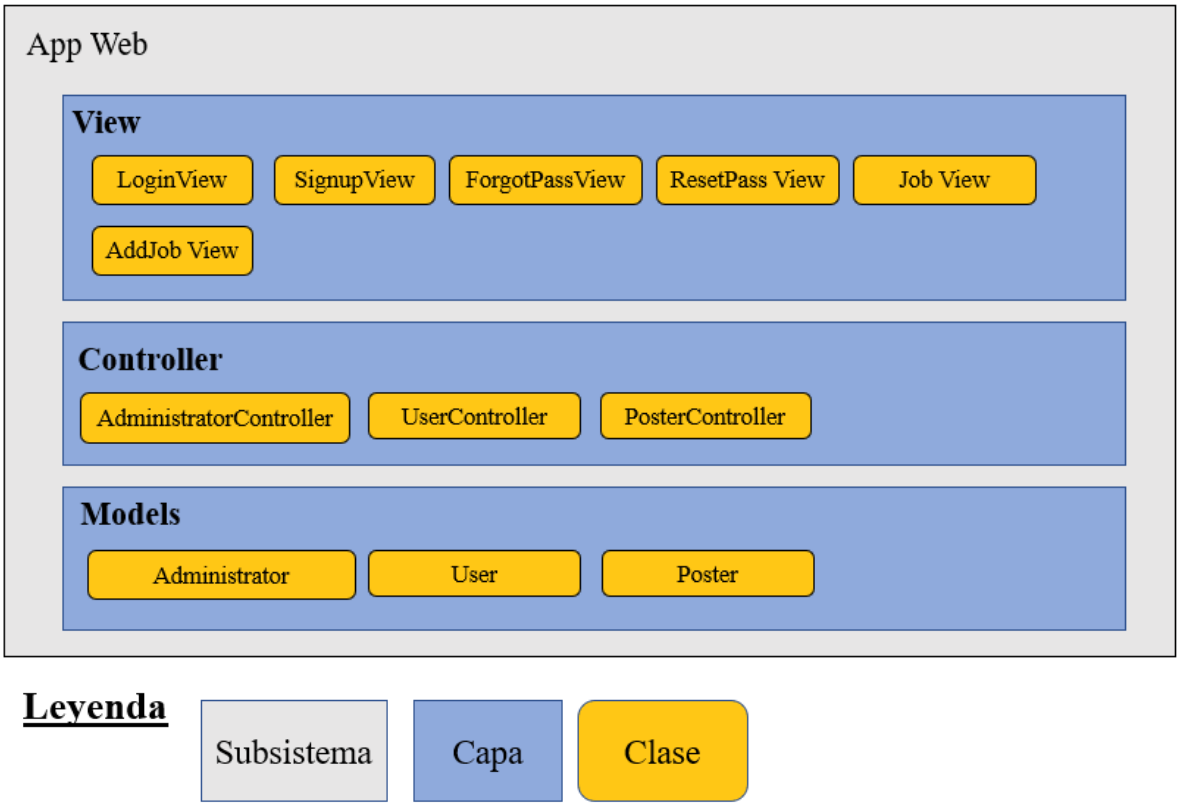

FIGURA 6 DECOMPOSITION STYLE - WEB APP

Element Catalog:

View: Contiene las vistas de la aplicación web.

- **LoginView**: Representa la vista para iniciar sesión.
- **SignupView**: Representa la vista para registro de nuevo usuario.
- **ForgotPassView**: Representa la vista de reinicio de contraseña de usuario para inserción de correo electrónico del usuario.
- **ResetPassView**: Representa la vista de reinicio de contraseña de usuario para la inserción de nueva contraseña de usuario.
- **JobView**: Representa la vista para las vacantes disponibles.
- **AddJob View**: Representa la vista para establecer las vacantes.

Controller: Contiene los controladores que se encargan de la interacción entre la capaView y la capa Model.

- **AdministratorController**: Control que administra las operaciones del administrador..
- **UserController**: Control que administra las operaciones de usuario.
- **PosterController**: Control que administra las operaciones del poster.

Models: Contiene los modelos que estructuran los objetos que se manejan en la aplicación y los atributos de cada uno.

- **Administrator:** Modelo que tiene los atributos que identifican al Administrador como parte del sistema.
- **User:** Modelo que tiene los atributos que identifican al usuario como parte del sistema.
- **Poster:** Modelo que tiene los atributos que identifican al póster como parte del sistema.

Decomposition Style - Web API

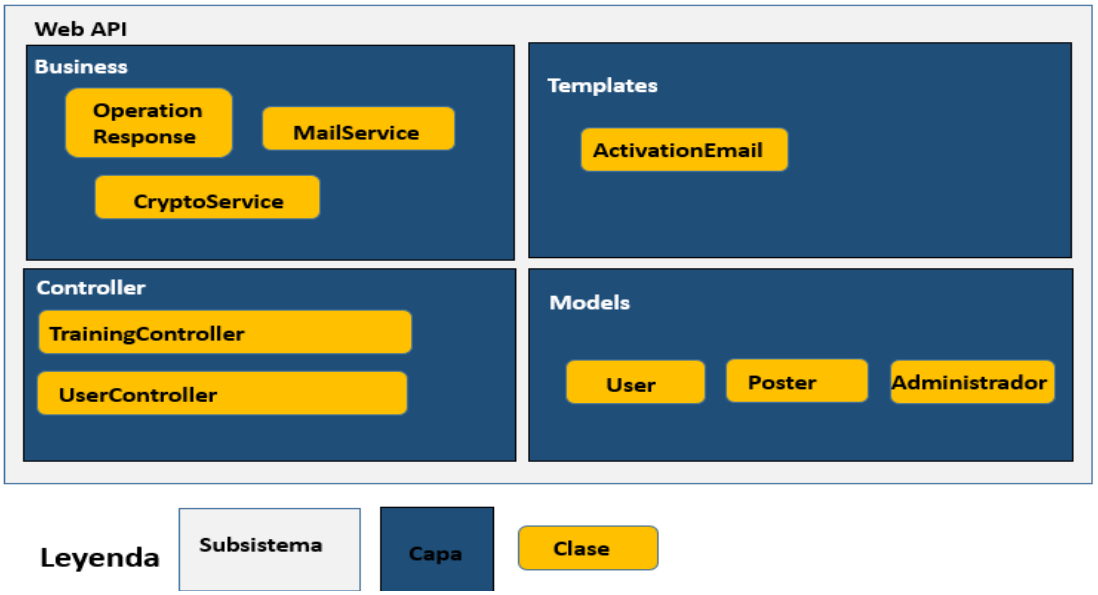

FIGURA 7 DECOMPOSITION STYLE - WEB API

Element Catalog:

Business: Contiene clases auxiliares para soportar las reglas de negocio del API.

- **OperationResponse**: Modelo que estructura la respuesta de los controladores del API.
- **EmailService**: Modelo de servicio para el envío de correos electrónicos para la activación de usuarios.
- · **CryptoService**: Modelo de servicio de tokenización y encriptación del API.

Templates: Contiene los aspectos visuales que maneja el API.

- **ActivationEmail**: Representa la vista enviada como contenido en el correo electrónico.

Controller: Contiene los controladores del API donde se reciben las peticiones de agentes exteriores.

- **TrainingController**: Controlador que maneja las peticiones relacionadas con la sesión de entrenamiento.
- ·**UsersController**: Controlador que maneja las peticiones relacionadas con los usuarios del sistema.

Models: Contiene los modelos que estructuran los objetos que se manejan en la aplicación y los atributos de cada uno.

- **Administrador**: Modelo qué los atributos de los usuarios que existen como atletas dentro del sistema.
- **User**: Modelo que tiene los atributos que identifican al usuario como parte del sistema.
- **Póster** :Modelo que tiene los atributos que definen la información personal del usuario como persona individual.

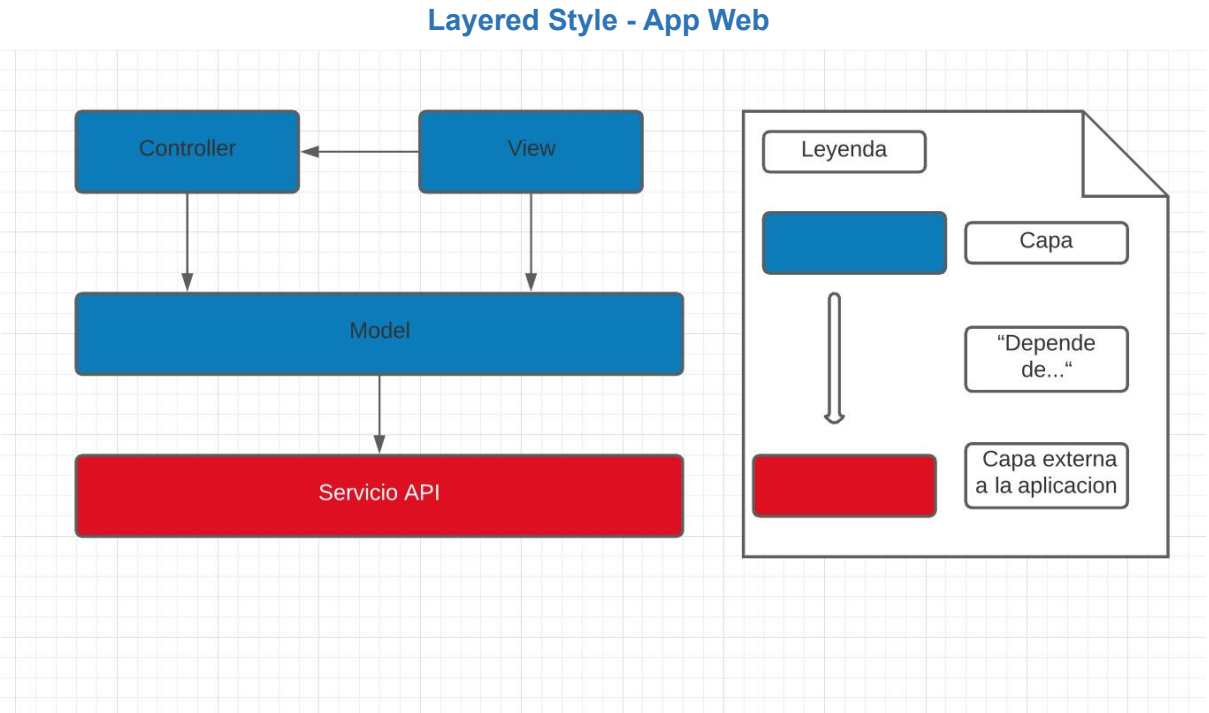

FIGURA 8 LAYERED STYLE - APP WEB

Element Catalog:

- **View**: Es la capa que representa las vistas de la aplicación web. Está Encargada de mostrar la data transformada del Model en una representación visual de la misma.
- **Controller**: Es la capa núcleo de la arquitectura y controla la lógica de la aplicación.Está encargada de la interacción entre el View y el Model, y es responsable de recibir inputs del View y trabajar con el Model para retornar vistas al View.
- **Model**: Es la capa que representa la lógica de negocio y la data. Está encargada de la obtención y guardado de la data en la base de datos.
- **API**: No es específicamente una capa de la arquitectura, sin embargo, es el servicio que provee la información para el modelo.

Layered Style - Web API

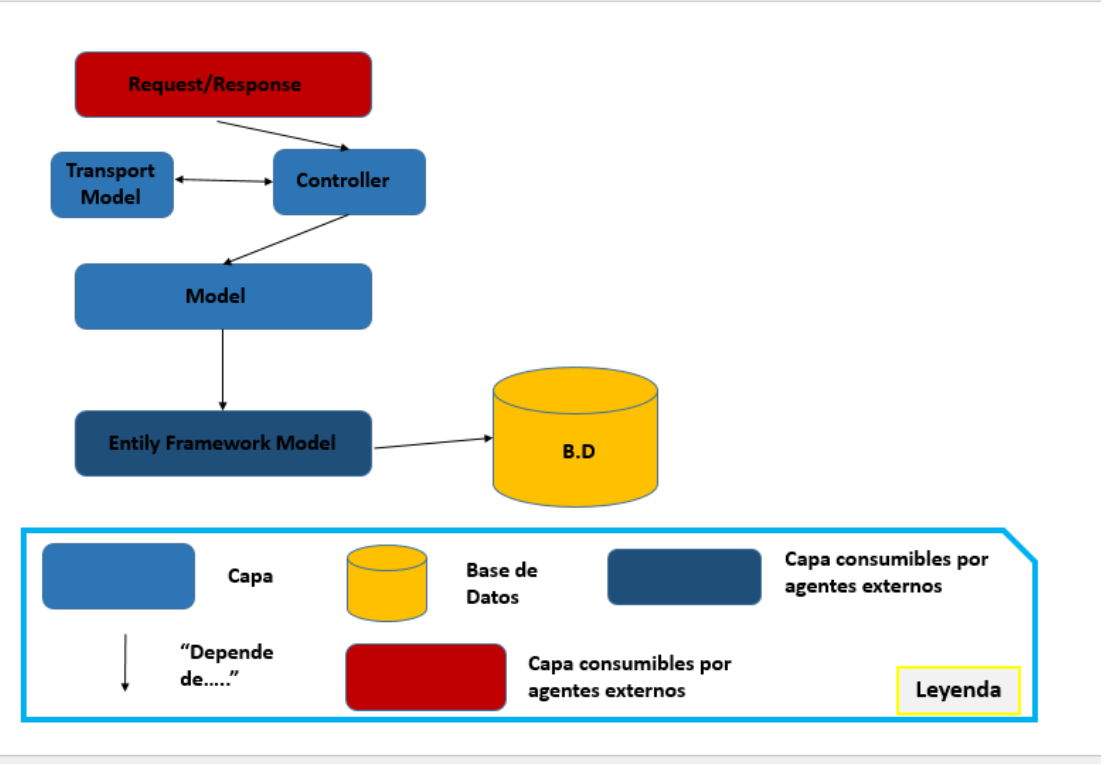

FIGURA 9 LAYERED STYLE - WEB API

Element Catalog:

- **Base de Datos**: Es la capa donde se almacena la data del sistema.
- **Entity Framework Model**: Es la capa que representa la lógica de negocio y la data. Está encargada de la obtención y guardado de la data en la base de datos Esta lógica se construye.a base de una base de datos ya estructurada.
- **Model**: Es una capa donde se modelan clases para imitar el de la base de datos con el fin de que se modifiquen y manejen sus datos sin manejar las clases de las entidades.
- **Controller**: Es la capa que maneja las peticiones y respuestas que vienen de otras aplicaciones al servicio web, se encarga de administrar los recursos de otras capas para procesar la información y hacerla llegar a su destino.
- **Transport Model**: Modelos de datos que se utilizan para serializar la información en las peticiones y respuestas.
- **Request/Response**: Archivos de texto en formato json que se reciben como peticiones para el Web API y se envían como respuesta a los clientes.

Vista Dinámica

Component and Connector View

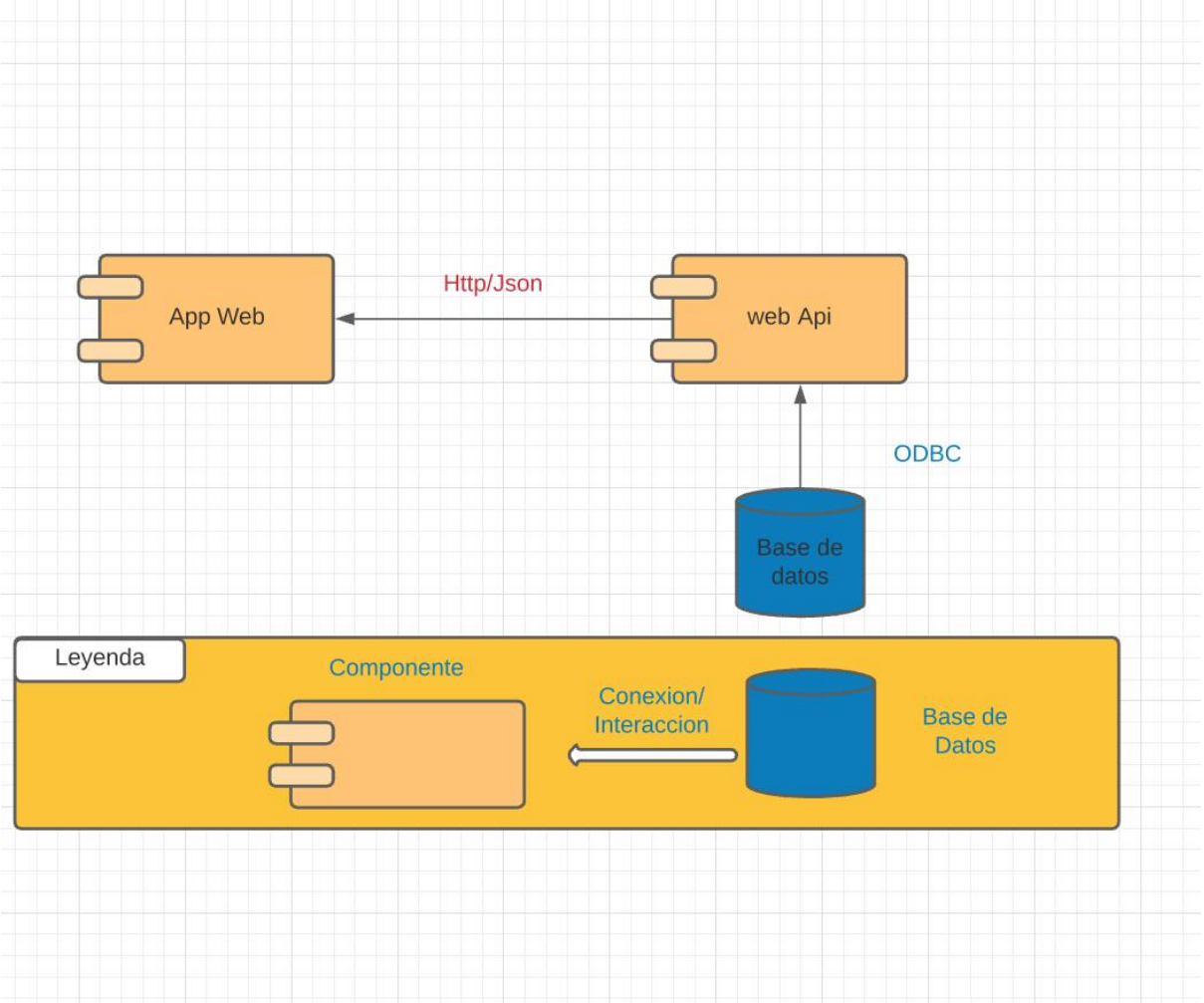

FIGURA 10 COMPONENT AND CONNECTOR VIEW

Vista Física

Allocation View – Deployment Style

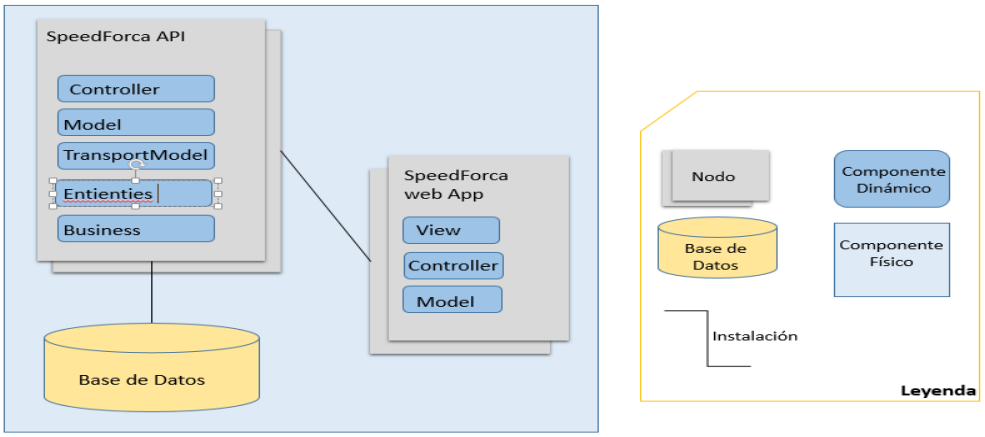

FIGURA 11 ALLOCATION VIEW - DEPLOYMENT STYLE

Element Catalog:

- **Web Tier:** Es el elemento que representa el alojamiento de todo componente que trabaja en la web.
- **Speedforca Web App:** Representa el software de la aplicación web para el uso de los entrenadores.
- **Business:** Son componentes que dictan ciertas reglas de negocio del servicio de API.
- **View, Controller, Model:** Son componentes que representan las diferentes capas del modelo MVC sobre la que funciona la aplicación web.
- **Base de Datos:** Representa la base de datos de donde la aplicación web obtiene los datos, y el cual se sincroniza con la base de datos local del dispositivo móvil.

.

Apéndice

Tabla de Desafíos

En la siguiente tabla listamos los desafíos registrados del proyecto y las piezas de software que fueron afectadas por el mismo.

Identificador del Desafío	Nombre del Desafío	Elementos Afectados
DS-001	Lista de Coordenadas	App Web,Base de Datos, Servicio Web
DS-002	Peticiones desde el API	Servicio Web

TABLA 5 DESAFÍOS DEL PROYECTO