PROYECTO IADIRECTORY

Gestión de proyectos de software

20 DE JUNIO DE 2023

UNIVERSIDAD DE CALDAS
Manizales - Caldas

Tabla de Ilustraciones

Contenido

Nombre del proyecto: IADirectory

Descripción general:

Este proyecto implementa una aplicación de Inteligencias Artificiales utilizando la arquitectura Domain-Driven Design (DDD) en C#. La aplicación está diseñada para gestionar IA en una pagina web.

Requisitos previos:

- Visual Studio 2019 o superior.
- .NET Core 3.1 o superior.
- Conocimientos básicos de C# y DDD.

Estructura del proyecto:

- Domain: Contiene las entidades del dominio y las interfaces de los repositorios y servicios del dominio.
- Infrastructure: Implementa los repositorios y servicios del dominio definidos en el proyecto Domain.
- Application: Contiene los casos de uso o servicios de aplicación que orquestan la lógica del dominio.
- Presentation: Incluye las capas de presentación y los controladores de la API.

Dependencias:

- Entity Framework Core: Utilizado para la persistencia de datos.
- AutoMapper: Para mapear objetos entre capas.
- ASP.NET Core: Utilizado para la creación de la API REST.

Configuración:

- La cadena de conexión a la base de datos se encuentra en el archivo appsettings.json en el proyecto EjemploDDD.Presentation.
- Asegúrese de ejecutar las migraciones de Entity Framework para crear la base de datos.

Uso:

La **API REST** expone los siguientes **endpoints** para gestionar:

- **ArtificialIntelligence:**
  - POST /api/ArtificialIntelligence: Agrega una nueva inteligencia Artificial en la base de datos.
  - GET /api/ArtificialIntelligence: Obtiene todas las inteligencias Artificiales de la base de datos.

    - PUT /api/ArtificialIntelligence: Actualiza la inteligencia artificial existente.
    - DELETE /api/ArtificialIntelligence/{id}:Elimina la inteligencia artificial existente.
    - GET /api/ArtificialIntelligence/{id}: Obtiene una inteligencia artificial por su identificador.
- **Auth-Token:**
    - POST /api/auth-token: Método que se encarga de generar el token de acceso.
- **CategoriesAI:**
    - POST /api/CategoriesAI: Crea una categoria en la base de datos.
    - GET /api/CategoriesAI: Obtiene todos las categorias de la base de datos.
      PUT /api/CategoriesAI: Actualiza una categoria existente.
    - DELETE /api/CategoriesAI/{Id}: Elimina una categoria existente.
    - GET /api/CategoriesAI/{Id}: Obtiene una categoria por su identificador.
- **Users:**
    - POST /api/User: Crea un usuario en la base de datos.
    - GET /api/User: Obtiene todos los usuarios de la base de datos.
    - PUT /api/User: Actualiza un usuario existente.
    - DELETE /api/User/{id}: Elimina un usuario existente.
    - GET /api/User/{id}: Obtiene un usuario por su identificador.
    - POST /api/User/RecoverPassword: Recuperar Contraseña del Correo Electronico.

Ejemplos adicionales:

la carpeta “Services” en el proyecto “Application” ejecuta lógica de dominio más compleja, como la validación de reglas de negocio adicionales.

Application.Services.Email: La clase **MailAddress** y la clase **SmtpClient** en C# se utilizan para enviar correos electrónicos utilizando el protocolo SMTP (Simple Mail Transfer Protocol). Documentación básica para ambas clases:

Clase MailAddress

Descripción: La clase MailAddress representa una dirección de correo electrónico. Se utiliza para especificar el remitente y los destinatarios de un mensaje de correo electrónico.

Propiedades:

Address (string): La dirección de correo electrónico.

Clase SmtpClient

Descripción: La clase SmtpClient permite enviar correos electrónicos utilizando un servidor SMTP.

Propiedades:

Host (string): El nombre o la dirección IP del servidor SMTP.

Port (int): El número de puerto del servidor SMTP.

EnableSsl (bool): Indica si se utiliza SSL para la comunicación con el servidor SMTP.

Credentials (ICredentialsByHost): Las credenciales utilizadas para autenticarse en el servidor SMTP.

Métodos principales:

Send: Envía un mensaje de correo electrónico utilizando el servidor SMTP configurado en SmtpClient.

Servicios Application.Services.ArtificialIntelligence, Application.Services.CategoriesAI, Application.Services.User: proporciona métodos para crear, obtener todo , actualizar, eliminar y obtener por su identificación. Todos cumplen con la tarea principal de Create, GetAll, Update, Delete, GetById.

Estos tres servicios ofrecen una funcionalidad similar.

Application.Services.Auths: El servicio Services.Auths se encarga de gestionar la autenticación y autorización de los usuarios de la aplicación. Su objetivo principal es garantizar que solo los usuarios autorizados puedan acceder a determinadas funcionalidades o recursos.

Application.Services.SaveImage: El servicio Services.SaveImage se encarga de recibir una imagen como entrada y guardarla en un sistema de almacenamiento determinado, como el sistema de archivos del servidor o una base de datos. Su objetivo principal es proporcionar una forma sencilla y segura de almacenar imágenes dentro de la aplicación.

Application.Services.WeeklyTaskService: cómo crear una tarea que ejecuta el GenerateEmail() cada semana o cada minuto. La tarea contiene un desencadenador diario que especifica cuándo se ejecuta la tarea y una acción ejecutable que ejecuta el GenerateEmail().

Uso de patrones de diseño:

- Patrón de diseño Mediator: El patrón de diseño Mediator se utiliza para facilitar la comunicación y la interacción entre diferentes objetos, evitando que los objetos se acoplen directamente entre sí. En lugar de que los objetos se comuniquen directamente, utilizan un objeto mediador que se encarga de orquestar y coordinar las interacciones entre ellos.
- Patrón de diseño CQRS: CQRS es un patrón de diseño de software que nos muestra cómo separar la lógica de nuestras aplicaciones para separar las lecturas de las escrituras.
- Patón de diseño inyeccion de dependencias: La inyección de dependencia nos permite inyectar otras clases y añadir funcionalidad transversal a medida.
- Patrón Repositorio (Repository Pattern) y Unidad de Trabajo (Unit Of Work):

El patrón repositorio consiste en separar la lógica que recupera los datos y los asigna a un modelo de entidad de la lógica de negocios que actúa sobre el modelo, esto permite que la lógica de negocios sea independiente del tipo de dato que comprende la capa de origen de datos.
Este centraliza las conexiones a la base de datos realizando un seguimiento de todo lo que sucede durante una transacción cuando se usan capas de datos y revertirá la transacción si Commit() no se ha invocado o existen incongruencias lógicas.

Modelo Entidad - Relación:

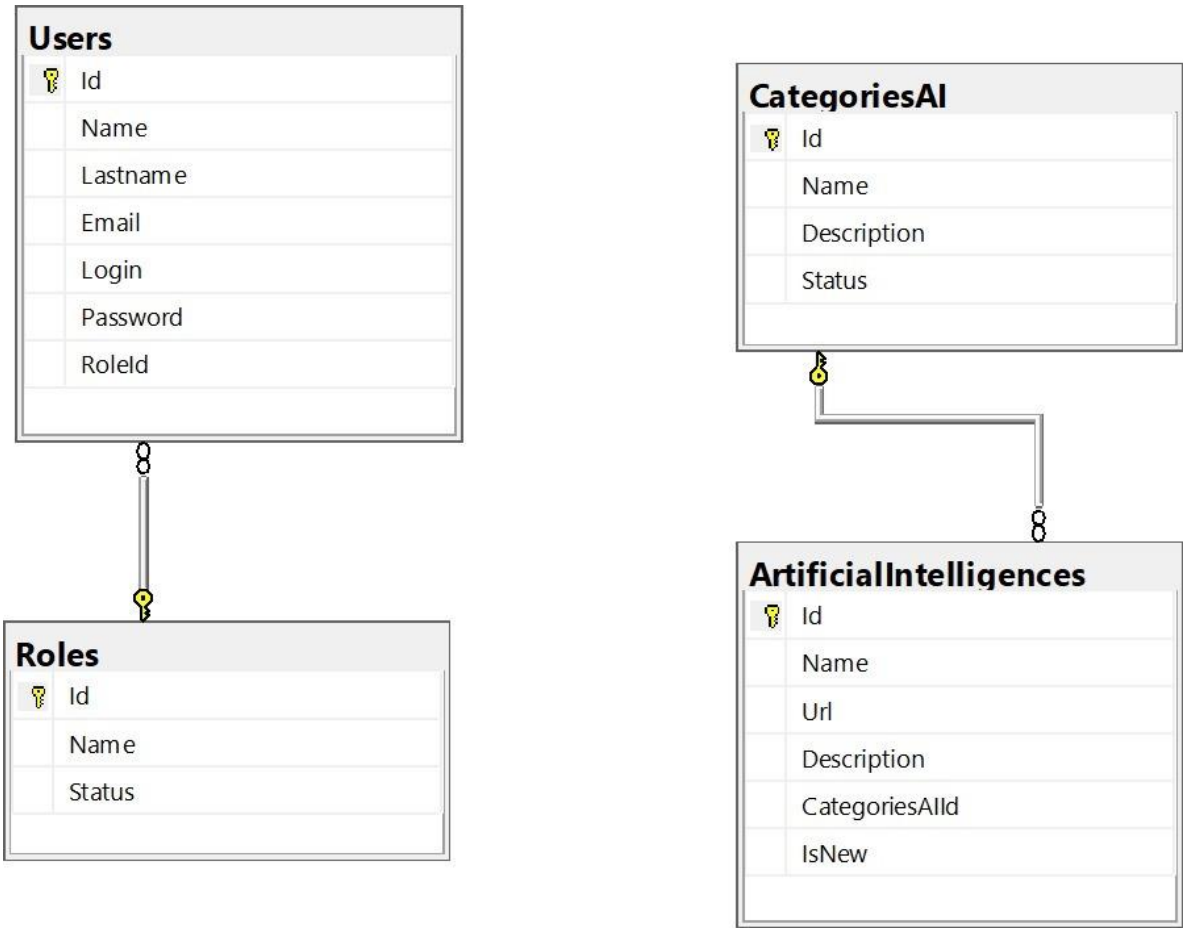

Ilustración 1 MER

En el caso, la entidad principal es "Usuario" y la entidad relacionada es "Rol". La relación entre ellas es de uno a muchos, lo que significa que un usuario puede tener asociado varios roles, pero un rol solo puede estar asociado a un usuario.

- Entidad "Usuario":
  Atributos:
  Id (identificación del usuario)
  Name (nombre del usuario

Otros atributos relevantes (como Email, Password, etc.)

- Entidad "Rol":
  Atributos:
  Id (identificación del rol)
  Nombre (nombre del rol)
  Status (descripción del rol)
- Relación "Usuario - Rol"
  Un usuario puede tener asociados varios roles (relación uno a muchos).
  Un rol solo puede estar asociado a un usuario.

En un modelo entidad-relación (MER), la relación uno a muchos entre una categoría y una inteligencia artificial implica que una categoría puede tener varias inteligencias artificiales asociadas, pero una inteligencia artificial solo puede estar asociada a una categoría:

- Entidad "Categoría":
  Atributos:
  Id (identificador de la categoría)
  Name (nombre de la categoría)
  Otros atributos relevantes (como description, status.)
- Entidad “ArtificialIntelligences":
  Atributos:
  Id (identificador de la inteligencia artificial)
  Name (nombre de la inteligencia artificial)
  Url (url de la inteligencia artificial)
  Otros atributos relevantes
- Relación "Categoría – Inteligencia Artificial":
  Una categoría puede tener asociados varias inteligencias artificiales (relación uno a muchos)
  Una intelegencia artificial solo puede estar asociada a una categoría.