SCHOOLME

ACADEMICS

Manual Tecnico Backend

SCHOOLME

ACADEMICS

SCHOOLME ACADEMICS

Grupo de Desarollo SchollMe

Jesus Fernan

Santiago Chaparro Riaño alfasan1481@gmail.com

Centro de la industria, la empresa y los servicios

ADSO 2900177

1. Introducción

El presente Manual Técnico del Backend de SchoolMe tiene como finalidad documentar de manera estructurada y detallada los aspectos técnicos, arquitectónicos y de instalación del componente servidor de la plataforma SchoolMe, una solución tecnológica orientada a la digitalización y optimización de los procesos académicos y administrativos en instituciones educativas.

Este documento proporciona una guía técnica integral que facilita la instalación, configuración, mantenimiento y despliegue del backend, asegurando la correcta comunicación entre los diferentes componentes del sistema: frontend web, aplicación móvil y base de datos.

El backend de SchoolMe ha sido desarrollado en C#, bajo el framework .NET Core, siguiendo una arquitectura en N capas que promueve la separación de responsabilidades y la escalabilidad del sistema. Dicha arquitectura está conformada por las siguientes capas o proyectos dentro de la solución:

- **Entity:** define las entidades de dominio, modelos de configuración y DTOs compartidos.
- **Data:** implementa el acceso a datos mediante los patrones *Repository* y *CQRS*, además de las configuraciones de persistencia con **Entity Framework** y el soporte para múltiples motores de base de datos (PostgreSQL, SQL Server o MySQL).
- **Business:** contiene la lógica de negocio principal, aplicando principios **SOLID** y **P.O.O.**, junto con patrones de diseño como *Factory* para el manejo modular de los servicios.
- **Utilities:** agrupa funcionalidades transversales como utilidades JWT, manejo de excepciones, servicios de almacenamiento en **Azure** o local, y gestión de caché con **Redis**.
- **Web:** corresponde al proyecto **ASP.NET Core API**, responsable de exponer los servicios RESTful que interactúan con los clientes web y móviles.

La aplicación se encuentra contenedorizada para su despliegue mediante Docker Compose, donde se orquestan tres servicios principales:

1. **API Backend** (ASP.NET Core).
2. **Base de Datos** (principalmente PostgreSQL).
3. **Redis Server** como sistema de almacenamiento en caché para mejorar el rendimiento.

Adicionalmente, el sistema incluye soporte para servicios de correo electrónico y almacenamiento de archivos (imágenes o documentos) de forma local o en la nube mediante Azure Blob Storage.

En conjunto, este manual busca estandarizar los procesos técnicos del backend, facilitar la comprensión de su estructura interna, y ofrecer un punto de referencia para desarrolladores y administradores que deseen instalar, mantener o extender la plataforma SchoolMe de manera correcta y segura.

1.1 Objetivo del documento

El presente manual técnico tiene como propósito describir detalladamente la estructura, configuración, instalación y funcionamiento del **backend de la plataforma SchoolMe**, desarrollado bajo el framework

ASP.NET Core.
Este documento sirve como guía de referencia para desarrolladores, administradores de sistemas y personal técnico encargado de la **implementación, mantenimiento y actualización del sistema backend** de la aplicación.
El objetivo principal de este manual es **garantizar la comprensión y correcta implementación del entorno servidor**, asegurando la adecuada comunicación entre los diferentes componentes del sistema (backend, frontend web, frontend móvil y base de datos).
Asimismo, busca facilitar la **replicación, despliegue y personalización** de la solución SchoolMe en diferentes instituciones educativas, manteniendo su integridad funcional y compatibilidad con los requerimientos definidos en el documento SRS.
En particular, este documento permite:

- Proporcionar las instrucciones necesarias para la instalación, configuración y ejecución del backend.
- Documentar la estructura interna del sistema, sus dependencias y servicios principales.
- Establecer lineamientos técnicos para la integración con la base de datos y otros componentes del sistema.
- Servir de guía de soporte para futuras actualizaciones, correcciones y despliegues en entornos productivos o de prueba.

1.2 Alcance del sistema

El sistema SchoolMe tiene como alcance el desarrollo y funcionamiento de una plataforma digital de gestión académica y administrativa orientada a instituciones educativas. El backend de la aplicación constituye el núcleo funcional encargado de procesar, administrar y centralizar la información académica, garantizando la comunicación eficiente entre los distintos módulos y usuarios del sistema.
Dentro de su alcance, el sistema permite:

- **Gestión de personal institucional:** registro, actualización y control de los diferentes actores del sistema (administradores, docentes, estudiantes, padres y acudientes).
- **Configuración académica:** administración de periodos escolares, horarios, materias, grupos y asignaciones docentes.
- **Gestión de agendas escolares:** creación y administración de agendas digitales por grado, incluyendo el registro diario de observaciones académicas y comportamentales.
- **Carga académica:** asignación de docentes, materias y horarios a cada grupo académico.
- **Módulo de seguridad:** manejo de roles, permisos y control de acceso a los distintos módulos del sistema.
- **Visualización y confirmación de agenda:** acceso de los acudientes a la información diaria de los estudiantes y confirmación de lectura.
- **Gestión de notas académicas:** consulta de calificaciones por parte de los acudientes.

El backend está diseñado para soportar estos procesos mediante una arquitectura modular en N capas, con una base tecnológica basada en .NET Core, C#, Entity Framework, Redis y PostgreSQL.
Asimismo, el sistema está preparado para operar dentro de un entorno contenedorizado con Docker Compose, donde se orquestan los servicios de la API, la base de datos y el servidor de caché.

Quedan fuera del alcance del sistema:

- Procesos contables, financieros o de pagos en línea.
- Personalización visual o de marca institucional (propia del frontend).
- Integraciones externas con plataformas de terceros (no contempladas en esta fase).
- Módulos de evaluación o analítica avanzada.

En esta primera fase, SchoolMe se centra en proveer una solución integral para la gestión de usuarios, estructura académica y seguimiento escolar diario, consolidando los fundamentos técnicos para su futura expansión.

1.3 Descripción general de la arquitectura

La arquitectura general de SchoolMe se basa en un enfoque modular y multicapa (N-capas), diseñada para garantizar la escalabilidad, mantenibilidad y separación de responsabilidades entre los distintos componentes del sistema.
El sistema está conformado por tres pilares principales: Backend, Frontend y Base de datos, los cuales se comunican mediante servicios web RESTful y se despliegan en un entorno contenedorizado con Docker Compose.

Arquitectura general
El sistema se estructura bajo una **arquitectura distribuida** conformada por los siguientes contenedores:

1. **API Backend (ASP.NET Core):** contiene la lógica de negocio, controladores y servicios de comunicación con la base de datos.
2. **Servidor de Base de Datos (PostgreSQL, SQL Server o MySQL):** gestiona la persistencia y consulta de datos.
3. **Servidor Redis:** actúa como sistema de almacenamiento en caché y soporte para operaciones de sesión temporal o colas de procesamiento.

Estos servicios se **orquestan mediante Docker Compose**, permitiendo el despliegue simultáneo, la portabilidad entre entornos y la independencia de cada componente dentro de su propio contenedor.

Arquitectura del Backend

El backend de SchoolMe está desarrollado en C# utilizando .NET Core, siguiendo una arquitectura en N capas que aplica principios SOLID y Programación Orientada a Objetos (P.O.O.).
Cada capa tiene una responsabilidad claramente definida, promoviendo el desacoplamiento y la reutilización del código.

Estructura de la solución:

Nombre de la solución: BackendSchoolMe

Capa / Proyecto	Descripción
Entity	Contiene las entidades de dominio, modelos de configuración, enumeraciones y DTOs. Define la estructura de los datos compartidos entre capas. Incluye las

	carpetas Context, ConfigModels, Model, Enum y Dtos.
Data	Implementa el acceso a datos mediante los patrones **Repository** y **CQRS**. Contiene los contratos, las implementaciones y la carpeta Infrastructure con la configuración de Entity Framework, interceptores y migraciones.
Business	Encapsula la lógica de negocio. Integra las capas Implements (módulos funcionales como Auth, Commands, Queries, Parameters, etc.) y Interfaces (definiciones contractuales de servicios). Aplica el patrón **Factory** para la creación de objetos y servicios de dominio.
Utilities	Proporciona herramientas transversales al sistema: gestión de excepciones, autenticación JWT, comunicación con Redis, almacenamiento de archivos (local o Azure) y mapeadores (MappersApp) organizados por dominio.
Web	Proyecto **ASP.NET Core API** que expone los endpoints RESTful. Contiene los controladores, middlewares, configuraciones (Program.cs, appsettings.json), archivos estáticos (wwwroot), y el **Dockerfile** para la construcción del contenedor de la API.

Patrones y principios aplicados:

- **Patrones de diseño:** *Repository, CQRS, Factory*.
- **Principios de desarrollo:** *SOLID, POO, separación de responsabilidades, inyección de dependencias*.
- **Servicios integrados:** envío de correos electrónicos, almacenamiento de imágenes, manejo de caché y autenticación basada en tokens JWT.

1.4 Público objetivo del manual

El presente manual técnico está dirigido principalmente al personal técnico encargado de la instalación, configuración, mantenimiento y evolución del backend de la plataforma SchoolMe. Su contenido está orientado a profesionales con conocimientos en desarrollo de software, administración de sistemas y despliegue de aplicaciones en entornos de servidor o contenedores.

De manera específica, este documento está destinado a los siguientes perfiles:

- **Desarrolladores Backend:** responsables de implementar nuevas funcionalidades, mantener la lógica de negocio y garantizar la correcta interacción entre las capas del sistema.
- **Ingenieros de DevOps o Administradores de Sistemas:** encargados del despliegue, configuración y supervisión del entorno de ejecución del backend, incluyendo la orquestación con Docker, el manejo de contenedores y la gestión de servicios auxiliares (Redis, Base de Datos, etc.).
- **Integradores o Desarrolladores Full Stack:** encargados de conectar el backend con los clientes web y móviles, consumiendo los endpoints expuestos por la API RESTful.
- **Técnicos de soporte y mantenimiento:** responsables de monitorear la operación del sistema, aplicar actualizaciones y resolver incidencias relacionadas con la infraestructura o los servicios del backend.
- **Estudiantes o equipos académicos** que deseen comprender la estructura técnica y arquitectónica del proyecto con fines de estudio, documentación o mejora continua.

El nivel de conocimiento esperado para la correcta comprensión y aplicación de este manual incluye:

- Fundamentos de **Programación Orientada a Objetos (P.O.O.)** y principios **SOLID**.
- Experiencia básica o intermedia en **C# y .NET Core**.
- Conocimientos de **Entity Framework**, **RESTful APIs**, y **bases de datos relacionales** (preferiblemente PostgreSQL).
- Familiaridad con **Docker, Docker Compose** y conceptos básicos de orquestación de servicios.
- Conocimientos generales sobre **configuración de entornos en Windows y Linux**, manejo de variables de entorno y servicios de red.

Este manual **no está dirigido a usuarios finales** (como administradores, docentes o acudientes del sistema), sino al **equipo técnico** encargado del desarrollo y operación del componente servidor de **SchoolMe**.

1.5 Convenciones y nomenclaturas utilizadas

Para mantener la coherencia, legibilidad y estandarización del código dentro del proyecto **SchoolMe**, se han establecido una serie de convenciones y reglas de nomenclatura aplicadas de manera uniforme en todas las capas del backend.
Estas convenciones siguen las buenas prácticas recomendadas por la comunidad de desarrollo de **.NET** y los lineamientos de **Microsoft C# Coding Standards**.

Estilo general de codificación

- El código fuente se desarrolla íntegramente en **C#**.
- Se aplican los principios de **Programación Orientada a Objetos (P.O.O.)** y **SOLID**.
- Todo el código y los nombres de identificadores (clases, métodos, variables, propiedades, etc.) se escriben en **inglés**, garantizando consistencia y comprensión internacional del proyecto.
- Los nombres deben ser **descriptivos y autoexplicativos**, evitando abreviaturas ambiguas.
- Se recomienda mantener una estructura clara con **sangrías de 4 espacios** y comentarios XML o de bloque cuando sea necesario documentar el comportamiento de un método o clase.

Convenciones de nomenclatura

Elemento	Convención	Ejemplo
Clases	PascalCase	UserManager, StudentService, AgendaController
Interfaces	Prefijo I + PascalCase	IUserRepository, IAgendaService
Métodos	PascalCase	GetUserById(), CreateAgenda()
Propiedades	PascalCase	FirstName, CreatedDate, IsActive
Variables locales	camelCase	userList, agendaItem, dbContext
Parámetros de métodos	camelCase	userId, agendaRequest, emailAddress
Constantes	UPPER_CASE con guiones bajos	MAX_RETRY_COUNT, DEFAULT_PAGE_SIZE

Enumeraciones (Enums)	PascalCase para el tipo y valores	UserRole.Admin, UserRole.Teacher
Nombres de archivos	Igual que la clase principal contenida	UserService.cs, AuthController.cs
Namespaces	PascalCase según estructura de carpetas	SchoolMe.Business.Auth
Migraciones EF	Prefijo con fecha y descripción corta	2025_03_10_CreateUserTable

2. Requisitos del sistema

El correcto funcionamiento del backend de SchoolMe depende de una configuración adecuada del entorno de ejecución, tanto a nivel de hardware, software como de conectividad de red. Esta sección define los requisitos mínimos y recomendados necesarios para garantizar un desempeño estable, seguro y eficiente del sistema en entornos de desarrollo, pruebas o producción.

2.1 Requisitos de hardware

Los siguientes parámetros establecen las especificaciones mínimas y recomendadas para los equipos o servidores donde se ejecutará el backend de **SchoolMe**, ya sea de forma local o en entornos virtualizados o en la nube.

Recurso	Mínimo requerido	Recomendado (Producción)
Procesador (CPU)	Intel Core i3 / AMD Ryzen 3 (2 núcleos, 2.4 GHz)	Intel Core i5 / Ryzen 5 o superior (4 núcleos, 3.0 GHz)
Memoria RAM	4 GB	8 GB o más
Almacenamiento	10 GB libres en disco HDD	20 GB en SSD
Tipo de almacenamiento	HDD	SSD (preferible para rendimiento en consultas de base de datos)
Resolución de pantalla (entornos de desarrollo)	1366x768	1920x1080
Conectividad	Internet estable ≥ 5 Mbps	Banda ancha ≥ 10 Mbps
Virtualización / Contenedores	Compatible con Docker y Docker Compose	Docker Engine y Compose actualizados

2.2 Requisitos de software

El backend de **SchoolMe** se basa en el framework **.NET Core** y requiere de herramientas específicas para su ejecución, compilación y despliegue.

Componente	Versión mínima	Recomendada / Soportada	Descripción
Sistema Operativo	Windows 10 / Ubuntu 20.04 LTS	Windows 11 / Ubuntu 22.04 LTS	Entornos de desarrollo y despliegue soportados.
.NET SDK / Runtime	.NET 6.0	.NET 8.0 (LTS)	Framework principal del backend.
Motor de base de datos	PostgreSQL 12 / SQL Server 2019 / MySQL 8	PostgreSQL 15	Base de datos relacional compatible con Entity Framework.
Redis	v6.0	v7.2	Servidor de caché y almacenamiento temporal.
Docker Engine	20.10	26.0 o superior	Ejecución de contenedores.
Docker Compose	v2.0	v2.27 o superior	Orquestación de contenedores (API, DB, Redis).
Visual Studio / VS Code	2022 / Última versión estable	2022 con extensión C# y Docker	IDE de desarrollo y depuración.
Git	2.30	Última versión estable	Control de versiones.
Azure CLI (opcional)	2.50	Última versión	Para despliegue en Azure Storage o App Services.

2.3 Requisitos de red y conectividad

El sistema requiere conectividad estable para la correcta comunicación entre los contenedores y servicios externos.

Elemento	Requisito / Configuración
Puertos expuestos (API)	5000 (HTTP), 5001 (HTTPS)
Puertos base de datos (PostgreSQL)	5432
Puerto Redis	6379
Firewall	Permitir conexiones internas entre contenedores Docker en red bridge o overlay
Protocolo de comunicación	HTTP/HTTPS con formato JSON
Requisitos externos opcionales	Acceso saliente para servicios de correo (SMTP) y Azure Blob Storage

2.4 Versiones y dependencias

El backend de **SchoolMe** utiliza diversas dependencias y paquetes de NuGet para su correcto funcionamiento.

A continuación, se listan las principales bibliotecas empleadas y su propósito:

Dependencia / Paquete	Versión	Uso principal

	recomendada	
Microsoft.EntityFrameworkCore	8.0.x	ORM para acceso y manipulación de datos.
Microsoft.EntityFrameworkCore.Design	8.0.x	Soporte para migraciones y scaffolding.
Microsoft.EntityFrameworkCore.Tools	8.0.x	Herramientas de CLI para migraciones.
StackExchange.Redis	2.7.x	Conexión y administración del servidor Redis.
Microsoft.Extensions.DependencyInjection	8.0.x	Inyección de dependencias.
Microsoft.AspNetCore.Authentication.JwtBearer	8.0.x	Autenticación basada en JWT.
AutoMapper	12.x	Mapeo entre entidades y DTOs.
FluentValidation	11.x	Validación de modelos de entrada.
MailKit / MimeKit	4.x	Envío de correos electrónicos.
Azure.Storage.Blobs	12.x	Almacenamiento de archivos en Azure (opcional).
Swashbuckle.AspNetCore	6.x	Generación de documentación Swagger/OpenAPI.

Estas dependencias garantizan la **modularidad, seguridad y extensibilidad** del backend, facilitando su mantenimiento y futuras actualizaciones.

3. Preparación del entorno

El backend de **SchoolMe** ha sido diseñado para ejecutarse dentro de un entorno **contenedorizado** mediante **Docker Compose**, lo que permite simplificar la instalación, despliegue y configuración de todos los servicios necesarios (API, Base de datos y Redis) en un único proceso automatizado.
Este enfoque garantiza que el sistema funcione de manera uniforme en entornos de desarrollo, pruebas o producción, independientemente del sistema operativo o las configuraciones locales del usuario.

3.1 Instalación del sistema operativo

El sistema puede ejecutarse en cualquier entorno compatible con **Docker Engine**, preferiblemente bajo alguno de los siguientes sistemas operativos:

Sistema Operativo	Versión recomendada	Compatibilidad
Windows 11 / 10 (Pro o	Última versión	Compatible con Docker Desktop

Enterprise)	estable	
Ubuntu Linux	22.04 LTS o superior	Compatible con Docker Engine nativo
macOS	Monterey o superior	Compatible con Docker Desktop

3.2 Configuración inicial del servidor

Antes de crear los contenedores, asegúrese de cumplir los siguientes pasos previos:

1. **Instalar Docker y Docker Compose:**
   - Verifique la instalación con los comandos:

```bash
bash

docker -v
docker compose version
```

2. Clonar el repositorio del backend:

```bash
bash

git clone https://github.com/SchoolMe/BackendSchoolMe.git
cd BackendSchoolMe
```

3. **Verificar la estructura del proyecto:**
   Dentro del directorio raíz del backend deben encontrarse los siguientes archivos y carpetas principales:

Grupo de Desarollo SchollMe
Jesus Fernando Carvajal jesusanacona017@gmail.com
Santiago Chaparro Riaño alfasan1481@gmail.com

```
mathematica

BackendSchoolMe/
├── Business/
├── Data/
├── Entity/
├── Utilities/
├── Web/
├── docker-compose.yml
└── README.md
```

4. **Configurar las variables de entorno (opcional):**

- El archivo appsettings.json o .env debe contener las credenciales y configuraciones de conexión de la base de datos y Redis, por ejemplo:

```json
{
  "ConnectionStrings": {
    "DefaultConnection": "Host=db;Database=schoolme;Username=postgres;Password=admin123"
  },
  "RedisSettings": {
    "Host": "redis",
    "Port": 6379
  },
  "JwtSettings": {
    "Secret": "ClaveSecretaParaFirmarTokens",
    "Issuer": "SchoolMeAPI"
  }
}
```

3.3 Instalación de dependencias principales

El proyecto **no requiere instalación manual de dependencias** en el sistema anfitrión, ya que todas ellas son descargadas y configuradas automáticamente durante la **construcción del contenedor Docker**.

3.4 Creación y ejecución del contenedor

Una vez clonado el repositorio y configuradas las variables necesarias, el entorno se puede iniciar mediante el siguiente comando desde la raíz del proyecto:

```bash
docker compose up --build
```

Este comando ejecuta los siguientes procesos de forma automática:

1. **Construye la imagen de la API Backend** con todas las dependencias .NET.
2. **Descarga y levanta** los contenedores de **PostgreSQL** y **Redis**.
3. **Crea la red interna** schoolme-network para la comunicación entre servicios.
4. **Inicia todos los servicios** en segundo plano.

3.5 Verificación del entorno

Para confirmar que los servicios se ejecutan correctamente, pueden usarse los siguientes comandos:

```bash
docker ps        # Verifica los contenedores en ejecución
docker logs api  # Muestra los registros de la API
docker exec -it db psql -U postgres -d schoolme  # Acceso a la base de datos PostgreSQL
```

3.7 Observaciones finales

- Todo el entorno se ejecuta de forma **aislada**, sin necesidad de instalar manualmente .NET, PostgreSQL o Redis en el host.
- Las actualizaciones de dependencias o código se aplican reconstruyendo la imagen con el comando --build.
- En despliegues de producción se recomienda configurar variables seguras, certificados SSL y almacenamiento persistente en volúmenes externos.

5. Anexos

Repositorio del proyecto:

https://github.com/JesusCarvajal017/schoolme-api.git

Grupo de Desarollo SchollMe
Jesus Fernando Carvajal jesusanacona017@gmail.com
Santiago Chaparro Riaño alfasan1481@gmail.com