

Prueba de desempeño Modulo .NET

Team Leader: **Javier Cómbita Téllez**

Grupo: **Bernners lee**

Reglas de la prueba:

- **Comunicación:** Está prohibido hablar o comunicarse de cualquier forma con otros estudiantes durante el examen.
- **Integridad Académica:** Cualquier forma de trampa, incluido el plagio, copia o uso de material no autorizado, resultará en una calificación de cero en el examen y puede llevar a sanciones adicionales según las políticas de RIWI.
- **Idioma:** Todo el código debe ser **100% en inglés** incluyendo los comentarios, lo único que se permite en español es la información en sí de los registros en la base de datos
- **Tecnologías Permitidas:**
 - CLI de .NET
 - Visual Studio Code
 - Git
 - GitHub
 - IA bajo los lineamientos descritos en el documento **Reglamento sobre el uso de IA** adjunto en los recursos de la prueba.
- **Material Permitido:** Solo se permite ver material de apoyo como lo son diapositivas o ejercicios realizados en clase, y notas tomadas en el cuaderno. Están permitidas las siguientes páginas para acceso:

[Riwi.io Medellín \(github.com\)](https://github.com)

[Guía de C#: lenguaje administrado de .NET | Microsoft Learn](#)

[Documentación de .NET | Microsoft Learn](#)



- **Permanencia en el Aula:** Una vez iniciado el examen, no se permite salir del aula hasta haber entregado el examen y descanso cada 3 horas.
- **Entrega:** Una vez finalizada la prueba, se debe subir una carpeta comprimida a la plataforma Moodle. La carpeta comprimida debe nombrarse como **PruebaNET_NombreApellido** del coder y debe contener:

La carpeta con el contenido del proyecto en formato .zip.

El proyecto debe incluir un **README bien estructurado**. En este archivo se debe proporcionar toda la información relevante del proyecto, incluyendo el enlace al repositorio de GitHub público, para que sea accesible de manera clara y organizada.

La hora máxima de entrega es a las 14:00 horas. Las pruebas enviadas después de esta hora **NO** serán consideradas.

Medio de Entrega:

El único medio para la entrega es la plataforma Moodle. Se recomienda enviar la prueba 10 minutos antes de la hora límite en caso de eventualidades

Introducción:

En el competitivo sector de la hotelería, la gestión eficiente de reservas es esencial para ofrecer un servicio de calidad. Esta prueba de desempeño busca desarrollar una API RESTful que permita a un hotel gestionar sus reservas de manera eficaz, con un enfoque en las funcionalidades para los empleados.

Trabajarás en una API que ayudará a un hotel a modernizar su sistema de reservas, enfrentando problemas como la falta de centralización y la necesidad de una interfaz amigable para los empleados.



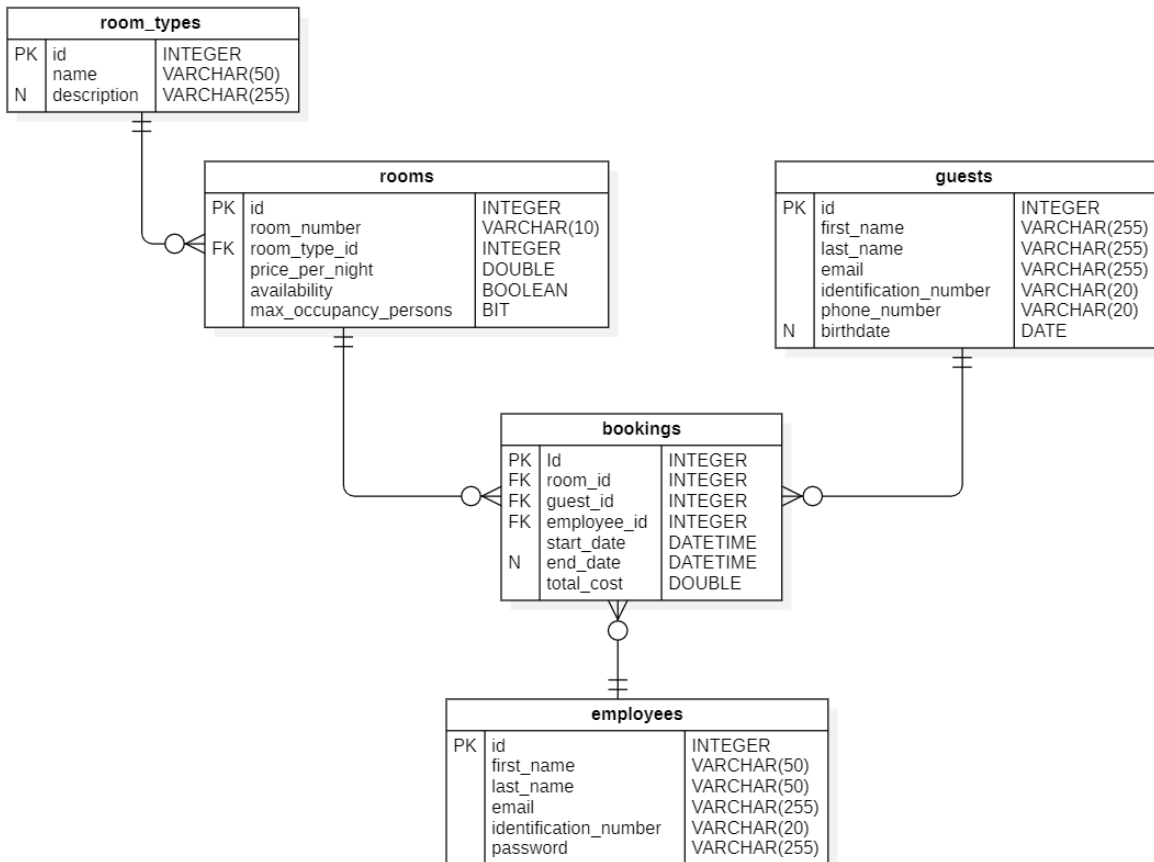
Objetivos clave:

- **Gestión de reservas:** Permitir a empleados y huéspedes crear, consultar, actualizar y eliminar reservas.
- **Disponibilidad de habitaciones:** Facilitar la consulta de habitaciones disponibles y sus tipos para tomar decisiones informadas.
- **Detalles de huéspedes:** Proporcionar información de los huéspedes para mejorar la atención.
- **Interfaz en Swagger:** Garantizar una experiencia intuitiva y bien documentada para facilitar el uso de la API.

Requisitos de la API

Introducción API: Implementa una página de bienvenida en la ruta base de la API mediante middleware. Si no lo haces, configura la API para que cargue directamente la interfaz de Swagger al iniciar, la cual servirá para verificar su funcionamiento y facilitar la exploración de endpoints.

Base de datos con MySQL: Utilizar MySQL como motor de base de datos y configurar migraciones para la creación de la base de datos desde el código. La estructura de la base de datos debe coincidir con el modelo proporcionado, asegurando que todas las entidades y relaciones estén correctamente definidas.



Variables de entorno: Implementar la configuración del acceso a la base de datos y otras configuraciones sensibles mediante variables de entorno para garantizar la seguridad.

Población de datos: Crear datos semilla para la entidad RoomType, asegurando que haya al menos cuatro tipos de habitación predefinidos en la base de datos:

Habitación Simple

- Descripción: Una acogedora habitación básica con una cama individual, ideal para viajeros solos.
- Capacidad: 1 persona
- Precio: \$50/noche

Habitación Doble



- Descripción: Ofrece flexibilidad con dos camas individuales o una cama doble, perfecta para parejas o amigos.
- Capacidad: 2 personas
- Precio: \$80/noche

Suite

- Descripción: Espaciosa y lujosa, con una cama king size y una sala de estar separada, ideal para quienes buscan confort y exclusividad.
- Capacidad: 2 personas
- Precio: \$150/noche

Habitación Familiar

- Descripción: Diseñada para familias, con espacio adicional y varias camas para una estancia cómoda.
- Capacidad: 4 personas
- Precio: \$200/noche

El hotel cuenta con un total de 5 pisos, y en cada piso hay 10 habitaciones, lo que da un total de 50 habitaciones disponibles para los huéspedes. Además, estas habitaciones deben ser cargadas en la base de datos mediante un seeder para asegurar que la información esté disponible al inicio del sistema.

JWT (JSON Web Tokens) para autenticación: Integrar un sistema de autenticación con JWT que proteja los endpoints sensibles y permita a los empleados gestionar las reservas.

Estructurar los controladores aplicando el principio de Responsabilidad Única (S de SOLID): Organizar los controladores de acuerdo con las operaciones HTTP (GET, POST, PUT, DELETE), asegurando una clara separación de responsabilidades en la lógica de negocio.

Uso de DTOs (Data Transfer Objects): Implementar DTOs para asegurar que las entidades del dominio no se expongan directamente a los usuarios.

Validación de datos: Incluir validaciones de datos usando DataAnnotations o Fluent API para garantizar la integridad de la información.

Relaciones entre entidades: Crear y gestionar las siguientes entidades:



- **Room:** Representa las habitaciones.
- **Booking:** Representa las reservas.
- **Guest:** Representa a los huéspedes.
- **RoomType:** Representa los tipos de habitaciones.
- **Employee:** Representa a los empleados del hotel.





Repositorio y servicios: Aplicar el patrón de repositorios y servicios para manejar la lógica de negocio y el acceso a la base de datos de manera eficiente.

Documentación con Swagger: Documentar todos los endpoints utilizando Swagger y garantizar que estén agrupados adecuadamente, sin la documentación mediante comentarios en el código para fomentar la claridad en la API.



Endpoints

Implementar un total de 16 endpoints que permitan realizar las acciones requeridas.








Endpoints Desbloqueados (Sin seguridad)

1.  **POST** /api/v1/auth/login
 - Historia: Como empleado, quiero iniciar sesión para obtener un JWT y gestionar las reservas.
2.  **GET** /api/v1/rooms/available
 - Historia: Como empleado, quiero ver solo las habitaciones que están disponibles para elegir una que no esté ocupada.
3.  **GET** /api/v1/room_types
 - Historia: Como empleado, quiero ver todos los tipos de habitaciones disponibles.
4.  **GET** /api/v1/room_types/{id}
 - Historia: Como empleado, quiero ver los detalles de un tipo de habitación específico.



5.  **GET** /api/v1/rooms/status
 - Historia: Como empleado, quiero ver un resumen de las habitaciones, indicando cuántas están ocupadas y cuántas están disponibles.
6.  **POST** /api/v1/guest
 - Historia: Como huésped, quiero registrarme en la plataforma para que me puedan asignar reservas en el hotel

Endpoints Protegidos (Requieren autenticación JWT)

1.  **GET** /api/v1/bookings/search/{identification_number}
 - Historia: Como empleado, quiero ver todas las reservas de huéspedes por número de identificación.
2.  **GET** /api/v1/bookings/{id}
 - Historia: Como empleado, quiero ver los detalles de una reserva específica.
3.  **POST** /api/v1/bookings
 - Historia: Como empleado, quiero crear una nueva reserva para un huésped.
4.  **DELETE** /api/v1/bookings/{id}
 - Historia: Como empleado, quiero eliminar una reserva que ya no sea necesaria.
5.  **GET** /api/v1/rooms
 - Historia: Como empleado, quiero ver todas las habitaciones del hotel.
6.  **GET** /api/v1/rooms/{id}
 - Historia: Como empleado, quiero ver los detalles de una habitación específica.
7.  **GET** /api/v1/guests



- Historia: Como empleado, quiero ver una lista de todos los huéspedes registrados en el sistema.
- 8. 🔒 **GET** /api/v1/guests/{id}
 - Historia: Como empleado, quiero ver los detalles de un huésped específico.
- 9. 🔒 **DELETE** /api/v1/guests/{id}
 - Historia: Como empleado, quiero eliminar un huésped del sistema.
- 10. 🔒 **GET** /api/v1/guests/search/{keyword}
 - Historia: Como empleado, quiero buscar un huésped utilizando una palabra clave.
- 11. 🔒 **PUT** /api/v1/guests/{id}
 - Historia: Como empleado, quiero actualizar los datos personales de un huésped.
- 12. 🔒 **GET** /api/v1/rooms/occupied
 - Historia: Como empleado, quiero ver solo las habitaciones que están ocupadas para gestionar las reservas adecuadamente.