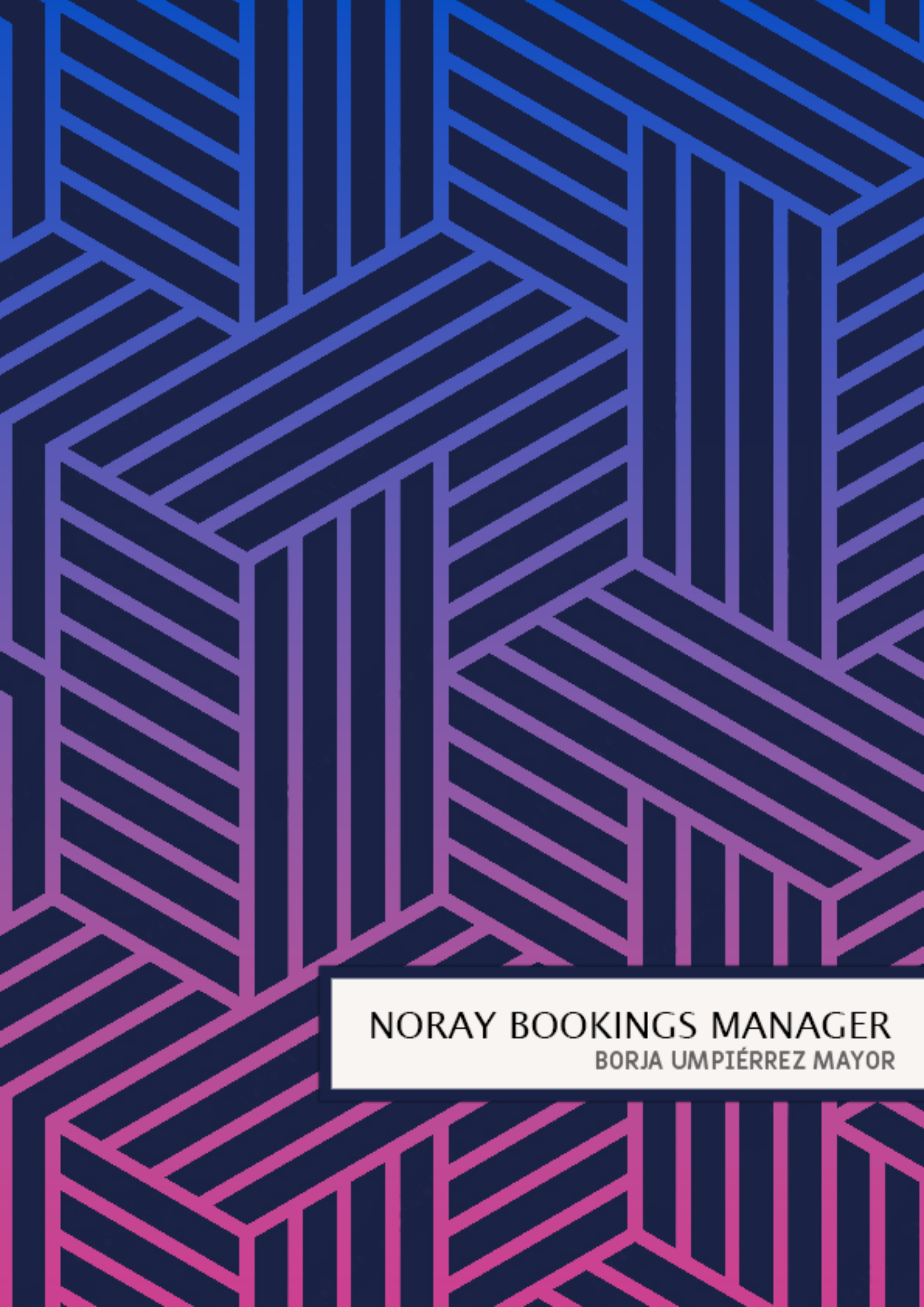
NORAY BOOKINGS MANAGER
BORJA UMPIÉRREZ MAYOR

Sumario

— Noray Bookings Manager —

Información:

- **Nombre y apellidos del alumno:**
  Borja Umpiérrez Mayor
- **Empresa donde se ha desarrollado el Módulo:**
  Noray
- **Lenguajes de programación usados:**
  C# y ASP.NET
- **Plataformas utilizadas, entornos, …:**
  Visual Studio 2019
  SQL SERVER EXPRESS 2012
- **Práctica desarrollada:**
  A continuación podrá leerse la práctica realizada.
  Es aconsejable ver el archivo adjunto con la presentación.

¿Qué es NBM?

Noray Bookings Manager es un proyecto destinado al mundo hotelero por la empresa Noray con la finalidad de permitir una rápida y eficiente gestión de las reservas de diferentes hoteles en las islas.

Mediante solicitudes HTTP el cliente o intermediario puede solicitar información a la empresa en formato XML, tales como las reservas desde semana santa a verano para procesarlas, así como cambiar su estado de procesada a no procesada y viceversa.

¿Cuál ha sido mi trabajo estas dos semanas?

Estas dos semanas han sido un trabajo constante de documentación y aprendizaje tanto de un nuevo lenguaje de programación (C#) y un nuevo software de trabajo (Visual Studio 2019) como el aprendizaje de los principios **SOLID** de desarrollo de software.

Dichos principios han sido un aporte y ayuda más que bien recibida que permiten mantener un código con mayor facilidad y rapidez, añadir nuevas funcionalidades de forma sencilla, y favorecen la reutilización y encapsulación del código.

Pequeño proyecto:

Durante las dos primeras semanas se me encargó analizar el proyecto **NBM** con la finalidad de entender dicho producto, poder modificar y añadir elementos funcionales, así como realizar una breve explicación del mismo.

No puedo mostrar muchas capturas del código dado que contienen información delicada de la empresa, tales como direcciones IP, claves de acceso o usuarios.

En el archivo adjunto con la exposición explico cómo he realizado un cambio que la empresa quería hacer en el proyecto como una cata. El cambio consistía en explicar cómo realizaría un cambio en las reservas de los hoteles, de forma que se pudiese almacenar tanto el país y el código iso del país donde se realiza la reserva, como de cada cliente individualmente.

Principios SOLID (POO):

Son una serie de recomendaciones que permiten la escritura de un mejor código que permita implementar una **alta cohesión** y **bajo acoplamiento**. No todos los códigos necesitan implementar estos principios, sin embargo, es importante aprender cuándo es necesario.

1 (SRP) Single Responsability Principle

Un módulo solo debe tener un motivo para cambiar, es decir, que debe tener una única responsabilidad para evitar dar demasiada carga o demasiada importancia a la misma clase.

Un ejemplo sería una clase destinada almacenar pedidos, a la cual se le asigna una tarea o responsabilidad de enviar notificaciones por correo. Lo mejor sería que hubiese una clase que se encargue del almacenamiento y otra que se encargue del envío.

2 (OCP) Open/Closed Principle

Una clase debe ser fácilmente extendible sin necesidad de modificarse internamente. Por ejemplo, si una aplicación permite enviar mensajes sms o de email, debería poder enviar en un futuro mensajes por otros canales como Telegram o Whatssap.

Para esto es útil la creación de interfaces que contengan un método común que se encargue del envío del mensaje en este caso, el cual, contendrá la lógica.

3 (LSP) Liskov substitution Principle

Una clase hija debería poder ser usada como si fuese la clase padre sin necesidad de alterar su funcionamiento. Esto permite generar herencias mucho más sólidas y estructuradas.

Un claro ejemplo sería indicar que una clase animal puede cazar, caminar y correr, y una tortuga al heredar esta clase tiene accesible cazar y correr cuando no puede. Para solucionarlo se debería crear una interfaz ICorrer, ICazar e ICaminar para permitir una mejor funcionalidad por cada animal.

4 (ISP) Interface Segregation Principle

En lugar de realizar interfaces que contengan muchas implementaciones y métodos, es mejor reducir éstos a componentes modulares que permitan aplicar mejor esos cambios.

Por ejemplo, en lugar de una interfaz CRUD completa, realizar una interfaz IReadable que permita acceder a los datos en modo lectura, IWriteable que permita actualizar o crear, e IRemovable que permita eliminar los datos.

5 (DIP) Dependency Inversion Principle

Las clases de alto nivel no deberían depender de las de bajo nivel, sino que ambas deberían depender de las abstracciones, al mismo tiempo que los detalles deben depender de las abstracciones y no al revés.