# **Formulario: Vista de Desarrollo del Proyecto de Software**

**Proyecto:** Hosteluxe  
**Grupo:** 8  
**Fecha:** 07-10-24

**1. Estructura del sistema**

Describe la estructura del sistema de tu proyecto, mencionando los módulos principales y cómo interactúan entre sí. Puede usar diagramas de arquitectura para complementar tu descripción.

**Descripción de la estructura:**

La base del sistema está conformada principalmente por el modulo de reservas, donde un usuario “X” puede ingresar, reservar para la habitación que el desee. Ya en la habitación él puede realizar pedidos de platos, tragos y todo lo que este en la carta de la propia cafetería/restaurante del hotel, también se extiende un poco más el servicio a la limpieza, donde también se puede pedir servicio al cuarto, etc. Todo esto mediante la página web la cual estará a cargo de un administrador/supervisor que realizara vigilancia del tráfico de la página, tiene una sección de mensajes donde los mismos clientes podrán mandar quejas, dudas, o lo que se requiera.

**Diagrama de la estructura del sistema:**   
*(Adjunta un diagrama UML o un diagrama de bloques que muestre los módulos principales y sus interacciones.)*

**2. Módulos principales**

Enumera y describe los módulos principales del sistema, explicando sus responsabilidades y cómo interactúan con otros módulos.

**Módulo 1** Reserva:  
**Responsabilidad:** procesa y gestiona las reservas del Hotel, en este módulo se basa el sistema   
**Interacción con otros módulos:** La reserva se relaciona a un usuario, lo puede controlar un administrador o secretario, se relacionan pedidos del restaurant/cafetería.

**Módulo 2** Pedido/menú:  
**Responsabilidad:** Gestiona y procesa los pedidos de un menú anteriormente dado, todo desde la vista del cliente donde puede elegir lo que el desee.  
**Interacción con otros módulos:** con módulo de pagos, también se relaciona con reserva y usuario otorgándose a quien lo realizo y de donde lo realizo.

**Módulo 3 Vista Administración:**   
**Responsabilidad:** Controla el flujo del sistema, mensajería, y puede añadir, modificar o eliminar diferentes elementos.  
**Interacción con otros módulos:** Principal recibidor de mensajes, desde el módulo de mensajería. Puede ver y controlar los pedidos del módulo 2. Puede gestionar reservas en caso de que el secretario/a no pueda.

**3. Dependencias y Librerías**

Especifica las dependencias y librerías/frameworks que estás utilizando en tu proyecto, indicando para qué funcionalidad o módulo se usan.

**Dependencias y Librerías:**

* "bcrypt": Para el hasheo o encriptación de contraseñas, se utiliza en diversos módulos (Administración, registro)
* "cookie-parser": Middleware que maneja las cookies de los módulos para que el servidor pueda interpretarlas
* "csurf": Protege los formularios del sistema ante el CSRF o XSRF (malware)
* "date-fns": Manejador de formatos de fecha y horarios
* "dotenv": Crea un archivo donde se pueden crear variables de entorno, sirven para ocultar ciertas claves o códigos personales del proyecto
* "dropzone": Simplifica proceso de subida de archivos al sistema, arrastrar y soltar
* "express": ayuda al manejo de errores, utilización de middleware, etc.
* "express-validator": Biblioteca de validación de datos, principalmente en formularios
* "flatpickr": Se utiliza para crear selectores de fecha y hora
* "jsonwebtoken": Crea “tokens” donde se puede registrar o guardar información temporalmente
* "multer": Maneja la subida de archivos en express y aplicaciones web
* "mysql2": Motor de base de datos
* "nodemailer": se utiliza para el sistema de mensajería interno del software, principalmente para el envío automático de correos de autenticación de usuario
* "nodemon": Solo de uso en desarrollo, se utiliza para reiniciar la aplicación web/servidor cada vez que se realiza un cambio en el código de la aplicación
* "pug": Es el motor de vistas de la aplicación web, similar a HTML5, pero simplificado en cuestiones de codeo, se guía solo por identacion
* "sequelize": Es un ORM que en vez de usar consultas SQL, las realiza desde un código JS el cual facilita la creación de modelos relacionales orientado a objetos.
* "autoprefixer": Facilita el codeo CSS
* "concurrently": Es una herramienta para ejecutar múltiples comandos de manera simultánea en el entorno de desarrollo de Node.js
* "postcss": Permite la utilización de plugins que hacen CSS más simple
* "postcss-cli": Es para procesar archivos CSS directamente de una terminal
* "tailwindcss": Framework que se utiliza para simplificar CSS, lo que le da una gran flexibilidad y control
* "webpack": es un módulo empaquetador para aplicaciones JavaScript que permite agrupar, transformar y optimizar los archivos de tu proyecto en un solo archivo (o en varios archivos) para su uso en el navegador
* "webpack-cli": permite interactuar con las herramientas de webpack mediante el CLI

**4. Organización del Código**

Explica cómo está organizado el código en términos de carpetas y paquetes dentro del proyecto. Incluye ejemplos o un esquema de la estructura de directorios.

**Organización del código:**

Se basa en la metodología MVC (Modelo Vista Controlador), el cual se basa en un controler (donde se manejan todos los métodos del módulo), el cual tiene un router asociado (donde se realiza el routing o se guardan las rutas), todo añadido a un modelo o model el cual es la definición de la tabla para la base de datos y la comunicación con esta (ejemplo: UsuarioModel, PedidoModel, reservaModel, etc..), y todos estos datos se ven en una vista, lo cual cada módulo tiene su carpeta de vistas asociadas.

*(Opcional: Adjunta una imagen o diagrama de la estructura de directorios del proyecto.)*

**5. Reglas de Integración**

Especifica las reglas que tu equipo sigue para la integración del código entre los módulos. Esto puede incluir restricciones sobre el acceso a datos, convenciones de desarrollo, etc.

**Reglas de integración:**

* Los módulos de alto nivel no deben depender de módulos de bajo nivel; ambos deben depender de abstracciones (interfaces)
* Los módulos deben comunicarse a través de interfaces bien definidas. Por ejemplo, si un módulo A necesita datos de un módulo B, debe hacerlo a través de un método público en la interfaz de B.
* Todos los módulos que reciben datos de otros módulos deben validar esos datos antes de usarlos. Esto ayuda a prevenir errores y ataques de inyección o csrf.

**6. Prácticas de desarrollo**

Detalla las prácticas de desarrollo que estás utilizando en el proyecto, como las pruebas unitarias, integración continua, revisión de código, etc.

**Prácticas de desarrollo:**

* Pull Requests: Se utiliza pull requests para que otros revisen el código antes de fusionarlo a la rama principal. Esto permite detectar errores y mejorar la calidad del código.}
* Documentación de código
* Documentación de proyecto
* Se maneja un estilo de programación definido donde todos entienden la misma sintaxis
* Captura de errores (Try, catch)
* Metodología ágil
* Variables de entorno

**7. Herramientas de desarrollo**

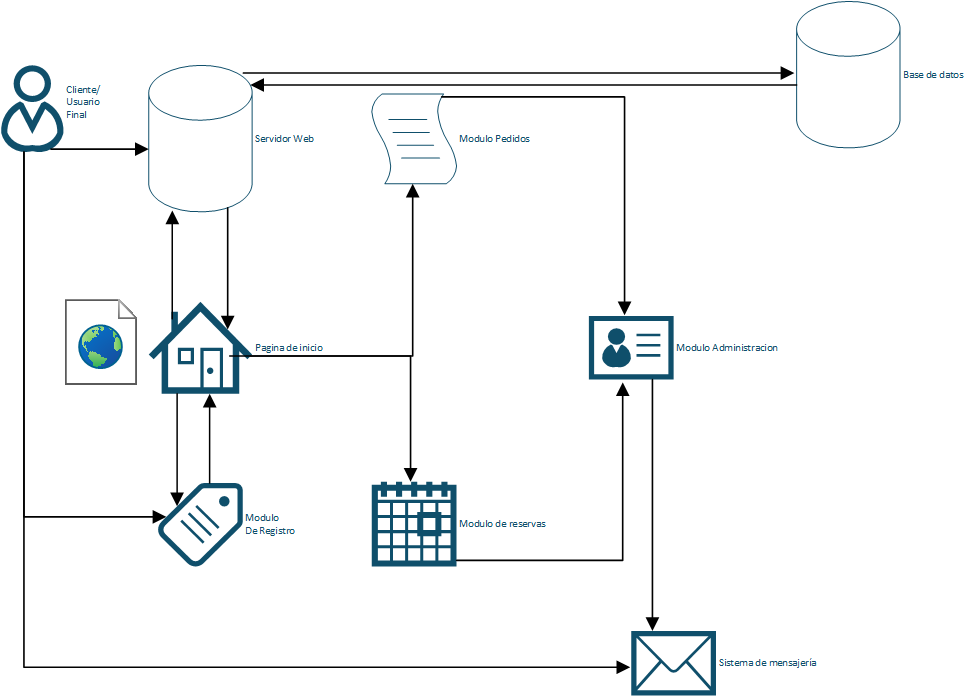
Mencione las herramientas que está utilizando para el desarrollo del proyecto (por ejemplo, IDE, sistemas de control de versiones, etc.).

**Herramientas:**

* Visual estudio code
* Npm
* Npx
* Webpack
* Sequalizer
* Mysql
* Table Plus

**8. Diagrama de Arquitectura del Sistema**

Adjunta un diagrama de arquitectura general del sistema que incluye los módulos, las interacciones entre ellos y las tecnologías involucradas.

**Diagrama de arquitectura general:**   
**