Instituto Industrial Luis A. Huergo
Desarrollo de Sistemas 2025
Cuatrimestre Nº1
Trabajo Práctico N°2

Objetivos

El presente trabajo tiene por objetivo poner en práctica los temas vistos acerca del diseño de sistemas y la tecnología backend web. Si bien habrá acompañamiento de parte del profesor, se espera que los alumnos desarrollen y tengan la autonomía para buscar librerías, documentación y cualquier recurso que consideren necesario para construir su sistema informático.

Se espera que el sistema funcione en su integralidad y cumpla con los requerimientos pedidos.

El sistema deberá poder reconocer dos tipos de usuarios distintos que tengan permisos de diferentes niveles. Se deberá poder interactuar con el sistema mediante una API que reciba requests de HTTP, modificando su estado u obteniendo respuestas de él.

Sobre el sistema a diseñar y desarrollar

Los alumnos pueden optar por la propuesta planteada debajo sobre el restaurante o pueden proponer otro sistema a diseñar y desarrollar. Para ello, el sistema propuesto deberá ser tanto o más complejo que el del restaurante y deberá cumplir con los **Objetivos** planteados para este trabajo. Se deberá acercar la propuesta y ser validada primeramente antes de su puesta en marcha. No se podrán tomar trabajos de otras materias.

API para el restaurante “Lo de Miguel”

Forman ustedes la empresa elegida para el desarrollo de un sistema informático para el restaurante **“Lo de Miguel”.**

El dueño les pide que se puedan manejar: el menú, los clientes, las mesas y los pedidos. Habrá dos tipos de actores que puedan interactuar con el sistema: clientes y un administrador.

Los clientes deben poder registrarse, hacer login e interactuar con el sistema a través de un método de autenticación que dé cuenta de su identidad. Ellos tendrán la posibilidad de reservar una mesa en el restaurante en caso que quieran ir a comer allí o efectuar un pedido.

Debe existir la posibilidad de consultar el menú del restaurante mostrando la variedad de platos que se pueden pedir. El administrador podrá eliminar o agregar platos al menú.

Los clientes deberán poder consultar la disponibilidad de las mesas y podrán reservar tan solo una que se encuentre disponible. Solo el administrador tendrá la posibilidad de disponibilizar una mesa que estaba previamente reservada. Las mesas cuentan un total de 15.

Respecto a los pedidos que pueden realizar los clientes tienen la posibilidad de seleccionar varios platos del menú. Una vez que el cliente solicita un pedido, este se pondrá en estado “pendiente”, luego pasará a “en cocina” y por último pasará al estado “enviado”. La transición entre esos estados será podrá ser manejada únicamente por el administrador. Tanto el administrador como el cliente podrán consultar el estado del pedido.

Los pedidos calcularán el monto total a partir de la sumatoria de los montos de cada plato. Los clientes tendrán un sistema de descuentos en función de la cantidad de pedidos que hayan hecho que aplicará sobre el monto total: de 0 a 3 pedidos no aplica descuento; más de 3 veces serán “Habitue” con un 10% de descuento; más de 5 veces, “Premium”, con un 20% de descuento; más de 7 veces, “TopPremium”, con un 50% de descuento.

Acerca del administrador, este podrá ser autenticado por el sistema mediante algún método de identificación.

Los datos que se manejan en relación a cada entidad son:

- Cliente: nombre, correo, teléfono, contraseña, dirección de domicilio.
- Mesa: número de mesa, estado (reservada o disponible).
- Menú (platos): nombre, descripción, precio, categoría (entrada, plato principal, postre).
- Pedidos: número de pedido, de qué cliente, estado (pendiente, en cocina, entregado), platos, monto total, porcentaje de descuento, domicilio de entrega.

Consigna

1. Se debe diseñar y desarrollar un sistema que cubra todos los requerimientos del cliente. Debe estar hecho con Typescript (Javascript) y debe poder ser ejecutado con Node.JS. Se deberá persistir toda información mediante una integración con una base de datos SQLite. El proyecto debe ser versionado con GIT. Se podrán utilizar cuantas librerías crean convenientes. Y se podrá utilizar el estilo de arquitectura que se crea conveniente: se puede diferenciar en microservicios, puede ser de tipo monolítico, se puede pensar en varias capas, etc.

2. Se debe documentar cada uno de los endpoints de la API definidos, esto es: definir qué método HTTP utiliza; cuáles son los diferentes HTTP status que puede devolver; cuál la ruta del endpoint; en caso de que el endpoint lo demande, qué forma debe tener el body enviado, cuáles son sus campos y sus tipos de datos; qué forma tiene la respuesta, cuáles son sus campos y sus tipos de datos.

Recomendaciones

Plantear primero todos los requerimientos de forma explícita y derivar de ello los casos de uso; definir cuáles serían los endpoints necesarios para atender a los casos de uso; modelar la base de datos en función de los casos de uso y las entidades que se

manejan; diseñar tentativamente qué forma tendrá la estructura del backend. Y si no queda ninguna decisión de diseño preliminar, comenzar a escribir el código.

Ejes de evaluación

El trabajo será evaluado en cuatro medidas distintas: en qué tanto se trabajó durante las clases, en qué grado los requerimientos fueron cubiertos por el sistema, en cuán completo está el trabajo en relación a las consignas y en cuántos de los contenidos vistos hasta la presentación del trabajo fueron aplicados al sistema: patrones de diseño y criterios de diseño de sistemas.

Se prestará especial atención a las ***Condiciones de entrega.***

Condiciones de entrega

Sobre el código fuente del punto 1: debe poder ser descargado de Github. Debe haber un archivo README.md en la carpeta raíz del repositorio que indique quiénes integran al grupo de trabajo, de qué forma se debe compilar el código fuente, las decisiones tomadas respecto al diseño del sistema y cualquier otra cuestión que crean conveniente observar.

El archivo de la base de datos, que debe ser interpretada por *sqlite*, debe hallarse en la carpeta raíz bajo el nombre *“bd”.*

Todo código que haya sido pusheado al repositorio remoto posterior a la fecha de entrega, no será tenido en cuenta para la corrección.

Respecto al punto 2: la documentación de la API se debe encontrar en un documento titulado “*api_doc”* en formato pdf.

Momento límite de entrega: *20/6 23:59hs*