UTN AVELLANEDA
Tecnicatura Universitaria en Programación

Programación III - TP Obligatorio

Autoservicio

Condiciones de aprobación

- El Trabajo Integrador está dividido en dos proyectos. Un proyecto frontend y un proyecto backend.
- La fecha de entrega del TP integrador es la última semana de cursada.
- La promoción del trabajo dependerá de, además del cumplimiento de las consignas individuales de cada proyecto, la correcta implementación y comunicación entre el frontend y el backend.
- En caso de que los alumnos no promocionen el TP Integrador pero si cuentan con la nota suficiente para aprobar la materia, podrán entregar el trabajo a modo de final, junto con todas las consignas agregadas (las de la cursada y las de fecha de final).
- En caso de que los alumnos no cuenten con la nota suficiente para aprobar el trabajo pasadas las fechas de recuperación al final del cuatrimestre, recursarán la asignatura.
- **AMBOS** proyectos deben de ser realizados por **AMBOS** integrantes del grupo. Se **revisarán commits** del repositorio para garantizar este requisito. Cualquier alumno que no cumpla con esto recibirá una quita de puntos individual. Esto puede resultar en que los alumnos del mismo grupo finalicen la cursada con estados de aprobación / promoción distintos, aumentando la exigencia para quienes no cumplan con su parte del trabajo.

UTN AVELLANEDA
Tecnicatura Universitaria en Programación

Requerimientos iniciales

1. Requerimientos funcionales.

Una gran empresa les propuso crear un sistema.

Los profesores actuarán como clientes y testers de su aplicación, quienes revisarán que todos los requerimientos se cumplan y **dictarán cambios acordes y necesarios**.

Este sistema cuenta con **dos proyectos** que poseen **tres partes reconocibles**.

Proyecto frontend:

- Una aplicación **frontend**, la cuál permite a los usuarios comprar distintas variantes de **DOS tipos de productos** a modo de **autoservicio (no e-commerce, se explica más adelante)**. Donde al finalizar la compra, el usuario recibe un ticket con su compra.

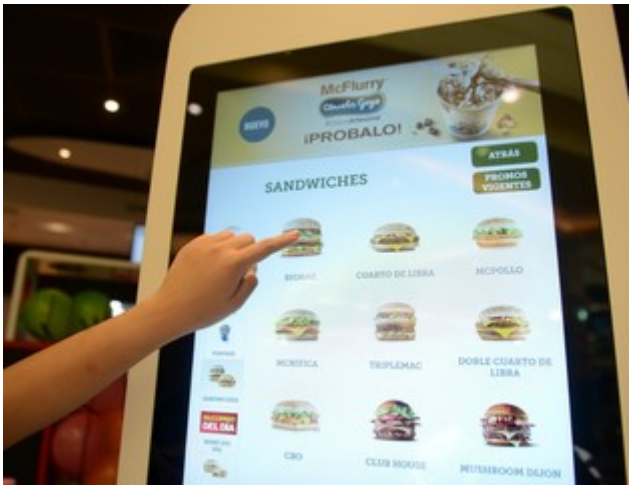
Proyecto backend:

- Una parte la cuál se va a encargar de poseer la base de datos y devolver los datos necesarios a modo de **API**.
- Una parte con **renderizado de vistas HTML** (**EJS**) para los **administradores** a modo de **back office**, la cuál permite realizar cambios en los productos disponibles a la venta (agregar, modificar, eliminar) y más acciones.

  Nota: las partes backend deben estar alojadas en el mismo servidor.

La empresa necesita que la aplicación pueda ser vista tanto en computadoras como en teléfonos móviles (**RESPONSIVE**).

Se adjuntó una imagen de referencia de un popular **autoservicio** de comida. En este ejemplo, los tipos de productos vendidos son hamburguesas y bebidas.



La aplicación puede vender dos tipos de productos de **cualquier tipo que deseen**, pero deben pertenecer al mismo rubro (o en su defecto, que sea lógico que sean vendidos en una misma aplicación).

Nota: aunque la elección es libre, **NO pueden utilizar productos de comida.** Pueden ser productos físicos o virtuales. Queda a su elección.

Las vistas HTML del sistema deben poder ser navegables entre distintos archivos html a través de botones. La navegación a través de botones debe estar **siempre disponible**. El cliente **NO debe tener que escribir una ruta a mano.**

Ambas partes con vistas en html y css deben contar con criterio a la hora de darle estilos, y se va a pedir que lo mejoren si es necesario.

---

2. Pantallas de la aplicación.

TODAS las pantallas deben poseer:

1. Logo de la aplicación
2. Nombre de la aplicación
3. Nombre de los alumnos
4. Barra de navegación (debe poder navegar a las páginas disponibles dependiendo en dónde se esté).

A continuación se detallan las pantallas principales de la aplicación que deben poder ser vistas por un cliente / administrador.
Nota: no se detallan todas las funcionalidades, **sinó las principales**.
Nota: si se nombra como pantalla, **debe ser una ruta distinta, no un modal**.

Cliente:

1. **Pantalla de bienvenida.** En esta pantalla el cliente podrá ingresar su **nombre** antes de poder continuar a ver los productos.
2. **Pantalla de productos.** En esta pantalla se deben poder visualizar los productos de la aplicación, divididos en categorías (dos categorías como fue mencionado anteriormente), los productos deben mostrar todos sus datos e imagen, a la vez que la capacidad para agregarlos o quitarlos del **carrito.**
3. **Pantalla de carrito.** En esta se debe ver el listado de productos que ya fueron añadidos al carrito, a su vez se debe poder agregar o quitar distintas cantidades de los productos seleccionados.
4. **Pantalla de ticket.** Una vez confirmado el carrito, se debe poder ver un ticket con los productos comprados, sus datos y el nombre del usuario.

Administrador:

1. **Pantalla login.** Debe poder permitir ingresar un **correo** y **contraseña** para acceder.
2. **Pantalla dashboard.** Debe poseer el listado de productos, así como las acciones correspondientes para el alta, baja y modificación de los mismos.

3. **Pantalla Alta de producto.** Debe permitir cargar un nuevo producto con todos sus datos y su imagen.
4. **Pantalla Modificar producto.** Debe permitir modificar todos los datos de un producto a través de su ID. Nota: se puede reutilizar la pantalla de alta.

---

3. Flujos de la aplicación.

A continuación se detallan las acciones principales que debe poder realizar un cliente / administrador. El resto de acciones se detallan en el cuarto apartado.

Flujo principal del cliente:

1. El cliente entra al sitio.
2. Se muestra la **pantalla de bienvenida**.
3. El cliente ingresa su nombre. (solo nombre).
4. El cliente toca el **botón continuar.**
5. Se redirige a la pantalla de productos.
6. El cliente puede navegar entre las dos categorías.
7. El cliente agrega un producto al carrito.
8. El cliente agrega varios productos al carrito al mismo tiempo.
9. El cliente elimina un producto del carrito.
10. El cliente accede a la pantalla del carrito.
11. El cliente agrega más cantidad de un producto ya agregado al carrito.
12. El cliente elimina una cantidad de un producto ya agregado.
13. El cliente finaliza la compra. El sistema redirige a la pantalla del ticket.
14. El ticket muestra los datos de los productos comprados, el nombre ingresado en el paso 2, la fecha de hoy y el nombre de la empresa.
15. El cliente le da al **botón de salir**.
16. El sistema vuelve a comenzar.

Flujo principal del administrador:

1. El administrador entra al sitio.
2. Se muestra la pantalla de login.
3. El administrador ingresa su **usuario y contraseña.**
4. El administrador presiona el **botón ingresar.**
5. Los datos son correctos, se redirige a la pantalla de dashboard. Se muestra el listado de productos completo.
6. El administrador presiona el botón agregar producto.
7. Se redirige a la pantalla de alta del producto. Se muestra el formulario de alta de producto.
8. El administrador completa el formulario. El administrador carga una imagen para el producto.
9. El administrador presiona el botón agregar.

10. Se redirige a la pantalla de dashboard. Se muestra el listado de productos, incluyendo el producto agregado.
11. El administrador elige un producto **activo para eliminar.** Presiona el botón eliminar de ese producto.
12. Se muestra un **modal** que pregunta si está seguro de eliminar. El administrador presiona el botón confirmar.
13. El producto eliminado pasa a mostrarse en estado **inactivo (baja lógica).**

Nota: En este momento, no debería poder verse dicho producto del lado del cliente.

14. El administrador elige un producto para editar. Presiona el botón editar de ese producto.
15. Se redirige a la **pantalla de editar** producto. Se muestra el producto seleccionado.
16. El administrador modifica todos los datos disponibles. El administrador carga una **nueva imagen** para el producto.
17. El administrador presiona el **botón guardar cambios**.
18. Se redirige a la pantalla dashboard. Se muestra el listado de productos, incluyendo el producto modificado.
19. El administrador elige un producto inactivo **para activarlo.** Presiona el botón activar de dicho producto.
20. Se muestra un modal que pregunta si está seguro de activar. El administrador presiona el botón confirmar.
21. El producto activado pasa a mostrarse en estado activo.

Nota: En este momento, debería poder verse dicho producto del lado del cliente.

---

4. Requerimientos aplicación Front-end para el cliente / usuario.

- El sistema debe preguntar su nombre al usuario al comenzar, sólo cuando se complete, se permitirá ver los productos para comprar.
- El sistema debe contar con un nombre de empresa / imagen de empresa. Se pide que las aplicaciones front-end cuenten con un favicon.
- El sistema debe tener cargados y mostrados productos variados para ambos tipos de productos al momento de ser probada por el cliente (los profesores evaluando). Los productos mostrados deben ser los productos que estén activos en ese momento (el producto debe tener la propiedad booleana activo).

  Nota: En caso de no tener el backend armado, los productos deben mostrarse tomados desde un JSON hardcodeado. Deben estar pensados para 1- mostrarse de forma responsive. 2- mostrarse de forma paginada.

- El sistema debe poder persistir en la base de datos el registro de las ventas realizadas exitosamente, junto con **el nombre del usuario que las efectuó, la fecha y el precio total.**
- El sistema debe poder mostrar el ticket de la compra una vez efectuada.
- El sistema debe poder permitir eliminar productos que no se deseen comprar.
- El sistema debe permitir comprar varios de un mismo producto (parámetro cantidad).

- El sistema debe lanzar un modal que pregunte si el cliente quiere confirmar la compra.
- El sistema debe mostrar un botón que permita volver a iniciar el proceso de compra al finalizar y mostrar el ticket.

  Nota: Recordar que una vez finalizada una compra y entregado el ticket, el sistema se reinicia. Esto es un autoservicio, NO un e-commerce.

- El sistema debe permitir descargar el ticket en PDF.
- El sistema debe poder paginar los productos para evitar que estén todos en pantalla al mismo tiempo.
- El sistema debe permitir cambiar el tema de la aplicación (al menos dos, claro y oscuro). Además de mantener el tema elegido si se recarga la página.
- El sistema debe poder tener un botón que nos redirija al login del panel del administrador.
- Al momento de testear la aplicación, debe haber suficientes productos activos para poder ver todas las funcionalidades.

---

5. Requerimientos aplicación Back-end.

- Configurar un servidor en Node.js utilizando Express para manejar las solicitudes y respuestas del sitio web front-end.
- **La devolución de mensajes debe ser mixta, basada en plantillas HTML (EJS) y en JSON.**

5.1. Requerimientos para el back-end: Vistas HTML.

- El sistema debe mostrar una pantalla login al momento de acceder.
- El login debe funcionar obteniendo los datos del usuario de la base de datos.
- El login debe poseer un **botón de acceso rápido**. Dicho botón debe **autocompletar** los input del login (correo y contraseña) para facilitar el acceso a esta sección a los testers (los profesores).

Nota: el uso de botones de acceso rápido es una feature muy importante para que los clientes puedan probar nuestras aplicaciones de forma rápida y eficaz. Logra que en un ambiente de testing, los tiempos en acciones sencillas y repetitivas se recorten para hacer foco en otras funcionalidades.

- Una vez se realiza el login, se redirige a la página que contendrá el panel de administrador.
- El panel de administrador debe mostrar los productos disponibles con sus detalles, separados por tipo.
- El panel debe poder permitir seleccionar un producto y:
  - Modificar el producto (nombre, precio, imagen).
  - Desactivar el producto (baja lógica → cambiar el valor de activo a false).
  - Reactivar el producto (cambiar el valor de activo a true).
- El panel debe poder permitir agregar un nuevo producto a la base de datos. (activo por defecto).

5.2 Requerimientos para el back-end: API en JSON.

- Todas las peticiones deben responder en formato JSON válido.
- Debe poder procesar **todas** las peticiones de **alta, baja, listados y modificación** de la base de datos que sean necesarias para la correcta funcionalidad de la aplicación (tablas productos, usuarios, ventas y cualquier otra que necesiten). Utilizar un ORM.
- Las **rutas** y los **endpoints** deben estar bien estructurados y de forma lógica (MVC).
- Debe permitir la carga de imágenes al dar el alta de un producto. Las imágenes se guardan **en el servidor**.
- La tabla productos y la tabla ventas deben poseer una relación mucho a muchos.
- La creación de un usuario administrador se realiza a través de un endpoint de la API.
- El guardado de las contraseñas de los usuarios debe estar **encriptado**. **NO** se guardan contraseñas como vienen, se cifran antes de guardarlas.
- Debe permitir traer los productos en forma de páginas (paginación).
- Debe validar **todos** los datos que recibe a través de middlewares.
- Debe poder devolver el listado de ventas junto con los productos asociados a las mismas.

Requerimientos extra por entrega tardía

Requerimientos extra - Pasado el segundo parcial

Para el cliente:

- Se debe crear la pantalla detalle, que debe permitirle al cliente ver los detalles de un producto. Se debe tomar el id del producto desde los parámetros de la ruta para traer los datos.

Para el administrador:

- Se debe crear la pantalla de registros.
- El administrador debe poder ver los registros de los 10 productos más vendidos.
- El administrador debe poder ver los registros de las 10 ventas más caras.
- El sistema debe guardar registro a modo de LOG de cada vez que un usuario administrador inicia sesión.
- El LOG de inicios de sesión debe poder ser visto en la pantalla de registros.
- Se deben mostrar dos estadísticas más sobre las ventas / productos / logs en tablas.
- El administrador debe poder descargar los datos de los registros en Excel.

Requerimientos extra para fecha de final

Para el cliente:

- Luego de ver el ticket y tocar el botón para salir, se redirige a la pantalla encuesta.
- La pantalla encuesta debe permitir ingresar la opinión del consumidor. Debe contener al menos 5 tipos distintos de input: textarea, email, checkbox, slider y file.
- Todos los datos deben estar validados. Se deben mostrar los mensajes correspondientes en caso de error.
- El input de tipo file debe ser una imagen que se guardará en el servidor.
- El input de tipo slider debe ser la puntuación que se le otorgó al servicio.
- Se debe permitir omitir la encuesta. El botón omitir debe ser visible pero no resaltar más que el resto de acciones en la pantalla.
- Se debe mostrar un mensaje de agradecimiento en forma de modal al finalizar la encuesta correctamente. Se debe guardar la información en la BD junto con la fecha actual.

Para el administrador:

- Se debe crear la pantalla asistencia, la cuál debe permitir ver los datos de las encuestas de ayuda.
- Se deben agregar dos tablas más a la pantalla de registros, estos deben mostrar datos relevantes sobre las encuestas / asistencias solicitadas. Además, se debe poder filtrar entre dos fechas ingresadas, según la fecha de creación de los registros.