Trabajo Práctico - Sistema Comedor
2º Cuatrimestre 2025

Integrantes:

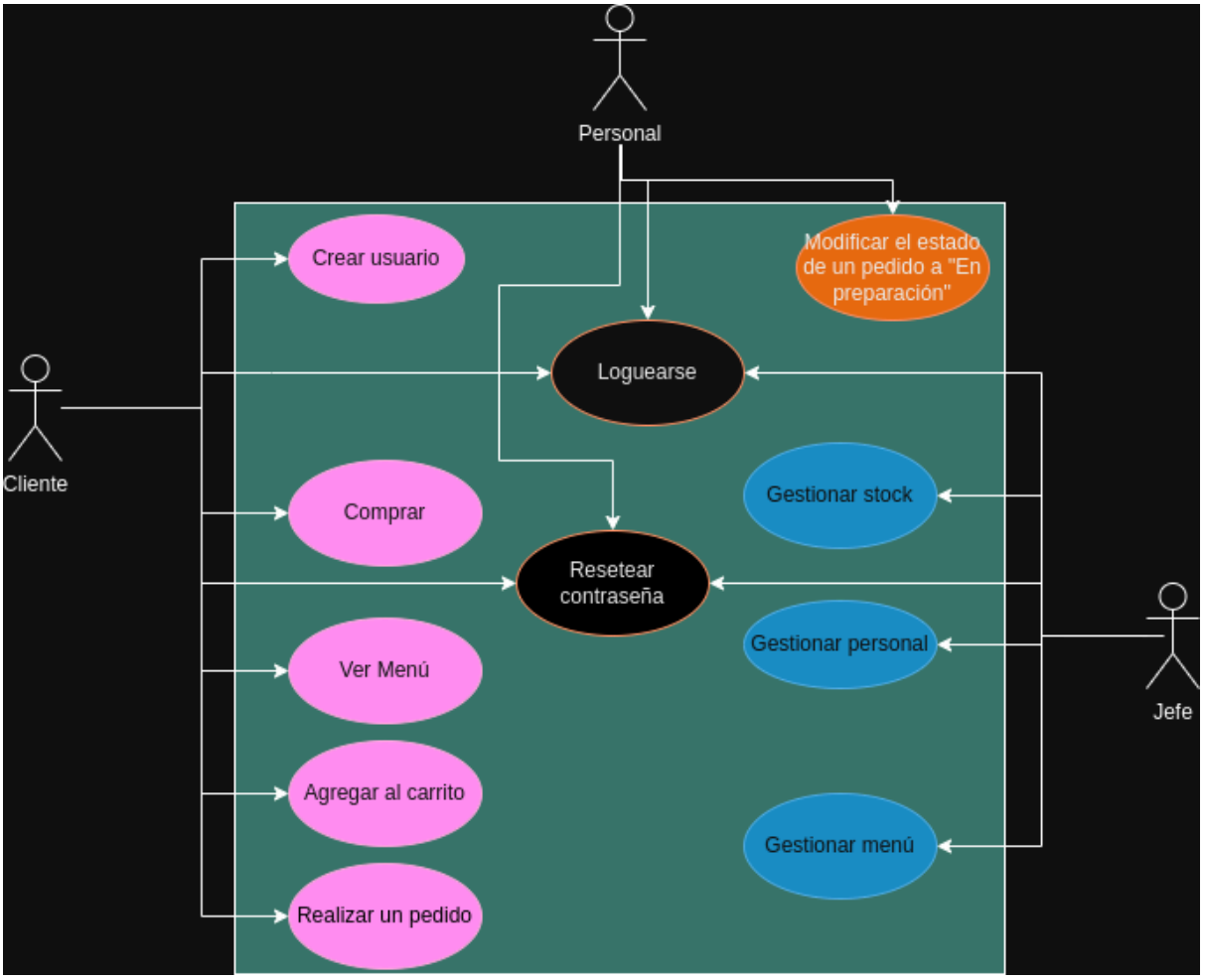
- Calderón Vasil, Máximo Augusto
- Cuello, Milagros
- Molina, Steven
- Moore, Juan Ignacio
- Rehl, Juana
- Tripaldi, Ulises

📁 PROYECTO → carpeta donde se organizó toda la documentación de las sprints y el trabajo en general, incluyendo retros, observaciones de QA, user stories, etc.

Sprint 1

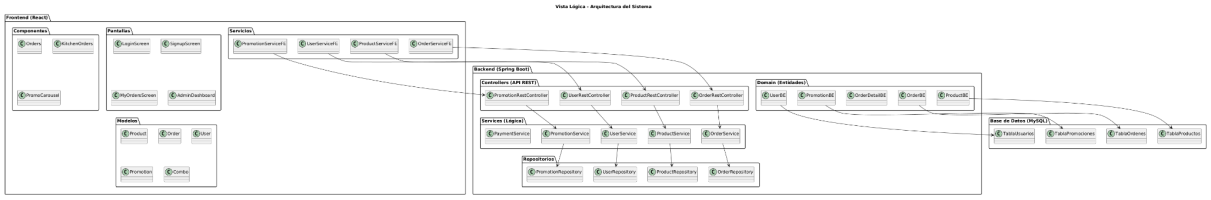
En primer lugar se estableció el alcance del proyecto, los modelos de vistas de la arquitectura 4+1, los atributos de calidad y el product backlog para tener en claro las tareas necesarias para cumplir con el trabajo práctico. Luego de que el Scrum Master guíe el planning poker, decidimos que el objetivo principal de este primer sprint era hacer todo lo que tenía que ver con el menú y cómo lo gestionaba el administrador del sistema. Se logró que se visualizara desde la vista del usuario, se pueda crear productos, modificarlos y eliminarlos.

Vista de escenarios:

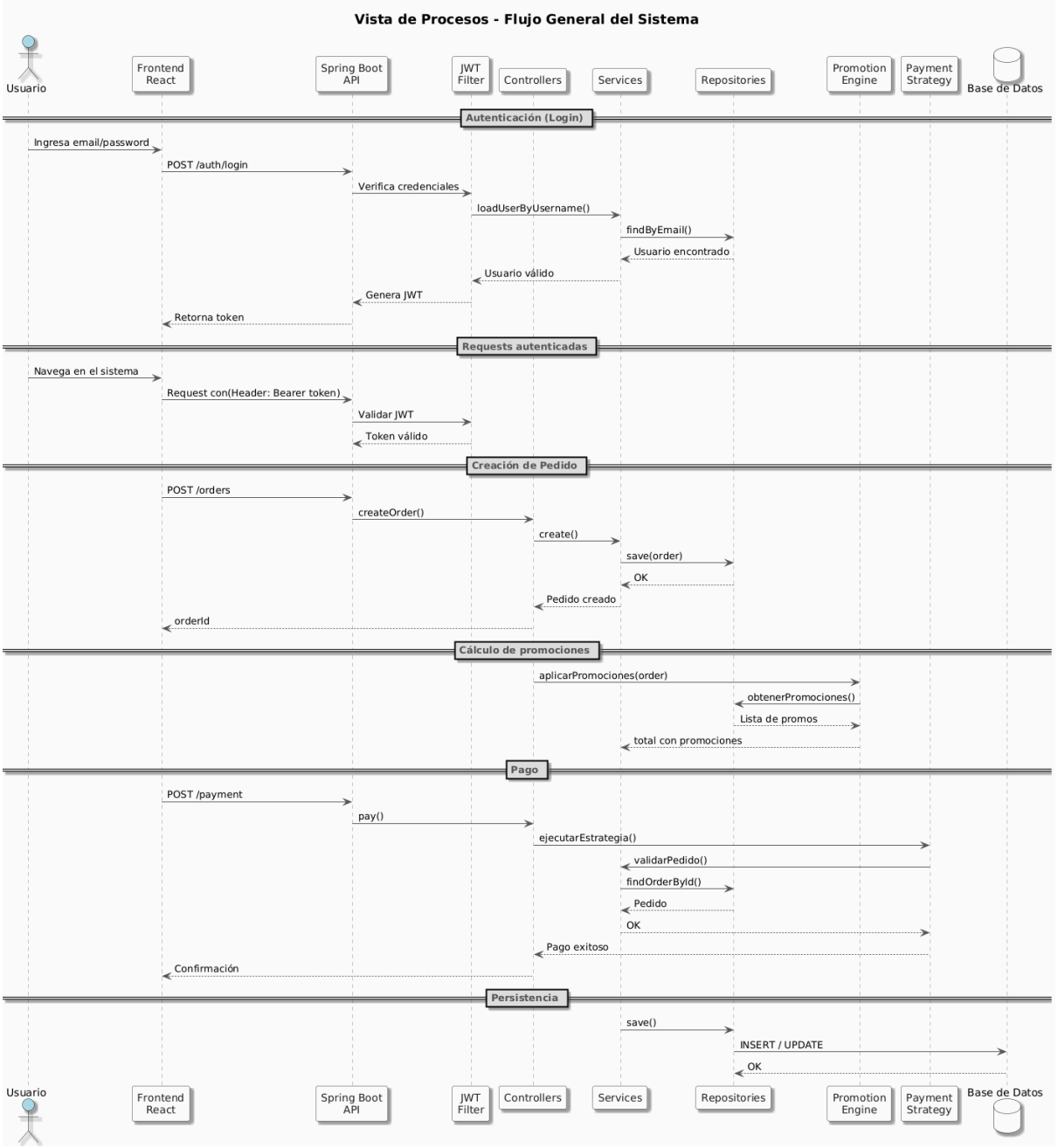


PROYECTO → carpeta donde se organizó toda la documentación de las sprints y el trabajo en general, incluyendo retros, observaciones de QA, user stories, etc.

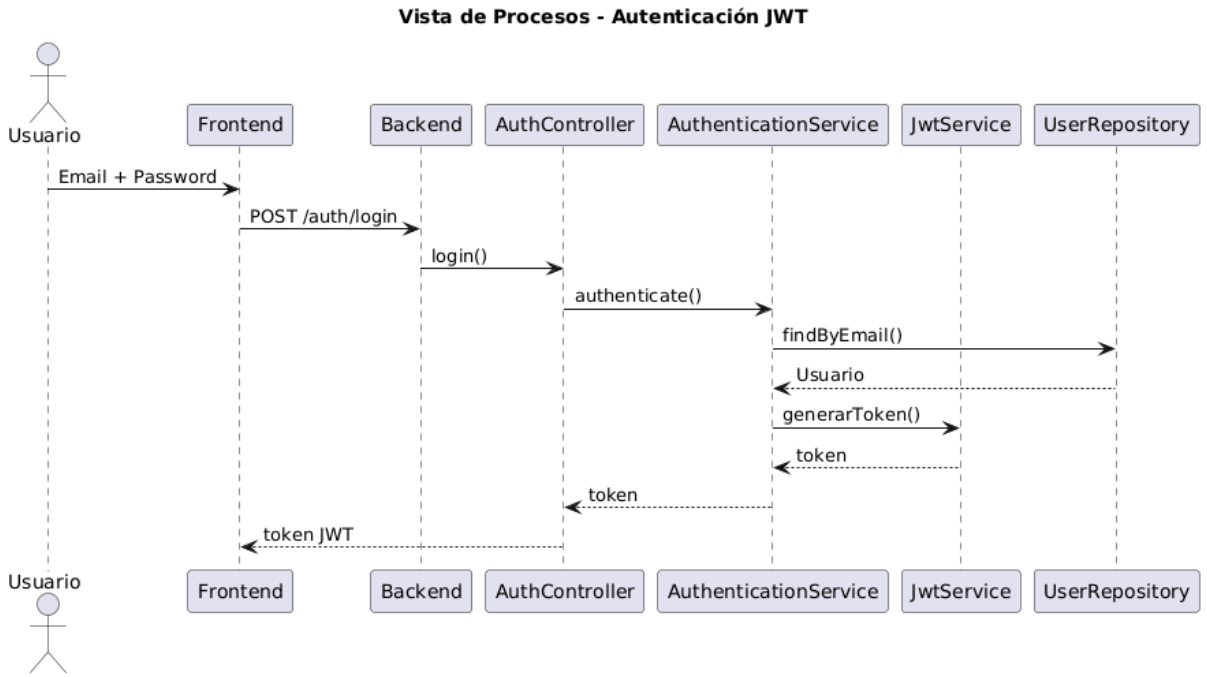
Vista lógica:



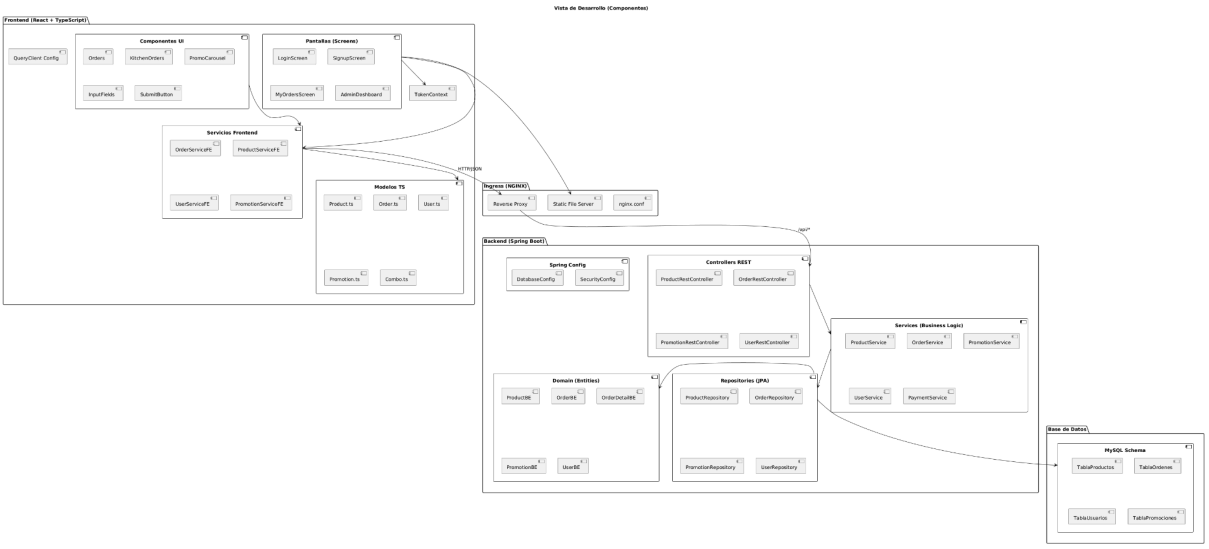
Vista de procesos:



PROYECTO → carpeta donde se organizó toda la documentación de las sprints y el trabajo en general, incluyendo retros, observaciones de QA, user stories, etc.

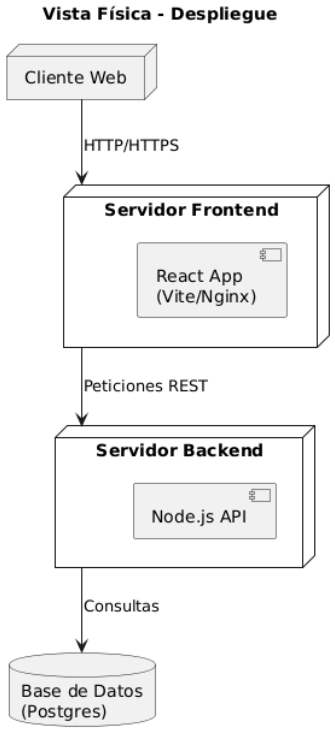


Vista de desarrollo:



PROYECTO → carpeta donde se organizó toda la documentación de las sprints y el trabajo en general, incluyendo retros, observaciones de QA, user stories, etc.

Vista física:



Atributos de calidad:

Seguridad

- Autenticación: el sistema deberá validar la identidad de todos los usuarios (ya sean estudiantes, profesores, personal de staff) antes de permitir el acceso a la aplicación. Esto se hará mediante un código vía email.
- Autorización: un usuario sólo podrá consultar el menú, realizar pedidos, y ver sus pedidos. Un personal de staff sólo podrá cambiar el estado de las órdenes.
- Encriptación de datos: información como las contraseñas de los usuarios deberán estar cifradas. No debe haber fugas de información personal.
- Bloqueo temporal de cuentas después de múltiples intentos fallidos de inicio de sesión.

Usabilidad

- Fácil extensibilidad (pagos): agregar un nuevo método de pago no deberá requerir modificar la lógica de pedidos, sólo añadir un nuevo componente.

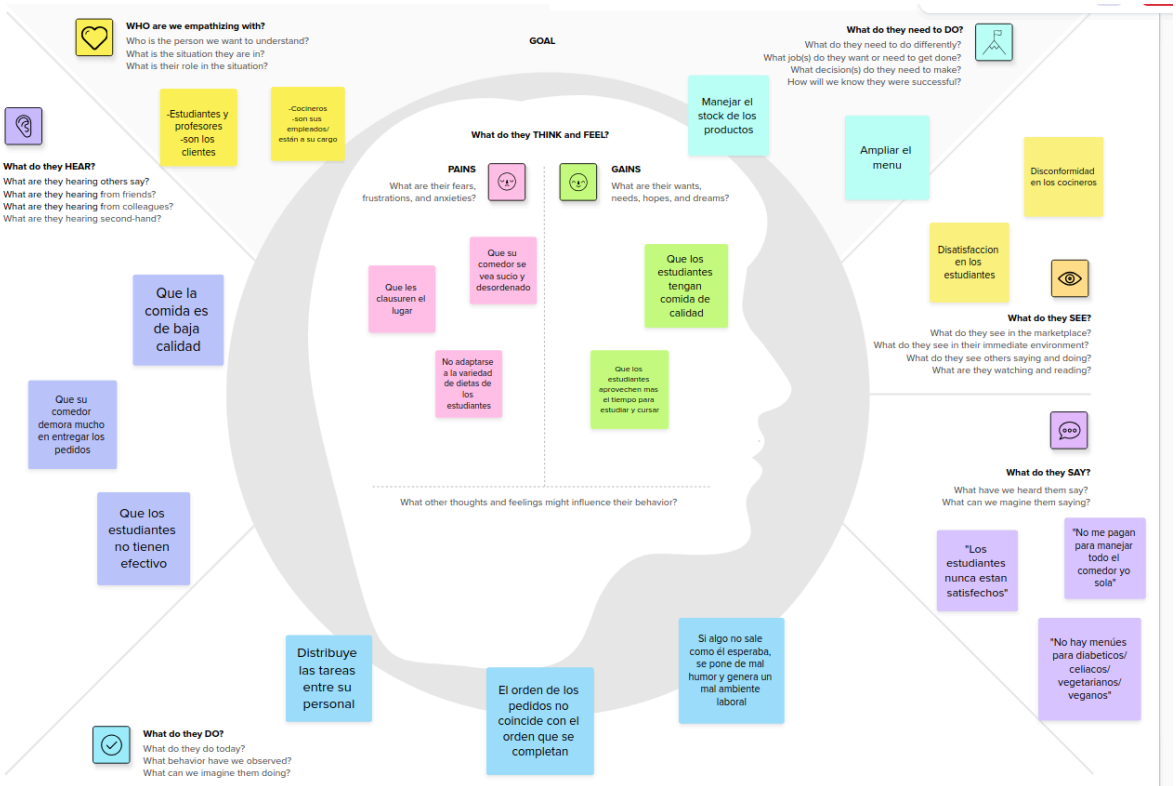
PROYECTO → carpeta donde se organizó toda la documentación de las sprints y el trabajo en general, incluyendo retros, observaciones de QA, user stories, etc.

- La aplicación alcanzará un grado alto de satisfacción en facilidad de uso gracias a los mensajes claros y colores llamativos. El flujo de pedido (selección → carrito → pago) deberá ser simple y requerir mínimos pasos.

Performance

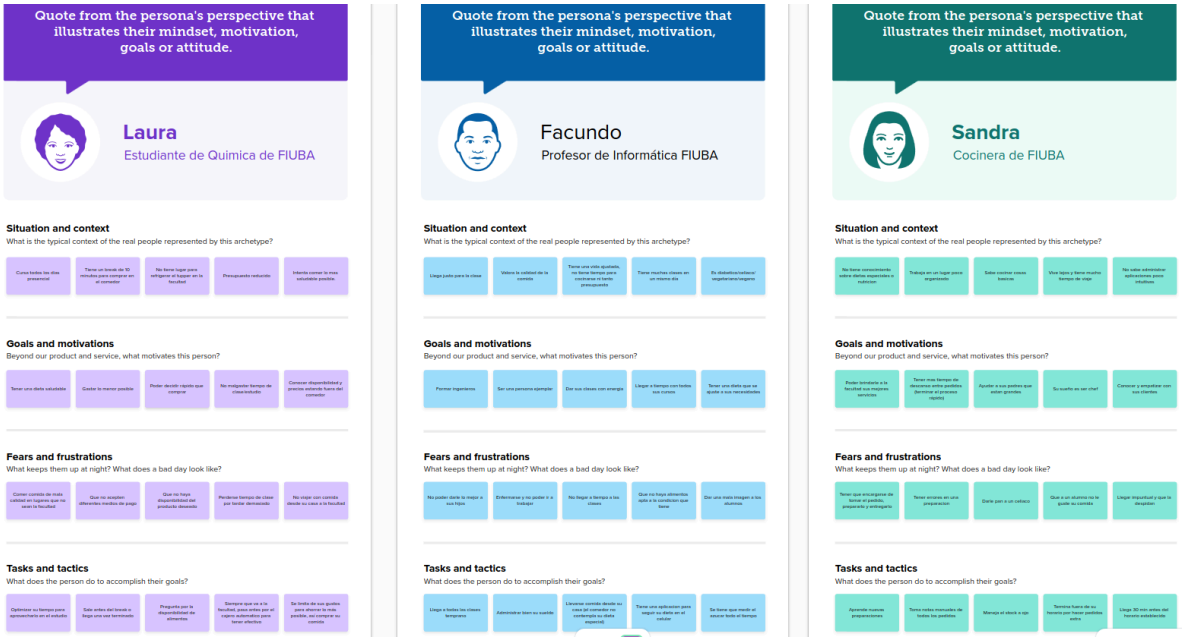
- El tiempo de respuesta para crear una orden, que implica validar ingredientes y aplicar descuentos, deberá ser mínimo.
- La consulta del menú deberá ser muy rápida, ya que es la operación más frecuente.
- Las consultas a la base de datos para validar stock y obtener menús deberán ser eficientes.

Mapa de empatía:



PROYECTO → carpeta donde se organizó toda la documentación de las sprints y el trabajo en general, incluyendo retros, observaciones de QA, user stories, etc.

User Persona:



Alcance del proyecto:

1. App web con interfaz diferencial para clientes, personal del comedor y persona a cargo
2. Manejo de roles para tales usuarios (cliente, jefe comedor, demás personal del comedor)
3. Manejo de Stock (por parte del jefe) y actualización en tiempo real
4. Actualización de promociones dinámicamente (atado al stock)
5. Visualización del menú
6. Gestión del menú/productos (por parte del jefe)
7. Realización de pedidos y gestión interna
8. Cancelación de pedidos que no hayan comenzado a prepararse
9. Autenticación estudiantes (JWT) - Restablecimiento de credenciales - Actualización de contraseña

PROYECTO → carpeta donde se organizó toda la documentación de las sprints y el trabajo en general, incluyendo retros, observaciones de QA, user stories, etc.

Product Backlog:

1. Armar vistas (html, css)
    a. Vista de Usuario
        i. Vista de Inicio de Sesión
        ii. Vista de Reset de Contraseña
        iii. Vista de Cambio de Contraseña
        iv. Vista de menú (Productos, Combos, Promociones)
        v. Vista de Reseñas
    b. Vista de Jefe
        i. Vista de Inicio de Sesión
        ii. Vista de Asignar/Desasignar rol de Personal de Comedor
        iii. Vista de manejo de Stock
        iv. Vista de manejo de Productos
        v. Vista de manejo de combos
        vi. Vista de manejo de Promociones
    c. Vista de Personal de Comedor
        i. Vista de Inicio de Sesión
        ii. Vista de actualización de estado de pedidos
2. Creación de tablas para: Menú, Productos, Ingredientes, Promociones, Usuarios, Pedidos.

3. Crear Funciones para manejar las tablas (BACKEND)
    a. Manejo de usuarios
        i. Sign up
        ii. Sign in
        iii. Restablecer contraseña
        iv. Actualizar contraseña
    b. Manejo de Ingredientes y Productos
        i. Agregar Stock (jefe)
        ii. Consumir Stock (No jefe)
    c. Manejo de Promociones
        i. Crear/Eliminar/Editar (jefe)
    d. Manejo de Pedidos
        i. Estados:
            1. Iniciado: Apenas el cliente lo pide
            2. Confirmado: Una vez pagado
            3. En preparación: (No jefe) se lo asigna (**YA NO** puede ser cancelado)
            4. Listo para retirar: Ya se preparó y el cliente lo tiene que retirar
            5. Completado: Una vez retirado
        ii. Chequeo de Stock antes de confirmar el pedido, una vez confirmado decrementar el mismo
4. Autenticación de Usuarios con JWT

PROYECTO → carpeta donde se organizó toda la documentación de las sprints y el trabajo en general, incluyendo retros, observaciones de QA, user stories, etc.

User Stories:

Descripción

Como Laura (Estudiante Universitaria de Química),

Quiero visualizar el menú disponible,

Para para saber qué opciones de almuerzo tengo con tiempo

Criterios de Aceptación

- El usuario debe poder ver una lista con los ítems del menú (nombre, descripción, precio).
- La información proviene de la tabla de menú en la base de datos.
- El menú debe ser accesible públicamente.

Descripción

Como Diego (Jefe del comedor),

Quiero poder crear y modificar ítems del menú,

Para mantener la información actualizada para mis clientes

Criterios de Aceptación

- El usuario puede obtener el rol de jefe mediante un código especial.
- El usuario puede agregar un ítem de menú indicando nombre, descripción y precio.
- El usuario puede modificar ítems existentes.
- Los cambios deberán verse reflejados en la vista del cliente inmediatamente.

PROYECTO → carpeta donde se organizó toda la documentación de las sprints y el trabajo en general, incluyendo retros, observaciones de QA, user stories, etc.

Descripción

Como Diego (Jefe del comedor),

Quiero poder eliminar ítems del menú,

Para que mis clientes no vean opciones que ya no están disponibles

Criterios de Aceptación

- El usuario deberá poder eliminar items del menú.
- Una vez eliminado, el item no se deberá mostrar más en la vista del menú para los clientes.

Retro:

Estado: Finalizada.

Dificultades:

Durante el desarrollo del módulo de Menú Online, una de las principales dificultades fue la interacción entre múltiples tablas de la base de datos, ya que fue necesario relacionar correctamente la información de los ítems del menú, las categorías y las configuraciones asociadas. Esto implicó ajustar consultas y estructuras para asegurar la coherencia de los datos y un correcto funcionamiento del backend.

Todas las tareas planificadas fueron finalizadas exitosamente, alcanzando el 100 % de cumplimiento del sprint según el tablero de Jira.
El burndown chart no refleja los puntos de historia por haberse cargado luego del inicio, pero se registraron correctamente al cierre.

Conclusión:

El Sprint 1 fue exitoso: el equipo entregó una versión funcional del menú online con las operaciones principales (consultar, crear, editar y eliminar ítems).
El flujo de trabajo en Jira fue claro, con buena comunicación entre los roles y compromiso general del equipo.

📁 PROYECTO → carpeta donde se organizó toda la documentación de las sprints y el trabajo en general, incluyendo retros, observaciones de QA, user stories, etc.

Sprint 2

Luego del planning poker, en este sprint se decidió hacer todo lo que tiene que ver con registro de usuario e inicio de sesión. Esto cumplió los requisitos como verificar el registro con un código enviado por mail, o recuperar la contraseña en caso de que el usuario la olvide. Además, se agregó la lógica de los combos compuesto por productos individuales y el carrito para que el cliente pueda hacer su orden (solo visual).

User Stories:

User Story 1:

Como Laura (Estudiante Universitaria de Química), **quiero** registrarme en el sitio web del comedor de FIUBA, **para** poder ver la disponibilidad del menú y los precios estando en una clase

Acceptance Criteria:

El registro debe pedir:

- nombre, apellido, email, foto (archivo), edad, género, domicilio, contraseña
- El email deberá ser validado:
  - Para validarlo el usuario debe ingresar un código que se le envía al mail
  - El registro no debe poder completarse hasta que el código no sea válido
  - Solo podrá haber un único usuario por mail

User Story 2:

Como Facundo (Profesor de Informática FIUBA), **quiero** iniciar sesión en la web del comedor, **para** poder ver la disponibilidad del menú

Acceptance Criteria:

El inicio de sesión (query: signin) debe pedir:

- Mail (debe haberse registrado previamente)
- Contraseña
- En caso de varios ingresos erróneos (5) para un mail dado, se bloquea y se envia mail al mail registrado para restablecer la contraseña

📁 PROYECTO → carpeta donde se organizó toda la documentación de las sprints y el trabajo en general, incluyendo retros, observaciones de QA, user stories, etc.

User Story 3:

Como Sergio (Profesor de Informática FIUBA), **quiero** poder reestablecer mi contraseña, **para** poder acceder al sistema en caso de olvidarla,

Criterios de Aceptación

- En la vista del login debe haber un botón que indique “olvidé mi contraseña”
- Haciendo click sobre el mismo:
    - Se debe pedir el mail al usuario y en caso de haber un usuario registrado con ese mail:
        - Se debe enviar un link al mail donde le permita restablecer la contraseña (query: change_password)

User Story 4:

Como jefe del comedor, quiero **crear combos** que agrupen ítems con precio especial para ofrecer promociones simples.

Acceptance Criteria:

- Un combo tiene varios ítems del menú.
- La disponibilidad depende del stock de todos los ítems.
- El precio total es configurable.

PROYECTO → carpeta donde se organizó toda la documentación de las sprints y el trabajo en general, incluyendo retros, observaciones de QA, user stories, etc.

Épica Carrito

- Meta: que el usuario pueda crear un carrito y agregar productos a él
  Como cliente del comedor, quiero agregar productos y combos a un carrito de compras para poder revisar y confirmar mi pedido antes de realizar la compra.
- Criterios de aceptación:
  - El carrito puede contener tanto productos individuales del menú como combos.
  - Se pueden agregar múltiples unidades de un mismo producto o combo.
  - El total del carrito se actualiza automáticamente al agregar o quitar elementos.
  - Es posible eliminar ítems específicos del carrito.
  - El carrito debe mantenerse al refrescar la página (persistencia local).
  - No se deben permitir cantidades que excedan el stock disponible.
- Subtareas:
  - Permitir agregar productos y combos al carrito
  - Permitir agregar múltiples unidades de un mismo producto o combo
  - Actualizar automáticamente el total del carrito
  - Eliminar ítems específicos del carrito
  - Persistir el carrito localmente al refrescar la página

Retro:

Qué salió bien

Se completaron las historias principales de la épica de Gestión de Usuarios y Autenticación, incluyendo:

- Registro de usuario con validación de email mediante código.
- Inicio de sesión con control de intentos fallidos y bloqueo de cuenta temporal.
- Recuperación de contraseña con envío de link al correo registrado.

Se implementó la estructura base de la funcionalidad de combos, incluyendo el diseño de la tabla y la lógica de creación.

Se avanzó en el carrito de compras, logrando agregar productos y combos, actualizar totales y mantener la persistencia local.

Se logró una buena coordinación entre los miembros del equipo y se mejoró la integración entre el frontend y el backend.

PROYECTO → carpeta donde se organizó toda la documentación de las sprints y el trabajo en general, incluyendo retros, observaciones de QA, user stories, etc.

Qué podría mejorarse

Hubo dificultades para conectar las tablas con la base de datos debido a estados inválidos o migraciones previas mal aplicadas.

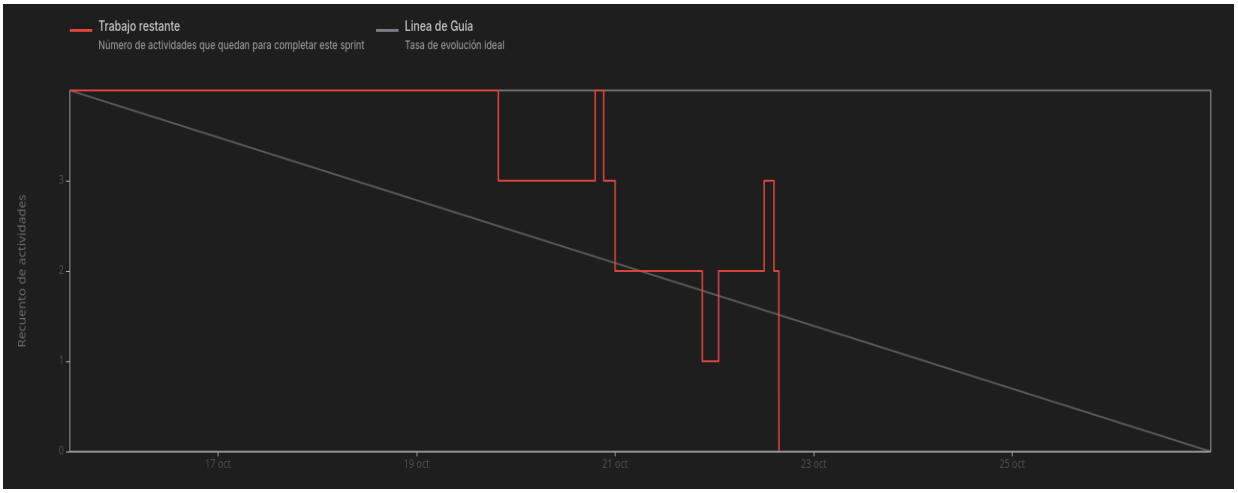
Se presentaron problemas de sincronización entre el frontend y el backend, especialmente al manejar los botones y eventos del inicio de sesión.

La integración entre el registro y el inicio de sesión fue compleja, ya que el sign in dependía directamente del correcto funcionamiento del signup.

Conclusión:

El Sprint 2 fue productivo, ya que se logró implementar la base del sistema de usuarios y se avanzó con las funcionalidades de combos y carrito. A pesar de las dificultades técnicas en la integración y en la base de datos, el equipo logró resolver los principales bloqueos y dejar una estructura sólida para seguir iterando en el Sprint 3.

Burndown Chart:



Podemos ver que el carrito no estaba pronosticado para hacer en este sprint, pero logramos concretar las tareas antes de lo esperado, dejando tiempo para sumar esa funcionalidad.

PROYECTO → carpeta donde se organizó toda la documentación de las sprints y el trabajo en general, incluyendo retros, observaciones de QA, user stories, etc.

Sprint 3

En este sprint se empezó con los pedidos. La idea fue que el usuario pueda crear pedidos a partir de los productos que se muestran en el menú y se agreguen en el carrito implementado en el sprint anterior. Aparece por primera vez el personal del comedor como partícipe del sistema, viendo los pedidos y cambiando el estado de los mismos.

User Stories:

1) **Como** estudiante, **quiero** poder seleccionar ítems del menú y crear un pedido con mis elecciones, **para** poder recibir mi almuerzo preparado.

Criterios de aceptación:

- Se pueden agregar ítems del menú al pedido.
- El pedido se crea en estado “Iniciado”.
- Se calcula el total según los precios de los ítems seleccionados.
- Solo puede hacerse un pedido si los ítems están disponibles.

2) **Como** personal del comedor, **quiero** poder cambiar el estado de los pedidos **para** reflejar el progreso de la preparación y entrega.

Criterios de aceptación:

- Estados válidos:
  Iniciado → Confirmado → En preparación → Finalizado → Entregado.
- Solo el personal de cocina o el jefe del comedor puede modificar estados.
- No se puede saltar estados (transiciones secuenciales).

📁 PROYECTO → carpeta donde se organizó toda la documentación de las sprints y el trabajo en general, incluyendo retros, observaciones de QA, user stories, etc.

3) **Como** cocinera, **quiero** ver los pedidos realizados **para** saber qué me toca cocinar

Criterios de aceptación:

- El usuario (con tipo de rol de cocinero/a) puede ver los pedidos en estado “Confirmado“ para empezar a cocinarlos.
- Cada pedido debe indicar los productos y/o combos que lo forman

4) **Como**: Desarrollador del sistema del comedor **Quiero**: Implementar la lógica de manejo de pedidos en el backend **Para que**: Los pedidos puedan ser creados, validados, calculados y gestionados correctamente

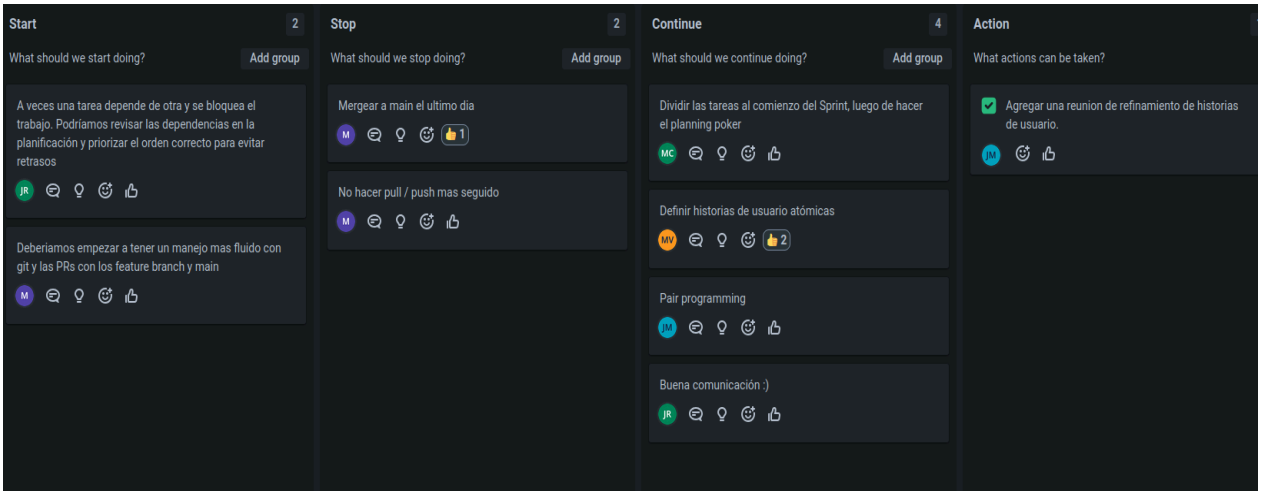
Criterios de aceptación:

- La lógica permite crear un pedido con uno o varios productos/combos, incluyendo cantidades.
- Antes de confirmar el pedido se valida que los platos seleccionados estén disponibles en el menú.
- Se calcula automáticamente el total del pedido sumando los precios de cada producto/combo por la cantidad seleccionada.
- Cada pedido tiene un identificador único y un estado inicial (“Iniciado”).
- La lógica permite actualizar el estado del pedido (Iniciado → Confirmado → En preparación → Finalizado → Entregado.).

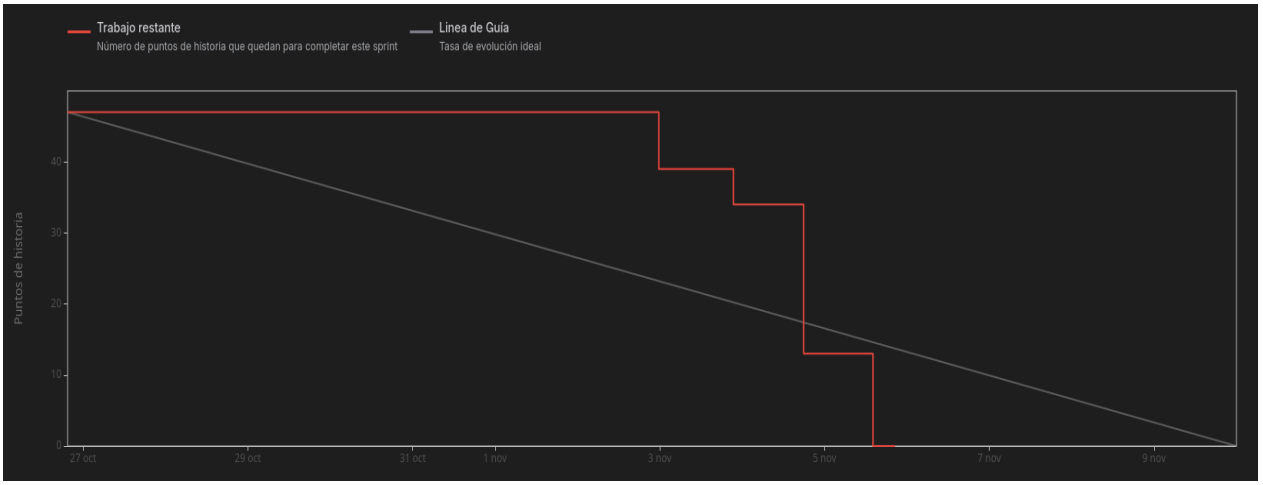
PROYECTO → carpeta donde se organizó toda la documentación de las sprints y el trabajo en general, incluyendo retros, observaciones de QA, user stories, etc.

Retro:

Para este sprint la retro se hizo con la herramienta que proporciona Jira.



Burndown Chart:



PROYECTO → carpeta donde se organizó toda la documentación de las sprints y el trabajo en general, incluyendo retros, observaciones de QA, user stories, etc.

Sprint 4

Este último sprint se centró en promociones y métodos de pagos. Por el lado de las promociones, se logró la gestión, visualización y posibilidad de compra de las mismas. Por el lado de pagos, se hizo que haya distintos métodos (simulados), y que una vez realizado se pueda ver el estado del mismo.

User Stories:

Title: Gestión de Promos	**Priority**	**Estimate**
User Story Como: Diego (Jefe del comedor) Quiero: Gestionar las promociones del comedor Para: Que los clientes tengan opciones económicas y beneficios de sus compras		
Acceptance Criteria: El usuario debe loguearse e internamente se le dará el rol de Jefe para que se le habiliten ciertos accesos privados de la aplicación: - Agregar/quitar/Editar promociones (query: new_promo, delete_promo, edit_promo) - Definir la cantidad de promociones a presentar		

Title: Asignar roles a empleados	**Priority:**	**Estimate:**
User Story Como: Diego (Jefe del comedor) Quiero: Poder definir roles a los empleados Para: Tener un equipo organizado, donde cada uno tiene su propia responsabilidad		
Acceptance Criteria: El usuario, que tiene el rol de jefe, puede otorgar los distintos roles a sus empleados. Así mismo, debería poder quitar roles. Los empleados deberían ver su cambio reflejado en la página, con acceso a distintas secciones privadas del comedor.		

PROYECTO → carpeta donde se organizó toda la documentación de las sprints y el trabajo en general, incluyendo retros, observaciones de QA, user stories, etc.

Title: Gestión de pagos	**Priority:**	**Estimate:**
User Story Como: Laura (Estudiante de FIUBA) Quiero: Tener distintos métodos de pago disponibles Para: Abonar de manera cómoda y según mis responsabilidades económicas		
Acceptance Criteria: Cuando el usuario realiza una compra en el comedor, el sistema debe mostrarle los distintos medios de pago disponibles (por ejemplo: efectivo, tarjeta, transferencia/QR). El usuario debe poder seleccionar el método de pago deseado antes de confirmar su compra. El sistema debe registrar correctamente el método de pago elegido junto con la transacción. Si un método de pago no está disponible temporalmente, el sistema debe indicarlo claramente. El sistema debe permitir agregar nuevos medios de pago a futuro sin afectar las compras existentes. Las opciones de pago deben mostrarse de forma clara y accesible desde la interfaz de pago.		

Title: Seguimiento de pedido	**Priority:**	**Estimate:**
User Story Como: María (Estudiante) Quiero: Que al realizar y pagar mi pedido, pueda ver el estado de mis pedidos Para: Poder organizarme y saber cuándo retirarlo sin tener que consultar al personal del comedor		
Acceptance Criteria: La página tiene una pestaña de “Pedidos en curso” con el estado del pedido. El estado cambia automáticamente (En preparación, Listo para retirar). Al estar listo, se muestra una notificación visual.		

PROYECTO → carpeta donde se organizó toda la documentación de las sprints y el trabajo en general, incluyendo retros, observaciones de QA, user stories, etc.

Retro:

Impedimentos

Formularios por tipo de promoción.

Dificultad de integración UI en componente carousel.

Update de promoción BuyGiveFree.

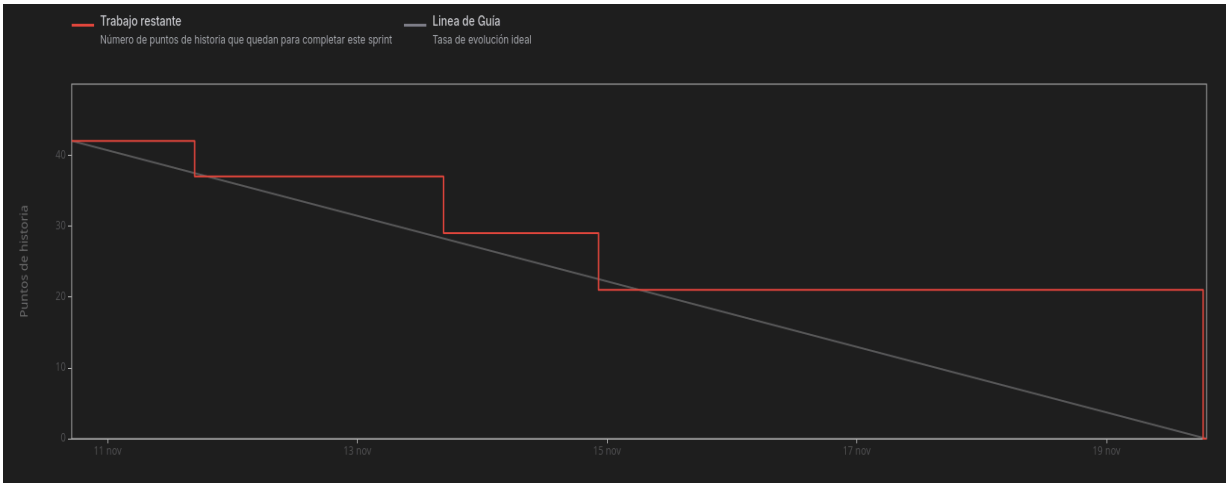
Continue:

- Velocidad de trabajo.
- Trabajo diario.
- Pair programming.
- Preguntar al PO por decisiones que tomaría el cliente.

Stop:

- No se cerraron historias hasta el último momento.

Burndown Chart:



PROYECTO → carpeta donde se organizó toda la documentación de las sprints y el trabajo en general, incluyendo retros, observaciones de QA, user stories, etc.

Código y Patrones

Esta sección tiene como objetivo demostrar las buenas prácticas que conseguimos al iterar en refactors y gracias a patrones vistos en la materia.

Extensibilidad en Promociones:

En la sección de promociones dinámicas optamos por hacer el patrón **Strategy**, respetando el principio de open/closed, dando una mejor calidad al código. El enunciado pedía hacer varios tipos de promociones (2x1, porcentaje de descuento, etc), para esto, hicimos una clase abstracta Promotion, de donde las clases que la implementan definen su propio método **apply()**. En OrderService se hace:

```
List<Promotion> promotions = promotionService.getPromotionsActiveNow();
for (Promotion promo : promotions) {
    promo.apply(order);
}
```

dónde está la buena práctica que quien llame la función no actúe diferente dependiendo la promo que sea, sino que hace apply para cualquier caso, respetando el polimorfismo.

📁 PROYECTO → carpeta donde se organizó toda la documentación de las sprints y el trabajo en general, incluyendo retros, observaciones de QA, user stories, etc.

Desacoplamiento en Pagos:

Se pretendía el soporte para diversos medios de pago (efectivo, QR, tarjeta). Para esto se utilizó el patrón **Factory**, haciendo la clase **PaymentMethodFactory** que recibe un identificador (string) y devuelve la implementación concreta de la estrategia de pago (PaymentMethod). Esto desacopla al controlador de la lógica específica de cada medio de pago.

```
PaymentMethod payment = paymentFactory.get(method);

PaymentResult result = payment.processPayment(order);
```

Funcionalidades de última hora:

De acuerdo a lo solicitado a último momento previo a la entrega del trabajo, se quieren agregar las siguientes reglas de promociones nuevas:

- 2x1 en pizzas después de las 18 hs.
- 10% si tenes mail de fiuba

Analizándolas, podemos apreciar que ya contamos con promociones del tipo 2x1 y del tipo 10% de descuento, BuyXPayYPromotion y PercentageDiscountPromotion respectivamente. Lo único que habría que modificar en el código existente sería agregar dos tipos de verificaciones extra para las promociones y atributos en la clase base Promotions. Estos son, las horas de validez de una promoción y el mail del usuario comprador. Al agregar estos campos, estamos ampliando aún más el abanico de posibilidades de configuración de promociones por parte del administrador del comedor. Y si por ejemplo, en un futuro, se desease que una promoción del tipo ThresholdDiscountPromotion sea aplicada a cierto dominio fi.uba.ar y después de las 14hs, se podría hacer sin ningún inconveniente.