

Ingeniería en sistemas de información

Cátedra Ingeniería y Calidad de Software

*Docentes del curso:* Boiero, Gerardo

Crespo, Mickaela

**TRABAJO PRACTICO N° 6**

**Requerimientos ágiles- Implementación de user stories**

*Curso*: 4K1

*Número de grupo:* 3

*Integrantes:*

* Beato Dominguez, Valentina- 86278
* Brocanelli Novillo, Valentina- 87030
* De los Santos, Martina - 87699
* Firpi Padro, Gonzalo - 85954
* Pacheco, Franco - 82419
* Rojo, Agustín - 85681

*Fecha de entrega:* 12/09/2023

**Análisis:**

| **Realizar un Pedido de "lo que sea"** |
| --- |
| **Descripción:**  Como Solicitante quiero realizar un Pedido de “lo que sea” para recibir algo en mi domicilio que no está disponible en los comercios adheridos |
| **Criterios de aceptación:**  ● Se debe indicar qué es lo que debe buscar el Cadete con un campo de texto  ● Se puede adjuntar opcionalmente una foto en formato JPG con un tamaño máximo de 5 MB.  ● Se debe indicar la dirección del comercio en forma textual (calle, número, ciudad y referencia opcional en formato de texto)  ● Se debe indicar la dirección de entrega (calle, número, ciudad y referencia opcional en formato de texto).  ● Se debe poder seleccionar la ciudad de un listado de ciudades disponibles.  ● Se debe poder visualizar el total a pagar, en función de la distancia y/o del costo del producto, antes de elegir la forma de pago.  ● Se debe seleccionar la forma de pago: Efectivo o Tarjeta de Débito/Crédito.  ● Si paga en efectivo debe indicar el monto con el que va a pagar.  ● Si paga con tarjeta debe ingresar el número de la tarjeta, nombre y apellido del Titular, fecha de vencimiento (MM/AAAA) y CVC.  ● Debe ingresar cuando quiere recibirlo: “Lo antes posible” o una fecha/hora de recepción |
| **Pruebas de aceptación:**  ● Probar realizar un Pedido de “lo que sea” en efectivo “lo antes posible” (pasa).  ● Probar realizar un Pedido de “lo que sea” con tarjeta “lo antes posible” (pasa).  ● Probar realizar un Pedido de “lo que sea” programando la fecha/hora de entrega (pasa).  ● Probar realizar un Pedido de “lo que sea” con una tarjeta inválida (falla).  ● Probar realizar un Pedido de “lo que sea” con una tarjeta de crédito MasterCard (falla).  ● Probar realizar un Pedido de “lo que sea” en efectivo sin indicar el monto a pagar (falla).  ● Probar realizar un Pedido de “lo que sea” programando una fecha/hora de entrega no válida (falla).  ● Probar realizar un Pedido de “lo que sea” sin especificar qué buscar (falla).  ● Probar realizar un Pedido de “lo que sea” adjuntando una foto (pasa).  ● Probar realizar un Pedido de “lo que sea” sin indicar la dirección del comercio (falla). |

Habiendo leído y analizado lo planteado en la User Story, se puede detectar que en su descripción comienza con *“Como solicitante…”*. De allí podemos afirmar que el rol del dominio que recibe el valor de esta historia de usuario es **Solicitante**.

Detectado el rol, se procedió a la elaboración de su tarjeta de rol:

| **Rol de usuario: Solicitante** |
| --- |
| Persona que realiza o programa un pedido desde su casa utilizando la aplicación mobile de DeliverEat. |

**Conversación:**

En la práctica, en el contexto de la metodología SCRUM que estamos utilizando, es fundamental abordar cualquier incertidumbre, consulta o pregunta relacionada con la funcionalidad a implementar a través de la interacción con el Product Owner del proyecto. En este caso específico, hemos designado a los docentes de la cátedra como nuestros representantes en esta función, y a continuación, enumeramos las consultas que hemos planteado y discutido con ellos:

**Cómo calcular el precio si no sabemos qué producto va a comprar el cliente.**

En la aplicación, el usuario debe introducir el monto a pagar en un campo de texto designado. Además, se aplicará un costo adicional de envío, que consiste en una tarifa fija de $500.

**Información sobre las ciudades de envío y si hay una distancia máxima.**

El servicio de envío está disponible en las ciudades de Córdoba y Carlos Paz. No se imponen restricciones en cuanto a la distancia máxima para los envíos; esto significa que se pueden realizar envíos dentro de la misma ciudad o entre estas dos ciudades sin limitaciones de distancia.

**Qué hacer si la ciudad de entrega no coincide con la del comercio.**

En caso de que tanto la dirección de origen como la de destino pertenezcan a la misma ciudad, el sistema procesará el envío sin realizar acciones adicionales o restricciones. Los envíos dentro de la misma ciudad se gestionarán sin complicaciones adicionales.

**Si existe un límite de caracteres al ingresar la descripción del producto.**

Sí, existe un límite en la longitud de los comentarios o descripciones ingresados en la aplicación. El límite establecido es de 240 caracteres.

**Si debemos seguir una paleta de colores específica en el diseño.**

La paleta de colores utilizada en la interfaz de la aplicación se compone de los siguientes colores:



**Si las imágenes cargadas por los usuarios deben mostrarse como miniaturas.**

Sí, la aplicación mostrará una miniatura de la foto relacionada con los elementos ingresados. Esta característica proporciona una vista previa visual de la foto, lo que facilita la identificación de elementos específicos en la aplicación.

**Debemos incorporar un logo del negocio en nuestro desarrollo de la aplicación.**

Por el momento, DeliverEat no ha desarrollado ningún logo. Por lo tanto, se solicita al equipo de desarrollo que agregue un logo a modo de marcador de posición (placeholder) en la ubicación donde debería ir el logo finalizado de DeliveryEat en la aplicación. Tomamos la libertad de crear uno para marcar el espacio:



**¿Se puede pedir más de un producto? ¿Hay un máximo de productos por comercio? ¿Para cada producto tendría que ser una descripción distinta o en el mismo cuadro de texto se pueden poner muchas descripciones?**

Se puede pedir la cantidad de productos que entren en la descripción de productos a buscar. Debería haber una sola caja de texto para ingresar todo lo que se desee pedir. Mientras todo se busque por la misma dirección, estará todo bien.

**¿Se puede hacer un pedido de dos o más comercios a la vez? ¿o sólo se puede hacer un pedido por comercio a la vez?**

La dirección de búsqueda debe ser una sola por pedido así que sólo se puede pedir de a un comercio a la vez.

**¿El monto del producto debe escribirse en la misma pantalla que la descripción del mismo?**

El monto de los productos debe ingresarse en otro campo de texto que diga "Costo de productos" que sólo acepte valores numéricos con el formato de moneda. Este formato debe ser el usado en argentina utilizando un punto (.) como separador de miles, una coma (,) como separador de decimales y con dos dígitos decimales.

**Se indica que se puede seleccionar la opción para recibir el producto determinando una fecha/hora de recepción. ¿Hay alguna fecha no laborable que deba considerarse? ¿Alguna limitación horaria? Para la opción "lo antes posible", alguna limitación?**

El negocio opera de 7:00 a 23:59 todos los días del año, sin excepción, incluyendo feriados y fines de semana.

**En la descripción del funcionamiento de DeliverEat! en el tp2 se indica: "Forma de pago(actualmente solo disponemos en efectivo y tarjeta VISA pero estamos investigando la integración con otras tarjetas de crédito). Sin embargo, en nuestra user hay una prueba que dice "Probar realizar un pedido de 'lo que sea' con una tarjeta de crédito MasterCard(pasa)". ¿Cuáles serían las tarjetas que se aceptan?**

Por ahora el negocio solo acepta tarjetas de crédito VISA y efectivo.

**¿En alguna parte debemos colocar una imágen de Google maps o semejante a un mapa que muestre la dirección?**

Efectivamente deben tener el mapa para la selección de la dirección en donde tienen el ingreso de dirección. En clase dijimos que podían estar mockeados, pero deben agregar la imagen del mapa en reemplazo del mapa real.

**Desarrollo:**

Para la implementación de la funcionalidad detallada en la User Story, se desarrolló una aplicación .NET utilizando el lenguaje C# simulando una aplicación mobile.

De manera complementaria, aunque no se encontraba incluído en la User Story implementada, se agregó una pantalla de inicio con un botón que permite luego realizar un pedido de lo que sea.

En nuestro proyecto de desarrollo de aplicaciones web en C#, hemos adoptado el uso del estilo de nomenclatura conocido como "camelcase" para nombrar variables, parámetros y campos. En este estilo de escritura, la primera letra de la primera palabra se inicia en minúscula, y las primeras letras de las palabras subsiguientes se inician en mayúscula.

Esta elección se basa en las convenciones de nomenclatura comunes en el mundo de la programación en C#. El uso consistente del "camelcase" en nuestro código ayuda a mejorar la legibilidad y coherencia de nuestro proyecto, facilitando la comprensión tanto para los miembros del equipo de desarrollo como para terceros que puedan revisar o colaborar en el código.

Además, es importante destacar que hemos optado por no utilizar el estilo de nomenclatura "Pascal case" en ningún lugar de nuestro código. Esta decisión nos permite mantener una estructura de código clara y coherente, lo que contribuye a un proceso de desarrollo más eficiente y mantenible.

**Convenciones de nomenclatura:**

1. Use camelCase para nombres de tipos, espacios de nombres y miembros públicos.
2. Use nombres descriptivos y evite nombres de una sola letra, excepto para contadores de bucles simples.
3. Prefiere la claridad sobre la brevedad.
4. Use camelCase para campos privados y variables locales.
5. Use camelCase para constantes.
6. Evite abreviaturas a menos que sean ampliamente conocidas y aceptadas en registros posicionales, use camelCase para los parámetros.
7. Use camelCase para campos privados e internos.
8. Use camelCase para los parámetros de método.
9. Asigne nombres descriptivos a los parámetros de tipo genérico, a menos que un nombre de una sola letra sea claro.
10. Considere indicar restricciones en el nombre de parámetros de tipo cuando sea relevante.

**Indentación y Espaciado**

1. Utilizaremos 4 espacios para la indentación en lugar de tabulaciones.
2. Dejaremos 1 espacio antes y después de los operadores binarios (por ejemplo, +, -, =, ==, !=, etc.).
3. Dejaremos 1 espacio después de las comas en listas y argumentos de método.
4. Dejaremos 1 espacio después de los dos puntos en declaraciones de casos switch.
5. No dejaremos espacios en blanco al final de las líneas.
6. Las llaves se colocarán en la misma línea que la declaración, sin espacios adicionales.

**Longitud de Línea**

Limitaremos 40-80 caracteres por línea para mantener líneas de código legibles.

**Comentarios**

Utilizaremos comentarios de una línea con // para comentarios en línea.

Ejemplo: // Esta clase representa un objeto de cliente en la aplicación

**Organización de Código**

Organizaremos el código de manera lógica y consistente, agrupando métodos relacionados y utilizando regiones si es necesario.

Las declaraciones using se organizarán alfabéticamente y se eliminarán los que no se utilicen.

Utilizaremos regiones para agrupar secciones relacionadas del código sólo cuando sea necesario.

En nuestro proyecto, empleamos una organización basada en carpetas para mantener una estructura ordenada. Cada carpeta cumple un propósito específico: 'Interfaces' alberga las interfaces de usuario, 'Entidades' contiene las clases de negocio, 'Control' gestiona la lógica principal, y 'Recursos' centraliza elementos visuales y multimedia como imágenes e iconos. Esta estrategia facilita la navegación y el mantenimiento de nuestro código

**Manejo de Excepciones e Interrupciones**

El manejo de excepciones e interrupciones es fundamental para crear código confiable y robusto. A continuación, se detallan las prácticas recomendadas para el manejo de excepciones e interrupciones en el código:

* Manejo de Excepciones:

Siempre utiliza bloques try-catch para capturar excepciones cuando sea necesario. Esto ayuda a prevenir que las excepciones no controladas detengan la ejecución del programa.

Limita el uso de bloques try-catch solo a excepciones específicas que esperas y que puedes manejar de manera significativa. Evita capturar excepciones genéricas como Exception.

Documenta las razones detrás de cada bloque try-catch para comprender mejor las circunstancias bajo las cuales se lanzarán las excepciones y cómo se manejarán.

* Lanzamiento de Excepciones:

Lanza excepciones que sean informativas y significativas para el contexto en el que ocurren. Proporciona mensajes de error claros y descriptivos.

Evita lanzar excepciones en situaciones normales de flujo de control. Utiliza el manejo de excepciones solo para situaciones excepcionales.

* Manejo de Interrupciones:

Si trabajas con operaciones largas o bloqueantes, considera el uso de hilos (threads) o tareas (tasks) para evitar que el hilo principal de la aplicación quede bloqueado. Esto garantiza una experiencia de usuario más receptiva.

Utiliza mecanismos de cancelación o tiempo de espera (timeout) cuando sea necesario para evitar que las operaciones bloqueantes se ejecuten indefinidamente.

* Pruebas de Excepciones:

Realiza pruebas exhaustivas de manejo de excepciones para garantizar que tu código se comporte correctamente en situaciones de error.

Verifica que las excepciones se capturen y manejen de manera adecuada, y que no se pierda información importante durante el proceso.

**Control de Versiones**

Utilizaremos un sistema de control de versiones (como Git) para gestionar el código fuente.

Seguiremos prácticas de ramificación y fusiones para un desarrollo colaborativo.

Documentamos nuestros commits de manera descriptiva y coherente

**Conclusión:**

La implementación de una User Story como desafío planteado a los estudiantes resultó en una oportunidad muy enriquecedora para profundizar la comprensión y promover la aplicación de los contenidos y tópicos desarrollados a lo largo de la asignatura.

En primera instancia, resultó un gran aporte para ejercitar la comprensión sobre las historias de usuario en su aproximación con la “Triple C” y la importancia de la conversación de las historias de usuario. En segunda instancia, el proceso de implementación de lo descrito en la historia de usuario también resultó en la ejercitación de habilidades técnicas y blandas que serán requeridas a futuro.

Se agradece a los docentes por el desafío planteado y la oportunidad brindada para la formación integral y profesional de los estudiantes.