FACULTAD
DE INGENIERIA
Universidad de Buenos Aires

PLAN DE PROYECTO

RESTAURANT ORDER MANAGMENT

ROM

75.47 - Taller de desarrollo de proyectos II

ALUMNOS

Angelani, Pablo
Garay Ojeda, Ignacio
López, Federico
Silva de Sousa, Gustavo

AYUDANTES

Degiovannini, Marcio
Sapia, Belén
Madeira, Mercedes

1. Alcance

Se requiere construir un sistema para gestionar los pedidos de un restaurante contemplado el flujo completo de los mismos, es decir desde que son encargados por un mozo en una determinada mesa pasando por su preparación hasta retornar al cliente que lo solicitó originalmente, y su correspondiente pago.

1.1 El proyecto incluye

Los requisitos funcionales son los siguientes:

- Gestión del armado del menú del restaurante con jerarquías de rubros y subrubros.
- Gestión del creación de mesas y su distribucion dentro del local.
- Apertura de mesas para atender clientes, con parámetros configurables.
- Armado y envio de pedidos según el menú para una mesa abierta.
- Administración de pedidos en la cocina/barra, gestionando y modificando el estado de preparación en que se encuentran los platos.
- Servicios de notificaciones para comunicación entre mozos y cocina/barra.
- Generacion de tickets y administracion de formas de pago para el cierre de las mesas
- Métricas varias por plato, mesa, empleado, etc

Los requisitos no funcionales son los siguientes:

- La aplicacion se dividirá en un frontend y en un backend. El backend será un servicio web mientras que el frontend será desarrollado bajo el sistema operativo Android.
- Ambas aplicaciones tendrán interfaces amigables y de navegabilidad intuitiva.
- Se proveerá acceso seguro y validación de usuarios

1.2 El proyecto no incluye

- **No** generará ningún tipo de factura al cerrar una mesa.
- **No** permitirá definir distribuciones físicas y/o planos del local.
- **No** incluye control de stock de mercaderías

2. Equipo de trabajo

El equipo de trabajo utilizará la metodología Scrum para la gestión y el desarrollo de software. Los roles principales de este marco de trabajo están definidos de la siguiente forma:

- ***Product Owner:*** representa la voz del cliente. Se asegura de que el equipo Scrum trabaje de forma adecuada desde la perspectiva del negocio. En el proyecto este rol estará representado por los profesores Marcio Degiovannini y Belén Sapia en la primera etapa, y por Mercedes Madeira en los sprints finales.
- ***ScrumMaster:*** encargado de eliminar los obstáculos que impiden que el equipo alcance el objetivo del sprint. El ScrumMaster no es el líder del equipo (porque ellos se auto-organizan), sino que actúa como una protección entre el equipo y cualquier influencia que le distraiga. El ScrumMaster se asegura de que el proceso Scrum se utilice como es debido. Durante el proyecto se elegirá de forma rotativa un ScrumMaster por iteración, con el fin de que todos los integrantes del grupo cumplan este rol.
- ***Equipo de desarrollo:*** tiene la responsabilidad de entregar el producto. El equipo esta conformado por 4 personas, que a su vez se dividirán en dos subequipos uno WEB y otro MOBILE, incentivado el trabajo de a pares.

3. Plan de entregas

El proyecto será administrado mediante el uso de la herramienta web *TargetProcess.* Esta es una herramienta de software intuitiva y flexible para administrar proyectos basados en SCRUM. Target Process permite definir Backlogs a nivel de User Stories, y a su vez desglosarlas en tareas, definir puntajes y duraciones de las mismas, obtener métricas como el Sprint Burndown Chart*,* etc.

El proyecto se organizará en iteraciones, todas ellas dentro de un mismo timebox. Se utilizarán User Story para definir los requisitos del cliente, las cuales a su vez serán agrupadas en Features. En cada iteración se seleccionan User Stories, priorizadas por el cliente, para ser realizados durante ese periodo de trabajo.

Se utilizarán los siguientes elementos de Scrum:

- ***Product backlog:*** es un documento de alto nivel para todo el proyecto que contiene descripciones genéricas de todos los requisitos, funcionalidades deseables, etc. priorizadas según su retorno sobre la inversión. Este documento debe realizarse antes de iniciar el primer sprint. Es abierto y sólo puede ser modificado por el *product owner*. Contiene estimaciones realizadas a grandes rasgos, tanto del valor para el negocio, como del esfuerzo de desarrollo requerido.
- ***Sprint***: es el período en el cual se lleva a cabo el trabajo en sí. Al final de cada sprint, el equipo deberá presentar los avances logrados, y el resultado obtenido es un producto potencialmente entregable al cliente. Para realizar el proyecto se definieron 6 sprints de 2 semanas de duración. Los mismos comienzan y terminan los lunes en la clase de la materia.
- ***Sprint backlog:*** documento detallado donde se describe cómo el equipo va a implementar los requisitos durante el siguiente sprint. Esta lista contiene las tareas necesarias para realizar los User Stories comprometidos en la interacción. Las tareas en el sprint backlog nunca son asignadas, son tomadas por los miembros del equipo del modo que les parezca oportuno.

4. Plan de estimación

La estimación del trabajo se hará a nivel de tareas de las User Story. Para esto se reunirá todo el grupo, analizando y desglosando cada User Story en las tareas que se requiere para poder completarla. Una vez que se termine de realizar esto se procederá a estimar.
La metodología de estimación utilizada será *Planning Pocker*. Ésta es una técnica para realizar una estimación basada en el consenso, en su mayoría utilizada para estimar el esfuerzo o el tamaño relativo de las tareas de desarrollo de software.

1. Se reparten cartas a cada integrante del equipo. Los valores de la carta siguen una numeración ascendente creciente, de manera de tener distancias pequeñas entre los números más bajos y altas en los números más grandes.
2. El Scrum Master lee cada una de las tareas de una User Story dada que debe ser estimada y todo el grupo trata de responder cada pregunta que surja.
3. Cada integrante del equipo debe elegir una carta con la estimación.
4. Cuando todos los integrantes del equipo han elegido, se descubren todas las cartas a la vez.
5. Las personas que hayan elegido los valores mayores y menores deben explicar el motivo de por qué consideran que se va a tardar más o menos.
6. Esta discusión hará pensar a todo el equipo sobre detalles no tenidos en cuenta, haciendo reiniciar el proceso hasta que en aproximadamente dos o tres rondas se debería llegar al consenso.

Los valores de las cartas representarán puntos, definiendo antes de estimar una tarea equivalente a 1 punto. Una vez definidos los puntos de cada tarea, se obtiene el puntaje total de la User Story. En la estimación del primer sprint se asignará una equivalencia entre puntos y horas de trabajo por consenso grupal. Luego a partir del trabajo realizado se podrá saber cuantas horas de trabajo efectivamente el grupo necesita para poder completar un punto.

La numeración de las cartas seguirá la serie de Fibonacci.

5. Plan de comunicación

La comunicación dentro del proyecto será realizada tanto mediante reuniones presenciales como por email. Al inicio de cada sprint (salvo el primero) se realizará una Sprint Demostration que contará como una entrega formal donde se demostrará lo realizado en el sprint anterior. Luego de esto, se realizará una Sprint Planning Meeting para el nuevo sprint, donde el Product Owner definirá y priorizará los User Story de la nueva iteración.

El siguiente lunes al inicio de cada sprint, se realizará una reunión informal con el Product Owner, para realizar consultas y mostrar el avance del equipo.

Una vez finalizado un Sprint, se hará una Sprint Retrospective Meeting en la cual sólo participará el equipo, con el fin de aprender de los logros y errores cometidos en la iteración anterior.

Por otro lado el grupo no realizará Scrum Daily Meetings, ante la imposibilidad de coincidir todos los días de la semana en la facultad.

Todas las reuniones con el Product Owner serán registradas mediante una Minuta de Reunion, que luego deberá ser enviada para ser validada. Además para cada reunión el grupo deberá presentar un reporte de avance.

6. Plan de administración de riesgos

Cada riesgo detectado será identificado con un Id, descripción y el cálculo de exposición al mismo. A su vez, se pondrá el responsable de hacer un seguimiento y control del mismo y la posibilidad de realizar un plan de mitigación y/o de contingencia.

La probabilidad de ocurrencia estará determinada por la siguiente tabla:

Rango de probabilidad	Promedio para el calculo	Expresión de lenguaje natural	Valor numérico
de 1% a 10%	5 %	Baja	1
de 11 % a 25%	18 %	Poco probable	2
de 26% a 55%	40 %	Media	3
de 56% a 80%	68 %	Altamente probable	4
de 81% a 99%	90 %	Casi seguro	5

Por otro lado, el impacto del riesgo seguirá estos valores:

Evaluación del Impacto de los Riesgos en los Factores Críticos de éxito					
Objetivos Proyecto	***Muy Bajo .05***	***Bajo .1***	***Moderado .2***	***Alto .4***	***Muy Alto .8***
Costo	Incremento insignificante del costo	Incremento del costo < 5%	Incremento del costo < 5 - 10 %	Incremento del costo < 10 - 20%	Incremento del costo > 20%
Tiempos	Desvíos insignificantes	Desvío < 5%	Desvío < 5 - 10 %	Desvío < 10 - 20%	Desvío > 20%
Funcionalidad	Ha decrecido muy poco	Poca funcionalidad ha sido afectada	Varias áreas han sido afectadas	La reducción de funcionalidad es inaceptable para el cliente	El producto final no será útil
Calidad	Ha decrecido muy poco	Ocasionales usos han sido afectados	La reducción de la calidad requiere aprobación del Cliente	La reducción de la calidad es inaceptable para el cliente	El producto final no será usable

Ya que en el proyecto actual no se manejan costos ni estándares de calidad, el impacto del riesgo se medirá en tiempo.

De esta forma se utilizará **5** tipos de probabilidad (Baja / Poco probable / Media / Altamente probable / Casi seguro) y **5** grados de impacto (Muy bajo/ Bajo / Moderado / Alto / Muy Alto).

El factor de exposición al riesgo se calcula como el producto entre la probabilidad de ocurrencia y el impacto al proyecto.

En caso de que el grado de exposición al riesgo sea **>= 0.1** se establecerá un plan de mitigación. Por otro lado, sólo los riegos con exposiciones elevadas tendrán un plan de contingencia definido. De esta forma se definió que si la exposición es **>= 0.6** se detallará un plan de contingencia, con su umbral disparador correspondiente.

El riesgo podrá evolucionar en las siguientes formas:

- = : Su grado de exposición se mantuvo con respecto al último estado.
- X : El riesgo fue cerrado.
- ↑ : Su grado de exposición aumentó con respecto al último estado.
- ↓ : Su grado de exposición disminuyó con respecto al último estado.

7. Plan de control de avance

Indicadores

Los indicadores de control son herramientas útiles a la hora de preguntarnos acerca del avance del proyecto. En distintas partes del proyecto necesitamos saber cómo nos encontramos en cuanto a lo realizado con respecto a lo planificado, también necesitamos saber de todos aquellos defectos encontrados, a que ritmo se están cerrando. Los indicadores ayudan a encontrar las respuestas a esas preguntas. En este proyecto utilizaremos tres tipos de indicadores. Cada uno de ellos se detallan a continuación.

Burndown Charts

Un diagrama burn down es una representación gráfica del trabajo por hacer en un proyecto en el tiempo. Usualmente el trabajo remanente (o *backlog*) se muestra en el eje vertical y el tiempo en el eje horizontal. Es decir, el diagrama representa una serie temporal del trabajo pendiente. Este diagrama es útil para predecir cuándo se completará todo el trabajo. Usualmente se usa en el desarrollo ágil de software, especialmente con Scrum.

Se van a utilizar los siguientes gráficos de esfuerzo pendiente:

- Días pendientes para completar los requisitos del producto o proyecto (Product Burndown Chart), realizado a partir del User Story.
- Horas pendientes para completar las tareas de la iteración (Sprint Burndown Chart), realizado a partir del Sprint Backlog.

Evolución de la prueba

Se registran los defectos nuevos a medida que aparecen. Se registran los cambios de estado de los mismos (cerrados, pendientes, suspendidos) hasta que se cierren definitivamente.

Nos permite conocer los tiempos de corrección y por lo tanto podemos obtener estadísticas de velocidad de corrección (cantidad de defectos nuevos por día o cantidad de defectos cerrados por día). Esto nos permite proyectar cuando tenemos una muestra representativa.

Ayuda a su vez al proceso de priorización de los casos de prueba.

Cobertura de la prueba

Al registrar los casos de prueba, se pueden obtener mediciones del avance real de la prueba en el proyecto, lo que permite comparar la cantidad total de casos planificados para el sistema, con las cantidades reales de casos ejecutados y ejecutados sin error. Este indicador del avance o cobertura de la prueba, es uno de los más claros y objetivos que se pueden obtener.

8. Plan de pruebas

Definir cuáles serán las condiciones a probar en un sistema será una etapa fundamental del proceso. Mediante las Historias de usuario que utilizaremos para definir el comportamiento del sistema, los diferentes cursos de acción de dichas historias darán origen a las condiciones de pruebas funcionales. Otra posible fuente de estas condiciones puede ser la lista de requerimientos de nuestro sistema.

Para obtener las condiciones de prueba no funcionales, será necesario consultar una mayor variedad de fuentes. La primera fuente a utilizar deberán ser los requerimientos no funcionales del sistema. Allí deberemos establecer todos los requisitos que debe cumplir el sistema en lo referido a tiempos de respuesta esperados, volúmenes de información a manejar, configuraciones de software y hardware sobre las que debe correr, recuperación ante fallas, seguridad, etc. Otras fuentes de requerimientos no funcionales son los estándares de desarrollo del desarrollador y del cliente, estándares de pantallas, informes, formatos u otros.

Una vez obtenidas todas las condiciones de prueba, podremos determinar la necesidad de priorizar unas por sobre otras, ya que no alcanzaremos a probar todas debido a que algunos módulos son más complejos o críticos.
Con las condiciones de prueba definidas comenzaremos a definir los casos de prueba necesarios para cada condición.

Mantenimiento de condiciones y casos de prueba

Los casos de prueba deberán poder ser reutilizados a lo largo de las diferentes pruebas realizadas al sistema durante su ciclo de vida, en sucesivas tareas de mantenimiento. Por eso, es necesario que ante cada cambio que sufra el sistema en su desarrollo o mantenimiento, se revisen los casos de prueba, para ver cuáles de ellos necesitan actualización y cuáles se seleccionarán para hacer pruebas de regresión luego de las modificaciones. Se deberá tener en cuenta : qué casos se vuelven obsoletos, para qué casos cambia el resultado esperado y si es necesario incorporar nuevos casos.

El entorno y los datos para las pruebas

Las pruebas se deberán realizar en un entorno aislado, asegurando que los datos sean extraídos de la realidad o que sean semejantes a la realidad y que, en la medida de lo posible, cubran la totalidad de casos previstos para realizar las pruebas.

Se deberá tener en cuenta el factor de riesgo de no poder contar con datos “reales” para las pruebas, la generación manual de datos puede producir errores que perjudiquen la consistencia de las mismas.

Definición de la prueba

Por cada User Story se definirán todos los casos de prueba para su verificación. Cada caso de prueba será identificado unívocamente, y contendrá en detalle los valores de entrada, los resultados esperados y los pasos a seguir para su realización.

Ejecución de la prueba

La ejecución de las pruebas también se asocia al User Story en cuestión. Cada ejecución lleva el número correspondiente a la definición del caso de prueba, su nombre, Estado de la ejecución, severidad del defecto encontrado y una observación en cuanto a la causa de su falla. Esta última corresponde a una apreciación del tester, con el fin de poder orientar a los especialistas a reparar el defecto.

Bugs

Los *bugs* que sean detectados en una ejecución de pruebas serán identificados con un ID autoincremental. Se registrará la User Story al que pertenece, una pequeña descripción del problema, la fecha en que fue encontrado, quien se encuentra solucionandolo y quien lo verificará.

El bug posee un grado de criticidad, el cual marca la prioridad para su solución.
Los grados posibles son:

- **Alto:** el cliente no puede completar el flujo del programa por ese defecto. Estos defectos son considerados **críticos**. Un User Story es liberado cuando carece de defectos de este tipo.
- **Medio:** error de programación pero que no impide completar el flujo normal de la aplicacion.
- **Bajo:** el cliente puede hacer uso de la funcionalidad completa. Se trata sólo de un defecto de estética o navegabilidad.

Los distintos estados por los que puede pasar un *bug* son los siguientes:

- **Nuevo:** cuando es detectado en la ejecución de una prueba.
- **En proceso:** cuando el bug está siendo tratado para ser solucionado.

- **Resuelto:** cuando se encuentra solucionado pero no verificado por la persona asignada
- **Cerrado:** cuando la persona asignada verificó que el bug se encuentra solucionado.
- **Rechazado:** cuando el bug termina siendo rechazado ya que no es considerado como tal por el Product Owner.
- **No reproducible:** cuando el bug no pudo volver a ser visualizado por el grupo.
- **Suspendido:** se decide no solucionarlo por el momento. Solo los *bugs* con grado de severidad **Bajo** pueden suspenderse. Generalmente se suspenden aquellos no severos que son difíciles de solucionar o que no son prioridad por el cliente.
- **No resuelto:** bug que por diversos motivos se decidio no resolver.

8.Plan de aceptación

Con la definición de cada User Story que conforman el Product Backlog, luego de priorizarlos y elegir cuales formarán parte del próximo Sprint, se definirá con el Product Owner los casos de prueba que se ejecutarán para su aceptación.

Al finalizar el sprint se mostrarán los User Story realizados probando los casos de prueba definidos en el comienzo de la iteración.

De esta metodología de aceptación se espera que, llegado el caso, sólo surjan defectos relacionados con los requisitos no funcionales del sistema.

Criterio de aceptación

La aceptación será realizada en la Reunión de demostración (Sprint demostration) mediante la ejecución de **sólo** los casos de prueba determinados en la definición del User Story. Se llevará el sistema con los User Story realizados hasta el momento, instalado en una notebook al lugar de reunión. El Product Owner ejecuta los casos de prueba y se verifican los resultados.

La aceptación de la entrega se encuentra definida por el siguiente criterio:

- ningún bug de criticidad alta por funcionalidad
- hasta 1 bug de criticidad media por funcionalidad
- hasta 2 bugs de criticidad baja por funcionalidad

Plan de trazabilidad

La trazabilidad se mantendrá directamente entre los artefactos. Cada User Story posee un número que lo identifica unívocamente. La herramienta *TargetProcess* mantiene la trazabilidad entre User Story, tareas asociadas, responsables y Sprints.
Con respecto a las pruebas, se mantendrá la Definición de la prueba la cual asocia cada User Story con todos los casos de pruebas. Por otro lado se tiene la Ejecución de la prueba la cual una de sus columnas es el número de Definición de la prueba.

Finalmente, todos los Defectos (Bugs) detectados en la Ejecución de la prueba llevarán un número identificatorio que permitirá saber en qué prueba fue encontrado y en que corrida.
De esta manera se puede saber, a partir de cualquier artefacto, por ejemplo un *bug,* en qué prueba fue detectado, y por lo tanto a que User Story pertenece. En caso de que el *Product Owner* determine la modificación de un User Story se podrá saber cuáles tareas, casos de prueba, ejecuciones y bugs son afectados.