Memoria

**1.-Introducción**

Inicialmente, nos pusimos a debatir sobre qué herramientas serían más convenientes para el *Product Backlog* y el *Sprint Backlog*.

Tras haber acordado que usaríamos *Jira* para gestionar el *Product Backlog* y el *Sprint Backlog*, el equipo se dividió en dos grupos. Uno de ellos estaba formado por el ***Product Owner* y el *Scrum Master***, que se dedicaron a escribir sobre las historias de usuario que formarían posteriormente el *Product Backlog*.

Al principio, quedaron bastantes Historias de Usuario épicas, pero con ayuda de todo el grupo se fueron simplificando y dividiendo en unas más simples, que serían sobre las que se harían los criterios de aceptación posteriormente.

Tras ser validadas por todo el equipo, serían acompañadas de los criterios de aceptación.

El otro grupo, que estaba formado por los **desarrolladores**, se dedicó a practicar con las herramientas y tecnologías que se usarían en el proyecto (mientras el Product Owner y el Scrum Master elaboraban las Historias de Usuario) para más tarde crear el prototipo.

Se acordó que el prototipo tendría varias funciones básicas que permitiera tanto manejar la parte de interfaces gráficas de Netbeans, como crear y mantener una base de datos con SQLite (previamente creada) en lenguaje Java. Además, habría que hacer un *login* al iniciar el prototipo.

Tras probar el prototipo, analizar y corregir errores, se preparó para que se pudiera ejecutar desde un *.jar* y generara automáticamente la base de datos.También se elaboraron las instrucciones de ejecución correspondientes, y se redactó las dificultades encontradas a la hora de realizar el prototipo.

**2.-Historias de usuario**

En los siguientes apartados explicaremos las herramientas usadas para que el Product Owner y el equipo de desarrollo puedan gestionar el Product Backlog y el Sprint Backlog, respectivamente.

También se explicará el proceso de validación de las Historias de Usuario, y cómo el Product Owner las priorizó posteriormente.

Como añadido, hemos incluido los Mockups de la aplicación, y perfiles de usuario.

1. **Herramienta usada para gestionar el Product Backlog y el Sprint Backlog**

Para el *Sprint Backlog* y *Product Backlog* decidimos usar **Jira**.

Una de nuestras opciones era usar *Trello* para la gestión del Sprint Backlog, debido a que su uso era sencillo. Sin embargo, Jira tiene más funcionalidades, tiene un uso bastante extendido, y con otras herramientas como GitHub (que también usaremos, pero para el control de versiones), es más difícil dividir el *Product Backlog* en tareas.

Primero probamos *Trello*, que resultaba ser muy amigable para el usuario, pero no tenía demasiadas funcionalidades. Posteriormente, comprobamos que en GitHub había que crear las tareas de cada Historia de Usuario a mano, por lo que vimos que era muy costoso.

Por último, usamos *Jira*. Siguiendo algunos tutoriales, no resultaba demasiado difícil de manejar, y era una herramienta bastante útil para metodologías ágiles.

Tras probar las herramientas anteriormente mencionadas y debatir cuál sería la más adecuada, decidimos usar ***Jira***, pues a pesar de que es algo complejo de usar al principio por la cantidad de funcionalidades que ofrece, teníamos la oportunidad de usar una herramienta más completa y a la que se le suele dar uso de forma profesional.

**b) Validación de Historias de Usuario**

**c)** **Priorización de las Historias de Usuario**

**d)** **Mockups creados**

[Insertar mockups, y explicarlos brevemente]

**e) Perfiles de usuario**

[Insertar perfiles de usuario]

**3.-Prototipo**

En este apartado se explicará tanto las tecnologías probadas en el prototipo que realizamos, como su desarrollo, problemas encontrados, cómo se han resuelto, y finalmente se explicará cómo ejecutarlo y probar las funcionalidades que este ofrece.

**a) Herramientas y tecnologías probadas**

Además de las herramientas usadas para que el *Product Owner* gestione el *Product Backlog*, debatimos sobre qué herramientas usar para desarrollar, y para gestionar las versiones del código de proyecto, documentación, etc.

El *Product Owner* indicó que sería una aplicación de escritorio, por lo que para programar barajamos varias opciones como **Eclipse**, o **Netbeans**, ya que son plataformas para programar en Java bastante conocidas y amigables. Elegimos esta última, pues aunque ya habíamos trabajado con Eclipse anteriormente, Netbeans incorporaba el *JavaFx*, una herramienta que sirve para diseñar interfaces gráficas. En caso de elegir Eclipse, hubiéramos tenido que instalar el plugin aparte, por lo que nos decantamos por la elección más fácil a la hora de instalación (**Netbeans**), ya que esta plataforma no difiere mucho de Eclipse.

Para el servidor de la base de datos, estuvimos debatiendo sobre el uso de **Xampp** y **Wamp**, debido a que a la hora de hacer la conexión de la base de datos con el código es bastante simple, y varios integrantes del grupo lo han utilizado previamente, lo que puede a ayudar a resolver dudas en este periodo de prueba de herramientas.

Otra ventaja que aportaban estos programas es que, al unificar varios servidores, permiten elegir en la instalación qué servidor queremos usar.

Ya que el prototipo sería ejecutado desde un *.jar*, y se tendría que generar la base de datos a partir de dicho ejecutable sin depender de software externo, se propuso también el uso de **SQLite**, ya que va integrado dentro del propio proyecto, y sería sencillo realizar la conexión a la base de datos.

Debido a las restricciones de la entrega del prototipo, y habiendo probado las plataformas mencionadas, decidimos usar **SQLite** para las bases de datos. Tras acordar esto, se comentó que con **SQLiteStudio** se podría gestionar fácilmente la base de datos creada.

Tras haber probado el *SQLiteStudio*, y haber creado algunas tablas en la base de datos que usaríamos para el prototipo, nos dimos cuenta de que fallaba la inserción porque el *SQLiteStudio* nos daba problemas. Ya que este programa no era tan amigable como pensábamos, probamos también con el ***DB browser for SQLite*,** una herramienta similar.

Estuvimos probándola, y además de que no nos daba ningún error al hacer las inserciones en la base de datos (ni de otro tipo), resultó ser más intuitiva y amigable que el *SQLiteStudio*, por lo que decidimos usar al final el *DB browser for SQLite* para gestionar las tablas de la base de datos.

**b)** **Desarrollo del prototipo**

A la hora de crear el prototipo nos dividimos en **pequeños subgrupos** dentro de los propios desarrolladores, y nos repartimos equitativamente pequeñas partes de lo que sería el prototipo. Independientemente de la parte de código que implementara cada persona del equipo, nos aseguramos de que todo el mundo (incluidos el *Product Owner* y el *Scrum Master*, pues les podría servir para poner en práctica lo aprendido de metodologías ágiles) hubiera practicado con las herramientas que serían necesarias para el proyecto, para que el desarrollo de este fuera lo más fluido posible.

Comenzamos haciendo las interfaces gráficas que componen el prototipo, y las implementaciones de los botones que las forman, de las cuales se encargaron Carlos, y Guillermo. Hubo problemas al principio sobre todo con el manejo del repositorio de GitHub, y también con la creación de interfaces gráficas, pues era la primera vez que usábamos tanto GitHub como NetBeans. Dicha inexperiencia se solucionó consultando foros y tutoriales de internet, en los que hacen una explicación sencilla sobre cómo usar dichos programas.

Posteriormente, Marina, Jun, Sergio y Laura se encargaron de programar la parte de Controlador, así como de la creación de Factorías, de las SA, los DAOs, y de la conexión con la base de datos. Aquí volvimos a tener un problema por inexperiencia con NetBeans, que se solucionó de la misma forma que con las interfaces. Sin embargo, a la hora de hacer la conexión con la base de datos mediante el *SQLite*, (generando al principio las tablas con el *SQLite Studio)* hubo ciertas dificultades, pues, de nuevo, nos encontrábamos ante un programa que no habíamos usado anteriormente. Gracias a la colaboración de otros compañeros del equipo que habían usado el SQLite antes, y de nuevo, a tutoriales y foros de internet, pudimos realizar la conexión a la base de datos correctamente.

Posteriormente integraríamos todo. Debido a que hicimos todo el código por partes, tuvimos que integrarlo tras haber programado cada uno nuestra parte. De la integración se encargaron inicialmente tanto Diego, como Eduardo, que se encargaron de modificar, y organizar tanto el código como las clases java para que pudiera ejecutarse correctamente. No resultó ser una tarea fácil, pues cada persona programaba de una forma concreta; ponía nombres de variables, clases, eventos, etc. distintos. Por ello, se pusieron de acuerdo las personas encargadas de integrar y adaptar todo para hacerlo de la misma manera, y así unificarlo todo.

Tras haber ejecutado por primera vez todo junto, surgieron errores inesperados que después solucionaríamos. En primer lugar, se mostraba constantemente una ventana cuando no debía hacerlo. Solo tuvimos que añadir unas líneas de código para solucionarlo.

Nos dimos cuenta de que, al insertar información en la base de datos, nos saltaba un error que no entendíamos de dónde venía, pues esa parte de código estaba correctamente implementada. Después descubrimos que fue debido al *SQLite Studio*, pues este corrompía los archivos de la base de datos, impidiendo así la inserción de datos en esta. Como barajamos la posibilidad de usar el *DB browser for SQLite,* y comprobamos que también era intuitivo y amigable, como ya se ha mencionado anteriormente, acordamos usarlo para gestionar las tablas de la base de datos, en vez de usar el *SQLite Studio.* Con este nuevo programa, no nos dio ningún problema al hacer las inserciones, y las operaciones realizadas sobre la base de datos funcionaban correctamente.

Tras tener un prototipo funcional básico, Eduardo se encargó de implementar un menú login, en el que tendrías que introducir la contraseña del administrador (el usuario y contraseña estarían en la base de datos guardados) para acceder al menú principal del prototipo. Surgieron errores de los que se encargaron tanto Guillermo como Diego (entre otros, que este menú se bloqueaba si no introducías la contraseña correcta), que se solucionaron modificando código.

También se añadió un *“bracket”* compuesto por cuatro equipos, y un panel en el menú principal para mostrar los mensajes de error y confirmación. Hubo algunos problemas con el bracket, pues se bloqueaban los equipos, y queríamos que cada vez que se seleccionara uno de los dos equipos que avanzaría (pues estaban enfrentados de dos en dos), se cambiara el equipo “finalista”. Se solventó, de nuevo, modificando código.

Por último, se solucionó un error que permitía que se insertaran datos con el mismo nombre en la base de datos, en el que simplemente hubo que añadir más comprobaciones en el código.

**c) Instrucciones de ejecución y cómo probar el prototipo**

Para ejecutar el proyecto, primero es necesario **ejecutar el .jar**.

Aparecerá la ventana de **login**. Hay que **iniciar sesión** con la contraseña *“admin”.* En el caso de introducir la contraseña errónea, simplemente saldrá un mensaje de error. Si se introduce correctamente, aparecerá el menú principal del prototipo.

Inicialmente, la **base de datos estará vacía**. Tendrán que insertarse datos introduciendo los nombres que se desea tener en la base de datos en el cuadro de texto, y pulsando el botón “Insertar”. Se mostrará un mensaje de confirmación si se ha podido insertar en el panel del menú, y si por el contrario ha habido algún error, saldrá un mensaje de inserción fallida.

Para **mostrar los datos introducidos**, solo hay que pulsar el botón “Mostrar”.

En el menú principal aparecerán también el **bracket** cuatro **equipos predefinidos** “enfrentados” de dos en dos. Cada vez que se haga click en uno de los cuatro equipos, avanzará como finalista.

Finalmente, al pulsar el **botón “Hola Mundo”** aparecerá una alerta en la que se muestra un “Hola mundo!”.

**4.- Dinámica grupal**

Nuestro grupo desarrolló una dinámica grupal para mejorar la convivencia y trabajar mejor en equipo.

A continuación, haremos una breve descripción de la dinámica, explicaremos cómo se desarrolló, y cuáles fueron los resultados de esta.

1. **Descripción de la dinámica**

Para esta ocasión, una vez que el grupo ha tenido la oportunidad de conocerse y trabajar juntos, se ha decidido realizar el juego “Cuenta una historia” En esta actividad, un miembro del grupo empieza con una palabra (vale cualquier tipo de palabra: sustantivo, verbo en cualquier forma, pronombre, adjetivo, etc.…) y el resto va añadiendo una palabra más a la anterior dicha, creando así una historia entre todos los participantes. Este juego se ha elegido para dar importancia a mostrar atención a todos los miembros del equipo. En esta actividad no se han centrado tantos los esfuerzos en ejercitar la memoria, sino en que todos los participantes se vieran obligados a prestar atención a las palabras de los demás para el buen desarrollo del ejercicio.

1. **Desarrollo**

La actividad fue realizada el día 8 de marzo en la cafetería de la facultad. Esta vez los participantes fueron quienes organizaron la actividad entre todos. Se estableció un turno para poder recitar la historia y poder añadir una palabra nueva. Se hicieron varias historias, ya que cada vez que un jugador se equivocaba, se debía empezar de nuevo.

1. **Resultados**

Tras esta actividad, el ambiente en el grupo ha mejorado. Se tiende a prestar atención a cada persona cuando da su opinión o personalidad, sin silenciarlo ni ignorarlo.