Habilidades técnicas en la etapa de requerimientos

1. Gestión de proyectos

Acorde al ciclo de la ingeniería de requerimientos, la correcta administración y desglose de las tareas llevada a cabo por la gestión del proyecto puede suponer un aumento en la calidad de la documentación y los requerimientos como tal, tales errores pueden comprometer el diseño e implementación del proyecto. "La gestión de proyectos aporta al equipo objetivos precisos y determinados, así como sistemas mejorados para el desenvolvimiento de las tareas. Esto permite que todo el equipo de trabajo pueda mantener una sintonía, colaborando con mayor coherencia y fluidez." (Tiffin University, 2023).

> Entre las ventajas de la correcta gestión de los requisitos existe una mejora en la calidad al realizar la entrega final, menor costo de desarrollo, una entrega más rápida y una mejora general en la configuración del proyecto. (IBM, s.f).

La gestión de proyectos beneficia los costos de desarrollo, ya que al tener una correcta documentación de requerimientos, producto del desglose efectivo de actividades y priorización de estas, se pueden fijar errores en el sistema de manera más rápida durante el ciclo de desarrollo, evitando aumentos en los costes de este.

2. Análisis de seguridad

> Uno de los problemas más comunes en el desarrollo de software es que la seguridad se aborda en una etapa demasiado avanzada del proceso: la de pruebas, después de haber completado las tareas más importantes de diseño e implementación. (Red Hat, s.f).

Dentro de la ingeniería de requerimientos, definir las necesidades y limitaciones en seguridad de un sistema final permiten anticipar o prevenir posibles problemas durante etapas más avanzadas del desarrollo. La gestión de la seguridad puede llegar a ser una de las piedras angulares de un software, y una de las necesidades tanto del cliente como de los usuarios:

> Estas cuestiones pueden ser pasadas por alto incluso en la etapa de pruebas si no se definen con anterioridad, y puede incrementar altamente los costos y provocar una regresión en el proceso de desarrollo. (Red Hat, s.f).

Es por lo anterior que la importancia de la seguridad en el ciclo de ingeniería de requerimientos es claro, ya que puede llegar a comprometer el sistema e incluso la necesidad de especificación de otro(s) requerimiento(s), así como su calidad y tiempo de desarrollo.

3. Análisis de factibilidad

La factibilidad de un proyecto se define a partir de la etapa de requerimientos, ya que genera conclusiones acerca de las funcionalidades del sistema y es un inicio a la gestión de los requerimientos, a la negociación con el cliente, la estimación de costos y tiempo de desarrollo. Pressman (2010) menciona al estudio de la factibilidad como una de las actividades desempeñadas para el proceso de V&V, ya que se extiende hasta el área de pruebas y las necesidades de los usuarios. (p. 384).

Lo anterior apunta a que un correcto estudio inicial de la factibilidad trae beneficios en la gestión completa de los requerimientos, lo cual facilita procesos posteriores y minimiza los cambios en los requerimientos.

Especificación de Casos de Uso vs Historias de Usuario.

Las Especificaciones de Casos de Uso y las Historias de Usuario son dos herramientas utilizadas en el desarrollo de software para capturar los requerimientos del sistema desde el punto de vista del usuario. Sin embargo, tienen diferencias significativas en términos de su enfoque y nivel de detalle.

Especificación de Casos de Uso

"Los Casos de Uso son artefactos que describen en términos coloquiales, es decir, en el lenguaje del usuario, lo que un sistema debe permitir a un actor que interactúa con él" (Cesar, 2020). Un Caso de Uso es la declaración de una interacción de un Usuario con el Sistema. "Los Casos de Uso llegan a más detalle en la documentación" (Ablancodev, 2022). Se recomienda su uso cuando se tiene un conjunto de características que son similares en más de un caso de uso y no se desea mantener copiada la descripción de la característica.

Historias de Usuario

Las Historias de Usuario sirven para describir lo que el usuario desea ser capaz de hacer. Las historias de los usuarios se centran en el valor que viene de usar el sistema en lugar de una especificación detallada de lo que el sistema debe hacer. Las historias de usuario dejan detalles sin especificar, para provocar conversaciones en las reuniones scrum.

Cuando usar cada una

Los Casos de Uso son útiles cuando se necesita un nivel de detalle más alto y una especificación más formal de los requerimientos. Por otro lado, las Historias de Usuario son más adecuadas en un ambiente de colaboración y se utilizan para fomentar la comunicación. "Las Historias de Usuario son especialmente útiles en metodologías ágiles de desarrollo de software" (Sánchez, A. L. L, 2016).

Ejemplo

Supongamos que estamos desarrollando un sistema de gestión de bibliotecas. Un Caso de Uso podría ser "Un usuario busca un libro por título". Este caso de uso tendría un flujo de eventos detallado, precondiciones, postcondiciones, etc. Por otro lado, una Historia de Usuario para el mismo sistema podría ser "Como usuario, quiero poder buscar un libro por título para que pueda encontrar rápidamente lo que estoy buscando". Esta historia de usuario es menos formal y más centrada en el valor que el usuario obtiene del sistema.

Producto resultante en la etapa de diseño

Responsive Web Design (RWD): Es un método asosciado a la creación de interfaces de usuario, mediante el cual se garantiza la visibilidad del sitio en cualquier dispositivo, pues se sigue una serie de caracteristicas porporcionadas por el wireframe que maneja el sistema para el que se esté desarrollando la interfaz. Gracias a este método se reducen los costos de creación y mantenimiento, ya que el diseño de las pantallas es similar entre dispositivos de diferentes tamaños.

En el caso de nuestro proyecto, usamos el RWD para desarrollar la interfaz de usuario de Turiamigos, pues tomamos como guía el wireframe que proprociona Apple para desarrollar aplicaciones en sistema IOS.

Principios del Manifiesto presentes en metodologías

Design Sprint

En la metodología design sprint se pueden encontrar varios de los principios de las metodologías ágiles del manifesto:

1- Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.

En esta metodología, con cada sprint se realiza un prototipo en un lapso de cinco días. El último de estos cinco días es dedicado a probar el prototipo, llevándolo a testers y al cliente en cuestión, después de esto, se realizan entrevistas tanto al cliente, como a los testers para así saber qué se debe modificar para la realización del producto final. (De Sá Araújo et al., 2019)

Este principio representado en el proyecto de equipo, se implementa al momento de la realización de los prototipos, en el caso de nuestro proyecto, de media fidelidad, para así poder tener una idea del funcionamiento del software.

2- Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.

Cada sprint cuenta con cinco días, de los cuales tres van destinados a la pura planeación del proyecto, en los cuales se puede realizar cualquier modificación. De igual manera, en los siguientes dos días se puede realizar la modificación de algún requisito o

algún caso faltante, ayudándose de las entrevistas que se harán el último día. (De Sá Araújo et al., 2019)

En el caso de TuriAmigos los requisitos están en constante mantenimiento y cambio conforme cada sprint, se pueden tanto agregar, como desechar, lo cual usualmente depende de varios factores, tales como, alguna idea nueva o el caso de la dificultad de algún requisito.

3- El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.

"It is important to drive the sprint in the same room throughout the week so we can have white-boards that help us keep the best ideas and check if the focus of the work is not distancing itself from the main idea." [Es importante que el sprint sea llevado a cabo en la misma habitación durante la semana para así tener tableros que nos ayuden a conservar las mejores ideas y revisar si el enfoque del trabajo no se está distanciando de la idea principal.] (De Sá Araújo et al., 2019, p. 297)

Aplicado en TuriAmigos, se podría cambiar el estilo de las reuniones, en vez de ser en línea, los integrantes del equipo podríamos reunirnos después de clases gracias a que los horarios de todos los miembros coinciden.

Extreme Programming XP

1- Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.

"El tratamiento de las historias de usuario es muy dinámico y flexible, en cualquier momento historias de usuario pueden romperse, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas." (Letelier, 2006)

En el caso de TuriAmigos los requisitos están en constante mantenimiento y cambio conforme cada sprint, se pueden tanto agregar, como desechar, lo cual usualmente depende de varios factores, tales como, alguna idea nueva o el caso de la dificultad de algún requisito.

2- Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.

La idea es producir rápidamente versiones del sistema que sean operativas, aunque obviamente no cuenten con toda la funcionalidad pretendida para el sistema pero que constituyan un resultado de valor para el negocio. Una entrega no debería tardar más 2 meses. (Letelier, 2006)

Con la realización y mantenimiento constante de un prototipo de mediana o alta fidelidad, en un periodo menor a dos meses, para así poder evaluar dicho prototipo e irlo mejorando con cada iteración.

3- A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.

La fase de producción requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente. Al mismo tiempo, se deben tomar decisiones sobre la inclusión de nuevas características a la versión actual, debido a cambios durante esta fase. Es posible que se rebaje el tiempo que toma cada iteración, de tres a una semana. Las ideas que han sido propuestas y las sugerencias son documentadas para su posterior implementación (por ejemplo, durante la fase de mantenimiento). (Letelir, 2006)

Para la aplicación del anterior principio en TuriAmigos, se realizarían reuniones específicamente para la revisión del desempeño y la mejora de este, para así poder llevar a cabo la realización de un mejor proyecto.

Método o técnica para incluir pruebas durante el desarrollo de software.

La integración continua es una práctica de desarrollo de software mediante la cual los desarrolladores combinan los cambios en el código en un repositorio central de forma periódica, tras lo cual se ejecutan versiones y pruebas automáticas.

La necesidad de implementar el método de integración continua es que, al momento de realizar los cambios, la acumulación de errores en el código es menor. La integración continua permite enviar los cambios de forma periódica al repositorio compartido con un sistema de control de versiones como Git. Un ejemplo en nuestra experiencia dentro del desarrollo de proyecto sería la utilización de GitHub Desktop como interfaz visual de Git que nos permita enviar los cambios de manera constante directo al repositorio.

Habilidades indispensables que un Ingeniero de Software debe incluir en su formación académica para el desarrollo de Aplicaciones de Software Seguras.

Para ser ingeniero de software y tener una completa seguridad de que tu software esta protegido es crucial adquirir habilidades sólidas en el desarrollo seguro de aplicaciones.Esto incluye una profunda comprensión de lenguajes de programación como Python, Java y C/C++, así como la capacidad de aplicar principios de diseño orientado a objetos para crear software flexible y resistente. Además, se requiere una competencia destacada en pruebas de software y depuración para garantizar la robustez y fiabilidad del código.

La seguridad en el desarrollo de software implica integrar prácticas seguras en todas las fases del ciclo de vida del desarrollo, desde la concepción del diseño hasta la implementación y el mantenimiento continuo. La conciencia de la criptografía y los principios de seguridad de redes es esencial para salvaguardar la confidencialidad y la integridad de los datos. Además, mantenerse actualizado con estándares y regulaciones de seguridad garantiza que las

aplicaciones cumplan con las normativas de la industria, proporcionando así un entorno digital seguro y confiable para los usuarios finales.

Principales obstáculos que se podrían presentar a nuestro proyecto de desarrollo respecto los valores contenidos en el Manifiesto Ágil

1- Expectativas poco claras de los turistas

Si los usuarios de la aplicación tienen expectativas vagas o no pueden expresar claramente sus preferencias y expectativas de encuentro, la entrega temprana y continua de características valiosas podría generar cambios frecuentes en los requisitos, lo que podría crear tensiones en el equipo de desarrollo.

2- Regulaciones de privacidad y seguridad en constante evolución:

En un entorno donde la aplicación debe cumplir con regulaciones de privacidad y seguridad en constante cambio, la prioridad en la entrega de nuevas funcionalidades sobre la documentación extensiva podría entrar en conflicto con la necesidad de mantener una documentación detallada para garantizar la seguridad y la conformidad.

3- Rotación constante de usuarios y fluctuación en la popularidad de destinos turísticos:

Si la base de usuarios de la aplicación cambia constantemente debido a la rotación de turistas o a fluctuaciones en la popularidad de los destinos turísticos, la formación de comunidades estables y la autoorganización de grupos de viajeros podrían verse afectadas.

4- Presión de tiempo para aprovechar la temporada turística:

En proyectos donde se necesiten actualizaciones constantes la aplicación debe aprovechar al máximo las temporadas turísticas, la presión de cumplir con plazos ajustados podría entrar en conflicto con la entrega continua, ya que la calidad y la funcionalidad podrían verse comprometidas**.**

Falta de apoyo de patrocinadores clave o entidades turísticas locales:

La falta de comprensión o respaldo de patrocinadores clave o entidades turísticas locales podría generar conflictos en la implementación de prácticas ágiles**.**

Referencias

IBM (s. f.). *¿Qué es la gestión de requisitos?* Recuperado 16 de noviembre de 2023, de https://www.ibm.com/mx-es/topics/what-is-requirements-management

Tiffin University (2023). *Qué es la gestión de proyectos de software?* Tiffin University. https://global.tiffin.edu/noticias/en-que-consiste-la-gestion-de-proyectos-de-software#:~:text=La%20gesti%C3%B3n%20de%20proyectos%20de%20software%20consiste%20en%20una%20serie,que%20surja%20en%20el%20proceso

Red Hat (s.f) *Seguridad en el ciclo de vida de desarrollo del software*. https://www.redhat.com/es/topics/security/software-development-lifecycle-security

Pressman, R. S. (2010). *Ingeniería del Software. Un enfoque práctico* (7.a ed.). McGRAW-HILL INTERAMERICANA EDITORES, S.A. DE C.V. http://www.javier8a.com/itc/bd1/ld-Ingenieria.de.software.enfoque.practico.7ed.Pressman.PDF

Ablancodev. (2022, 16 julio). *Historias de usuario vs casos de uso*. Antonio Blanco Oliva. https://ablancodev.com/gestion-de-proyectos/historias-de-usuario-vs-casos-de-uso/

Beck, K., Beedle, M., Bennekum, A. van, Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., & Thomas, D. (2001). *Manifesto for Agile Software Development*. Manifesto for Agile Software Development. https://agilemanifesto.org

Cesar. (2020, 15 mayo). *Historias de usuario vs. casos de uso*. gegolabs. https://gegolabs.org/2020/05/15/historias-de-usuario-vs-casos-de-uso/

Continuous Integration. (2023), Amazon Web Services. https://aws.amazon.com/es/devops/continuous-integration/#:~:text=La%20integraci%C3%B3n%20continua%20es%20una,ejecutan%20versiones%20y%20pruebas%20autom%C3%A1ticas.

De Sá Araújo, C. M. M., Santos, I. M., Canedo, E. D., & Araújo, A. (2019). Design Thinking versus Design Sprint: A comparative study. En *Lecture Notes in Computer Science* (pp. 291-306). https://doi.org/10.1007/978-3-030-23570-3_22

Juan. (2023, January 9). *¿Qué habilidades necesitas para ser Ingeniero de Software? | COHETE.digital*. COHETE.Digital. https://cohete.digital/guia-carrera-profesional/ingeniero-de-software/habilidades/

Letelier, P. (2006, 15 enero). Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP). *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP)*. http://www.cyta.com.ar/ta0502/b_v5n2a1.htm

Riccardi, Y., Vega, R., Miyares, E. (2018). Aplicación del Responsive Web Design en la creación e implementación del sitio Web del Centro de Histoterapia Placentaria. *Scielo, 10*(1), 16-27. http://scielo.sld.cu/scielo.php?pid=S1684-18592018000100003&script=sci_arttext&tlng=

Sánchez, A. L. L. (2016, 9 noviembre). *Requisitos vs casos de uso vs historias de usuario - Angel Lozano*. Angel Lozano. http://www.angellozano.com/requisitos-del-sistema-vs-casos-uso-vs-historias-usuario/