**Metodología de Trabajo como Estrategia de Desarrollo de Software Educativo**

**Introducción**

En el marco del desarrollo del curso diseño de software III se presenta un enfoque innovador en la concepción y diseño de un componente de mensajería pensado para optimizar la interacción entre usuarios, sistemas automatizados y soporte técnico. La propuesta integra tecnologías de inteligencia artificial, representadas por un chat-bot capaz de responder de manera eficiente a consultas frecuentes, y servicios personalizados de atención al cliente, asegurando un equilibrio entre automatización y soporte humano. El artículo detalla exhaustivamente las etapas de diseño, desarrollo e implementación del sistema, utilizando metodologías como Design Thinking para garantizar un enfoque centrado en el usuario. Asimismo, explora la arquitectura técnica, desde el diseño de bases de datos con MongoDB Atlas hasta la creación de una interfaz de usuario intuitiva. Estas herramientas y técnicas no solo mejoran la experiencia del usuario, sino que también establecen bases sólidas para la escalabilidad y evolución futura del sistema.

Con un desarrollo estructurado en tres etapas, el proyecto demuestra cómo integrar eficazmente tecnologías modernas para crear un sistema robusto y versátil, posicionándolo como un modelo a seguir en el diseño de soluciones de comunicación digital.

**Metodologías Aplicadas**

**1. Trabajo en Equipo y SCRUM**

SCRUM, como marco ágil, facilita la organización y ejecución de proyectos de desarrollo de software educativo.

La metodología Scrum ha sido fundamental en nuestro proyecto de desarrollo de software, específicamente en la creación de un componente de mensajería que optimiza la interacción entre usuarios y sistemas automatizados, esta metodología nos ha ayudado a organizar el trabajo de manera efectiva. Al dividir el proyecto en sprints, logramos enfocarnos en tareas específicas y obtener resultados rápidos. Cada miembro del equipo tiene un rol claro: el Product Owner, el Scrum Master y el equipo de desarrollo. Este enfoque ágil ha permitido estructurar nuestro trabajo de manera eficiente, facilitando la colaboración mutua en la construcción y desarrollo de nuestro proyecto.

"El marco de trabajo Scrum facilita la actividad coordinada de los programadores que dividen su trabajo en pequeñas tareas que pueden completarse dentro de ciclos de duración fija o 'sprints', rastreando el progreso y replanificando en reuniones regulares para desarrollar productos de manera incremental" (Lei et al., 2017).

Las reuniones diarias nos permiten mantener la comunicación y resolver problemas de inmediato, mientras que las revisiones al final de cada sprint nos ayudan a recibir retroalimentación y hacer ajustes necesarios. Además, las retrospectivas nos permiten reflexionar sobre nuestro trabajo y mejorar continuamente. En resumen, Scrum ha mejorado nuestra colaboración, adaptabilidad y eficiencia en la entrega del software educativo.

**Roles y Dinámicas**

Entre los roles se encuentran Product Owner, el cual es el encargado de definir los requerimientos educativos, luego sigue Scrum Master, el cual garantiza el cumplimiento de las prácticas ágiles y por ultimo el equipo de Desarrollo, los cuales se encargan de diseñar, implementar y probar el software.

**Ciclos de Trabajo**

Se estructuran sprints, donde el equipo desarrolla módulos educativos incrementales. Cada sprint incluye planificación, la cual constituye la Identificación de funcionalidades, luego se encuentra ejecución, en esta se lleva a cabo la Implementación del módulo y por último la revisión y retrospectiva, en esta se desarrolla la evaluación y ajuste de procesos.

**2. Aula Invertida**

El aula invertida fomenta el aprendizaje autónomo antes de la implementación en equipo. Los estudiantes revisan conceptos teóricos sobre programación y metodologías de desarrollo en casa y los aplican en sesiones colaborativas. Entre los beneficios que esta ofrece se destaca el aprovechamiento del tiempo en clase para resolver problemas prácticos, así como mayor interacción y retroalimentación.

**3. Creación de Componentes**

Los proyectos se dividen en módulos o componentes específicos (e.g., chatbots educativos, sistemas de retroalimentación). Cada equipo trabaja en un componente, promoviendo especialización y eficiencia. Por ejemplo,en un software educativo sobre matemáticas, un equipo desarrolla el motor de evaluación automática mientras otro se encarga del diseño de la interfaz interactiva.

**4. Programación en Paralelo**

Se forman dos equipos que desarrollan la misma funcionalidad de manera independiente. Al final, se comparan las soluciones para seleccionar la más eficiente y documentar mejores prácticas. Entre las ventajas que esta ofrece se destaca la Identificación de enfoques óptimos y la diversidad de soluciones.

**5. Programación en Pares**

Dos desarrolladores trabajan en una misma tarea, alternando roles como el driver, el cual es el encargado de escribir el código mientras que el observer, es el que se encarga de revisar el trabajo en tiempo real.

La programación en pares favorece la reducción de errores, mejora de la calidad del código y fortalece el aprendizaje colaborativo.

**6. Trabajo Independiente**

Cada integrante tiene asignaciones individuales que aportan al proyecto general, fomentando la autonomía y el desarrollo de habilidades personales.

**Implementación en Proyectos Educativos**

Estas metodologías se integran para maximizar el aprendizaje y la productividad. Por ejemplo, en un curso de introducción a la programación, los estudiantes desarrollan una aplicación educativa donde SCRUM organiza el trabajo, el aula invertida introduce conceptos teóricos, y la programación en pares garantiza la calidad del código.

**Resultados Esperados**

Se espera mejorar el aprendizaje, que los estudiantes entiendan los conceptos complejos a través de la práctica guiada, desarrollando habilidades de resolución de problemas y pensamiento critico por medio del trabajo en equipo, Mejor aprendizaje: Los estudiantes entienden conceptos complejos a través de práctica guiada. Así mismo se espera un software funcional como producto educativo de alta calidad listo para implementarse en entornos reales.

**Software I Diseño de la aplicación y análisis de requisitos**

Esta etapa fue crucial para sentar las bases del desarrollo del software, ya que permitió establecer una visión clara y estructurada de lo que sería el componente de mensajería. Se inició con el análisis de las necesidades del sistema, identificando las funcionalidades principales que serían implementadas: enviar y recibir mensajes en tiempo real, reconocer intenciones de los usuarios mediante un chat-bot con inteligencia artificial, y facilitar la interacción con soporte técnico humano. A través de la metodología Design Thinking, se desglosaron las ideas en módulos y casos de uso detallados, lo que permitió definir flujos de interacción y prioridades. Se crearon diagramas de flujo para visualizar estos procesos, y se clasificaron los requisitos según su urgencia y esfuerzo, asegurando que se atendieran primero los elementos más críticos. Esta planificación detallada permitió no solo minimizar posibles errores en etapas posteriores, sino también garantizar que las funcionalidades estuvieran alineadas con las expectativas de los usuarios. Además, se diseñó un modelo de casos de uso que ayudó a prever cómo interactuarían los usuarios con el sistema, mejorando la comprensión del alcance del proyecto. Este enfoque estructurado no solo proporcionó una base sólida para el desarrollo técnico, sino que también sentó las bases para la eficiencia y la escalabilidad del componente.

**Software II Persistencia de datos con backend**  
En esta etapa, el enfoque principal fue diseñar e implementar la infraestructura que garantizaría el almacenamiento, manejo y recuperación de los datos necesarios para el funcionamiento del sistema de mensajería. Se optó por MongoDB Atlas como base de datos, aprovechando su flexibilidad en esquemas y su capacidad para manejar grandes volúmenes de datos no estructurados. Se diseñaron colecciones específicas para usuarios, mensajes, intenciones y textos asociados, estableciendo relaciones claras mediante claves foráneas y referencias, lo que permitió organizar la información de forma eficiente. A partir de esto, se desarrollaron las operaciones CRUD (crear, leer, actualizar y eliminar) para cada colección, asegurando que los datos fueran fácilmente accesibles y manipulables. La lógica de negocio se implementó usando Node.js con Express.js, permitiendo una integración fluida con la base de datos y garantizando la escalabilidad del sistema. Se desarrollaron endpoints y APIs que fueron rigurosamente probados con herramientas como Postman, lo cual aseguró que todas las interacciones entre el frontend y el backend fueran precisas y seguras. Además, se implementaron medidas de seguridad para proteger los datos, como autenticación y manejo de errores, y se optimizaron las operaciones para garantizar un rendimiento eficiente. Esta etapa fue esencial para proporcionar un backend robusto que soportara las operaciones del sistema y asegurara la continuidad y confiabilidad del componente de mensajería.

**Software III Consumo de datos y desarrollo frontend**La última etapa del desarrollo se centró en crear una interfaz de usuario amigable e intuitiva que permitiera a los usuarios aprovechar al máximo las funcionalidades del sistema de mensajería. Se utilizó HTML y CSS para diseñar una interfaz responsiva que se adaptara a diferentes dispositivos, garantizando una experiencia consistente y accesible para todos los usuarios. Además, se empleó JavaScript para desarrollar la lógica que manejara eventos dinámicos, como el envío y recepción de mensajes en tiempo real, notificaciones y validación de formularios. La conexión con el backend se configuró mediante APIs, lo que permitió una comunicación eficiente entre la interfaz y el sistema central. Se crearon pantallas específicas para que los usuarios interactuaran con el chat-bot, otros usuarios y el soporte técnico, asegurando que las funciones fueran fáciles de usar y estuvieran claramente identificadas. Se llevaron a cabo pruebas exhaustivas de usabilidad y depuración para garantizar que la interfaz fuera intuitiva y que todas las funciones operaran sin inconvenientes. Esta etapa no solo mejoró la interacción de los usuarios con el sistema, sino que también consolidó el objetivo general del software al proporcionar una experiencia de usuario fluida y funcional. El trabajo en el frontend, en conjunto con las etapas previas, aseguró que el sistema de mensajería cumpliera con los estándares de calidad y las expectativas de los usuarios finales, contribuyendo significativamente al éxito del proyecto.

**Conclusión**

La combinación de metodologías ágiles y estrategias pedagógicas potencia tanto el aprendizaje de los estudiantes como el desarrollo de herramientas educativas innovadoras. Este enfoque integrador representa un avance significativo en la enseñanza de la programación y el diseño de software.

**Referencias**

Beck, K., et al. (2001). *Manifesto for Agile Software Development.*

Bergmann, J., & Sams, A. (2012). *Flip Your Classroom: Reach Every Student in Every Class Every Day.*

Lei, Y., et al. (2017). Adapting the scrum framework for agile project management in science. *PMC*. Recuperado de <https://pmc.ncbi.nlm.nih.gov/articles/PMC6441834/>