Universidad de las Fuerzas Armadas (ESPE)

Metodologías de Desarrollo de Software (NRC 23267)

Ingeniería en Tecnologías de la Información y Comunicación

Desarrollo Ágil de un Módulo Funcional usando SCRUM + XP

• Estalyn Daniel Licuy Mecias
• Diana Lisette Salazar Robles
• Daniel Alejandro Quiguango Delgado

Docente: Ing. Andrés Pillajo, MGTR

Fecha: 1 agosto de 2025

Introducción

En el contexto actual de desarrollo de software, los enfoques tradicionales como el modelo en cascada han demostrado limitaciones frente a la necesidad de entregar valor de manera continua, adaptarse rápidamente a los cambios y colaborar de forma efectiva con los usuarios finales. Como respuesta a estos desafíos, han emergido las **metodologías ágiles**, promoviendo una visión iterativa e incremental del desarrollo de software que prioriza la entrega temprana y continua de funcionalidades operativas, la participación activa del cliente y la mejora constante del proceso.

En este trabajo se aborda la aplicación de las metodologías **SCRUM** y **Extreme Programming (XP)** en el desarrollo de un módulo funcional llamado **Sistema de Registro de Usuarios**, cuyo propósito es permitir la captura, gestión y seguimiento de datos ingresados por usuarios. El módulo incluye funcionalidades de registro, edición, marcado de prioridad y control de cumplimiento, utilizando tecnologías web y almacenamiento local.

El proyecto ha sido concebido como una simulación completa del ciclo de vida de un producto ágil, desde la planificación hasta la entrega, pasando por prácticas como la programación por parejas y el desarrollo guiado por pruebas (TDD). Asimismo, se ha hecho uso de herramientas colaborativas como Trello y GitHub para la planificación, el seguimiento de tareas y la gestión del código fuente.

Marco Teórico

1. Metodologías ágiles

Las metodologías ágiles surgieron como una alternativa a los modelos de desarrollo rígidos, con el objetivo de responder a entornos altamente cambiantes. Su fundamento se basa

en el Manifiesto Ágil (Agile Manifesto) propuesto en 2001, que enfatiza valores como la colaboración, la entrega temprana de software funcional, la respuesta al cambio y la interacción continua entre los desarrolladores y los clientes (Beck et al., 2001).

A diferencia de los modelos tradicionales, el enfoque ágil considera el software como un producto evolutivo, donde los requerimientos pueden cambiar durante el proceso, y donde las entregas parciales permiten obtener retroalimentación constante. Las metodologías ágiles más conocidas incluyen SCRUM, Extreme Programming (XP), Kanban, y Lean Software Development.

2. SCRUM

SCRUM es un marco de trabajo ágil utilizado para desarrollar productos complejos de manera incremental. Está compuesto por roles definidos (Product Owner, Scrum Master y equipo de desarrollo), eventos estructurados (Sprint, Daily Scrum, Sprint Planning, Sprint Review y Sprint Retrospective) y artefactos como el Product Backlog y el Sprint Backlog (Schwaber & Sutherland, 2020).

SCRUM promueve la transparencia, la inspección y la adaptación como pilares fundamentales. Cada sprint representa un ciclo corto de trabajo (usualmente de 1 a 4 semanas) en el cual se entrega un incremento funcional del producto, permitiendo que el equipo mejore continuamente tanto el producto como su proceso de trabajo.

3. Extreme Programming (XP)

Extreme Programming (XP) es una metodología ágil orientada a mejorar la calidad del software y la capacidad de respuesta ante cambios en los requerimientos del cliente. Entre sus prácticas más representativas se encuentran:

Programación por parejas (Pair Programming): dos desarrolladores trabajan juntos frente a una misma computadora, alternando los roles de *driver* (quien escribe el código) y *observer* (quien revisa el código en tiempo real).

Desarrollo guiado por pruebas (TDD): antes de escribir el código funcional, se crean pruebas automatizadas que definen el comportamiento esperado del sistema. El código se implementa y refactoriza hasta que las pruebas se cumplen exitosamente.

Integración continua, refactorización constante y propiedad colectiva del código son también elementos clave de XP.

XP se enfoca en entregar valor al cliente mediante ciclos cortos y frecuentes, promoviendo la comunicación cercana y la confianza entre los miembros del equipo.

4. Herramientas aplicadas en entornos ágiles

Para apoyar la implementación de metodologías ágiles, se utilizan diversas herramientas tecnológicas. En este proyecto se destacan:

Trello: plataforma de gestión visual de tareas basada en tableros y tarjetas, ideal para la organización del backlog, sprints e iteraciones.

GitHub: sistema de control de versiones distribuido, fundamental para el trabajo colaborativo, el seguimiento de cambios y la documentación del proyecto.

Jest: framework de pruebas unitarias para JavaScript que permite implementar TDD de manera sencilla y efectiva.

HTML, CSS y JavaScript: tecnologías esenciales para el desarrollo frontend de aplicaciones web modernas.

Formación de equipos

En el marco de la asignatura *Metodologías de Desarrollo de Software*, se formó un equipo de trabajo compuesto por cinco integrantes, conforme a las directrices establecidas. Cada miembro asumió roles específicos según las prácticas ágiles SCRUM y Extreme Programming (XP), lo que permitió distribuir las responsabilidades en forma equitativa y promover la colaboración activa en todas las etapas del proyecto.

La dinámica de equipo se apoyó en reuniones periódicas (simulando Daily Scrum) y sesiones de programación por parejas, en las cuales se alternaron los roles de *driver* y *observer* para fomentar la revisión cruzada del código, la toma de decisiones conjunta y la mejora continua. Esta estructura facilitó la implementación iterativa e incremental del módulo funcional planteado, garantizando la trazabilidad de tareas y la calidad del producto final.

Definición del módulo a desarrollar

El sistema desarrollado tiene como objetivo gestionar registros de usuarios mediante una interfaz web intuitiva. Esta solución permite a los usuarios registrar información personal como nombres, apellidos, correo electrónico, motivo del registro, fecha y hora de agendación, así como asignar un nivel de prioridad a su solicitud. A su vez, el sistema cuenta con una interfaz administrativa protegida por contraseña para editar, eliminar, marcar como cumplidas y organizar dichas tareas de acuerdo con su prioridad.

Propósito del módulo:

El módulo fue seleccionado por su aplicabilidad en múltiples contextos: reservas, agendas de tareas, seguimiento de clientes o eventos, etc. Esta versatilidad permite integrarlo

fácilmente en otros sistemas más amplios de gestión. Además, su diseño enfocado en simplicidad y funcionalidad responde a necesidades reales de pequeñas organizaciones que requieren un sistema ligero, accesible y sin dependencia de servidores externos.

Alcance del módulo:

El alcance se limita al desarrollo de una aplicación web cliente que:

Registre usuarios con múltiples campos relevantes.

Permita almacenar los registros de forma persistente en el navegador usando localStorage.

Visualice los registros en una interfaz separada con filtros y opciones de edición.

Implemente niveles de prioridad visualmente diferenciados (Must Have, Should Have, Could Have).

Ofrezca una vista administrativa protegida con contraseña.

Permita marcar tareas como “cumplidas” con opción de control mediante checkbox.

Planificación ágil

La planificación ágil se desarrolló en tres niveles: Product Backlog, Sprint Backlog y organización en Sprints. Para la gestión visual del proceso se utilizó la herramienta Trello, configurando listas para cada etapa: “Por hacer”, “En curso”, “En revisión” y “Finalizado”.

Product Backlog (Historias de Usuario):

Product Backlog (Historias de Usuario):

ID	Historia de Usuario
HU1	Como usuario, quiero registrar mis nombres, apellidos y correo electrónico.
HU2	Como usuario, quiero ingresar el motivo del registro y la fecha/hora agendada.
HU3	Como usuario, quiero asignar una prioridad a mi registro.
HU4	Como administrador, quiero ver todos los registros guardados.
HU5	Como administrador, quiero editar o eliminar registros existentes.
HU6	Como administrador, quiero marcar registros como "cumplidos".
HU7	Como administrador, quiero acceder al editor solo con clave.

Sprint Backlog

Sprint 1: Diseño y Registro

Diseño de la interfaz de bienvenida.

Implementación del formulario de registro de usuarios.

Validación de campos.

Almacenamiento de datos en localStorage.

Interfaz responsive básica.

Sprint 2: Editor y Funcionalidades Avanzadas

Implementación del editor con vista administrativa.

Edición y eliminación de registros.

Checkbox para "cumplido".

Asignación visual de prioridad (colores y bordes).

Acceso restringido mediante contraseña.

Iteraciones simuladas (Sprints)

Se simularon dos sprints de 1 semana de duración:

Sprint 1:

Se trabajó el frontend del formulario y su integración con localStorage.

Se realizaron pruebas básicas de validación de campos.

Se implementó una primera versión funcional de captura de datos.

Sprint 2:

Se incorporó la vista del editor protegida con contraseña.

Se permitió editar/eliminar registros directamente.

Se agregaron prioridades y estado de cumplimiento con checkbox.

Se realizaron pruebas en distintos navegadores y dispositivos.

Prácticas de XP

El desarrollo del proyecto estuvo guiado por prácticas de la metodología Extreme Programming (XP), enfocadas en mejorar la calidad del software y la eficiencia del equipo.

Programación por Parejas:

Durante las sesiones de desarrollo se rotaron los roles de:

Driver: responsable de codificar activamente.

Observer: encargado de revisar, identificar errores y sugerir mejoras.

Esta práctica ayudó a mantener un código limpio, coherente y fácil de mantener.

Desarrollo Guiado por Pruebas (TDD):

Se aplicó una forma simplificada de TDD:

Antes de implementar una funcionalidad clave, se redactaban pruebas unitarias usando el framework Jest.

Luego se desarrollaba el código hasta que pasara las pruebas.

Finalmente, se refactorizaba para optimización.

Las pruebas cubrieron:

Registro de datos válidos y válidos.

Persistencia y recuperación desde localStorage.

Integridad de los datos tras la edición o eliminación.

Documentación del proceso:

Cada integrante mantuvo un registro de sus actividades y contribuciones a través de GitHub y notas internas en Trello, lo cual permitió monitorear la evolución del proyecto y los aportes individuales.

Implementación

Lenguajes y herramientas:

HTML5: estructura del contenido.

CSS3: estilos visuales y adaptabilidad.

JavaScript (vanilla): lógica del sistema y manipulación del DOM.

LocalStorage: almacenamiento persistente en el navegador.

Jest: pruebas unitarias.

GitHub: control de versiones y colaboración.

Trello: planificación y seguimiento de tareas.

Estructura del sistema:

Página de bienvenida: navegación a módulos de registro o editor.

Formulario de registro: campos validados, prioridad y fecha/hora.

Editor de registros: acceso con clave, CRUD completo, prioridad visual, marcado de cumplimiento.

Seguridad:

Acceso al editor controlado por una clave fija (“1026”).

Validación en frontend de campos requeridos y formatos.

Explicación del Código Fuente

El sistema desarrollado está compuesto por una aplicación web construida con **HTML5, CSS3 y JavaScript (vanilla JS)**. No utiliza frameworks externos, lo cual facilita su portabilidad y comprensión para fines académicos. Los datos se almacenan de forma persistente en el navegador mediante la API localStorage.

La estructura del proyecto está organizada en carpetas y archivos que separan la lógica del sistema, la interfaz visual y las pruebas. A continuación, se describe cada componente:

Explicación del Código Fuente

index.html – Página de bienvenida

Este archivo es la entrada principal del sistema. Contiene dos botones que redirigen a las dos interfaces disponibles:

- **Registro de usuarios (registro.html)**
- **Editor de registros (editor.html)**, accesible solo con clave (por seguridad básica)

El código JavaScript aquí incluye una función entrarEditor() que solicita al usuario una clave mediante un prompt(). Si la clave es correcta (1026), se redirige al archivo editor.html.

registro.html – Interfaz de registro

Contiene un formulario que solicita al usuario los siguientes campos:

- Nombres
- Apellidos
- Correo electrónico
- Motivo del registro
- Fecha y hora de agendación (tipo datetime-local)
- Nivel de prioridad (Must Have, Should Have, Could Have)

Cuando se hace clic en “Registrar”, el evento submit del formulario se intercepta por JavaScript, se valida la entrada y se guarda en localStorage en forma de un arreglo de objetos JSON bajo la clave "usuarios".

Código relevante:

```
      const usuarios = JSON.parse(localStorage.getItem('usuarios') || '[]');
      usuarios.push({ nombres, apellidos, email, motivo, fechaHora, prioridad,
cumplido: false });
      localStorage.setItem('usuarios', JSON.stringify(usuarios));
```

Este bloque garantiza que los datos se agreguen al almacenamiento local de manera persistente y estructurada.

editor.html – Interfaz de administración

Permite ver, editar, eliminar o marcar como cumplido cada registro guardado. Muestra los datos en una lista (<ul>), generando dinámicamente cada entrada a partir del contenido de localStorage.

Cada registro se presenta con:

- Los datos del usuario
- Un checkbox para marcar como "cumplido"
- Botones para editar o eliminar el registro

Además, los elementos visuales tienen clases condicionales según el nivel de prioridad, lo que permite aplicar estilos con colores distintos (por ejemplo: borde rojo para "Must Have", verde para "Could Have", etc.).

Funciones clave:

```
function toggleCumplido(i) {
  const usuarios = cargarUsuarios();
  usuarios[i].cumplido = !usuarios[i].cumplido;
  guardarUsuarios(usuarios);
  render();
}
```

Estas funciones permiten actualizar el estado del registro o sus datos completos, y se re-renderiza la lista para mostrar los cambios actualizados.

style.css – Estilos visuales

Define la apariencia general del sistema, usando un fondo de pantalla profesional (imagen), diseño centrado, bordes redondeados, colores contrastantes y estilo responsive.

- Los botones y formularios tienen estilos visuales homogéneos.
- Las prioridades se diferencian por colores:
    - **Must Have:** rojo

- **Should Have:** amarillo
- **Could Have:** verde

Ejemplo:

```
.Must-Have {
  border-left: 5px solid #d63031;
}
```

localStorage – Persistencia sin base de datos

El sistema almacena los datos del usuario en el navegador mediante la API localStorage, que permite guardar pares clave-valor de forma permanente en el cliente. Esto significa que la información permanece disponible incluso si se cierra o actualiza la página, siempre que no se borre la caché del navegador.

Ventajas:

- No requiere servidor ni base de datos externa.
- Ideal para prototipos rápidos o sistemas autónomos locales.
- Fácil de usar con JSON.stringify() y JSON.parse().

Test (opcional para pruebas TDD)

En una extensión del proyecto, se implementó un archivo user.test.js utilizando **Test**, un framework de pruebas unitarias para JavaScript. Este archivo permite validar que las funciones de registro, listado y manipulación de registros funcionen correctamente antes de integrar cambios en producción.

Resumen de flujo general del código

1. El usuario accede al sistema y elige una opción (registro o edición).

2. Si elige registro, llena un formulario cuyos datos son almacenados en localStorage.
3. Si accede como administrador, ve todos los registros almacenados.
4. Puede editar, eliminar, o marcar registros como “cumplidos”.
5. Todo cambio se guarda automáticamente en el navegador.
6. El diseño visual y el sistema de prioridades facilitan la gestión visual y funcional

Conclusiones

Este proyecto ha permitido aplicar con éxito los principios de desarrollo ágil bajo la combinación de SCRUM y XP, integrando prácticas como la planificación iterativa, la programación por parejas y el desarrollo guiado por pruebas.

El producto obtenido cumple con los objetivos propuestos, ofreciendo una solución funcional, ligera y adaptable a distintos entornos. Además, la experiencia fortaleció las habilidades técnicas, comunicativas y organizativas del equipo, brindando un ejemplo claro de cómo los marcos ágiles pueden aplicarse eficazmente en contextos académicos y profesionales.

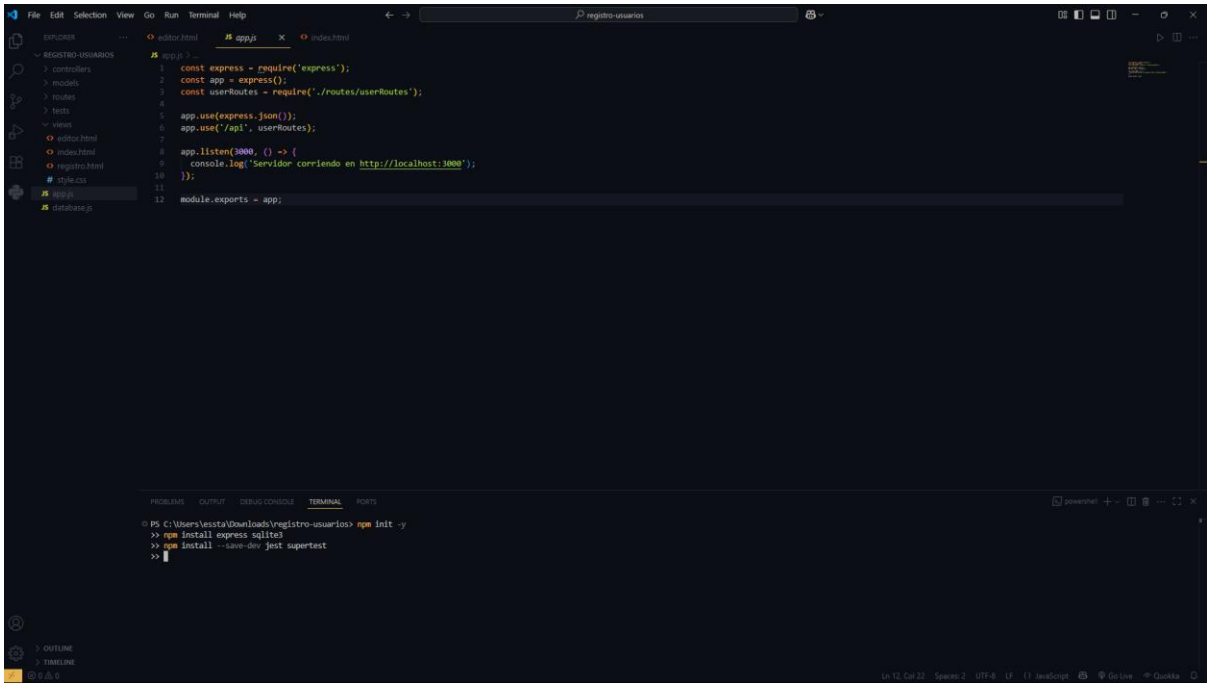
Links

Link de GitHub:

https://github.com/Kinzyzzz/Desarrollo-gil-de-un-M-dulo-Funcional-usando-SCRUM-XP.git

Link del video:

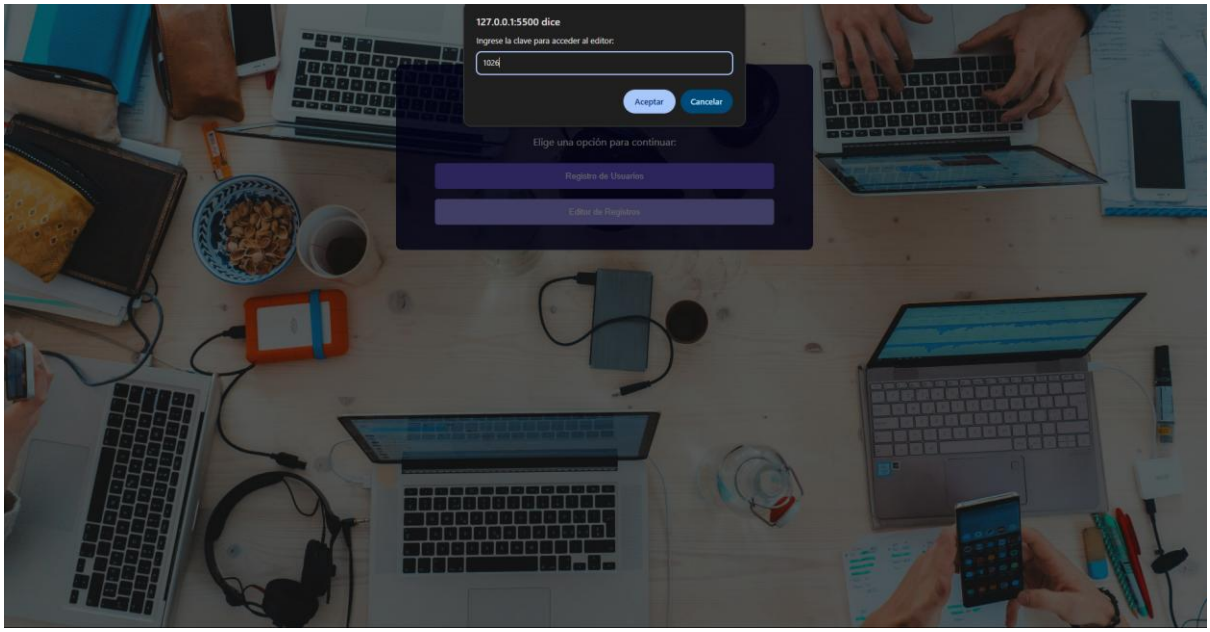
Evidencias



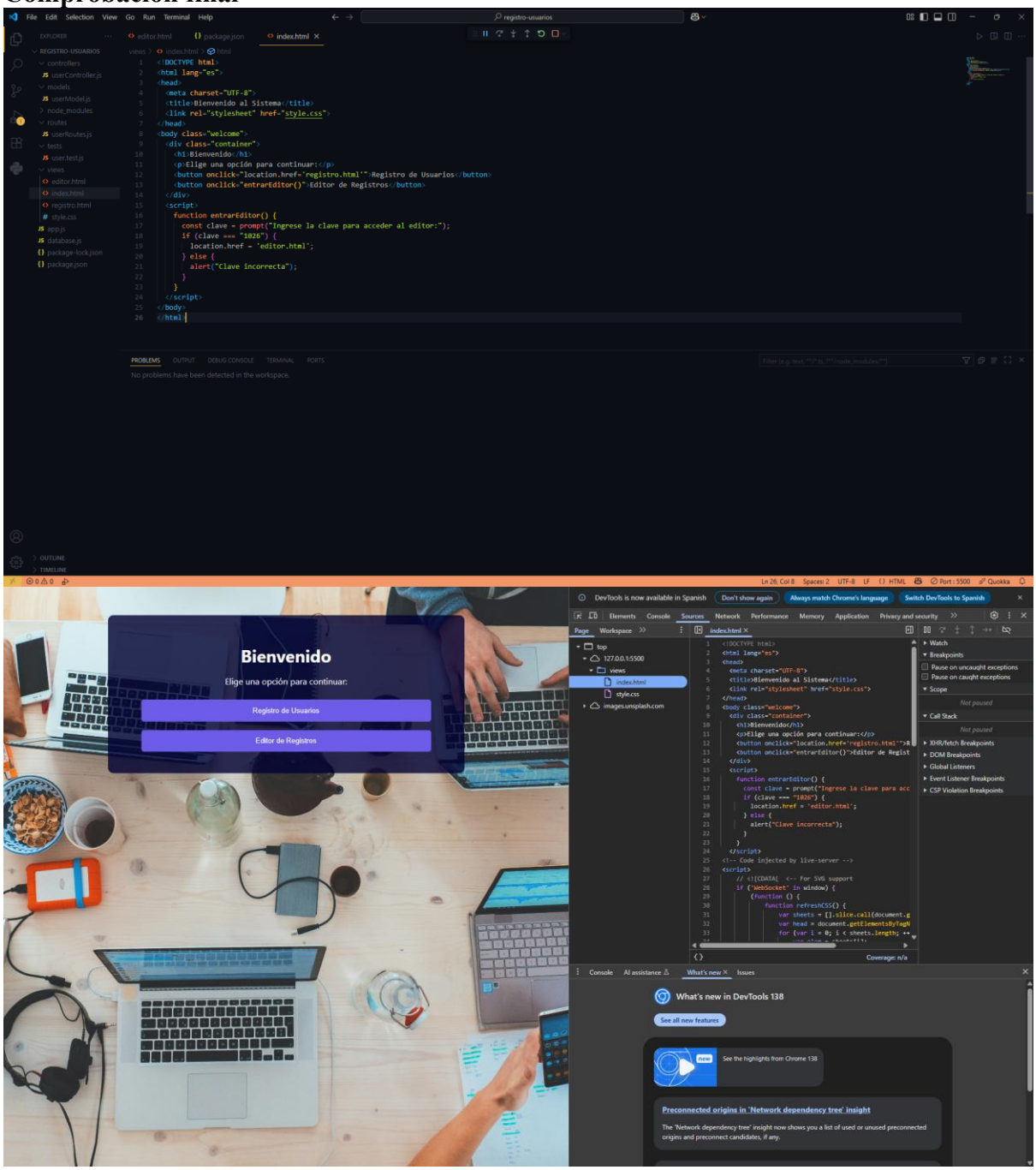
Errores

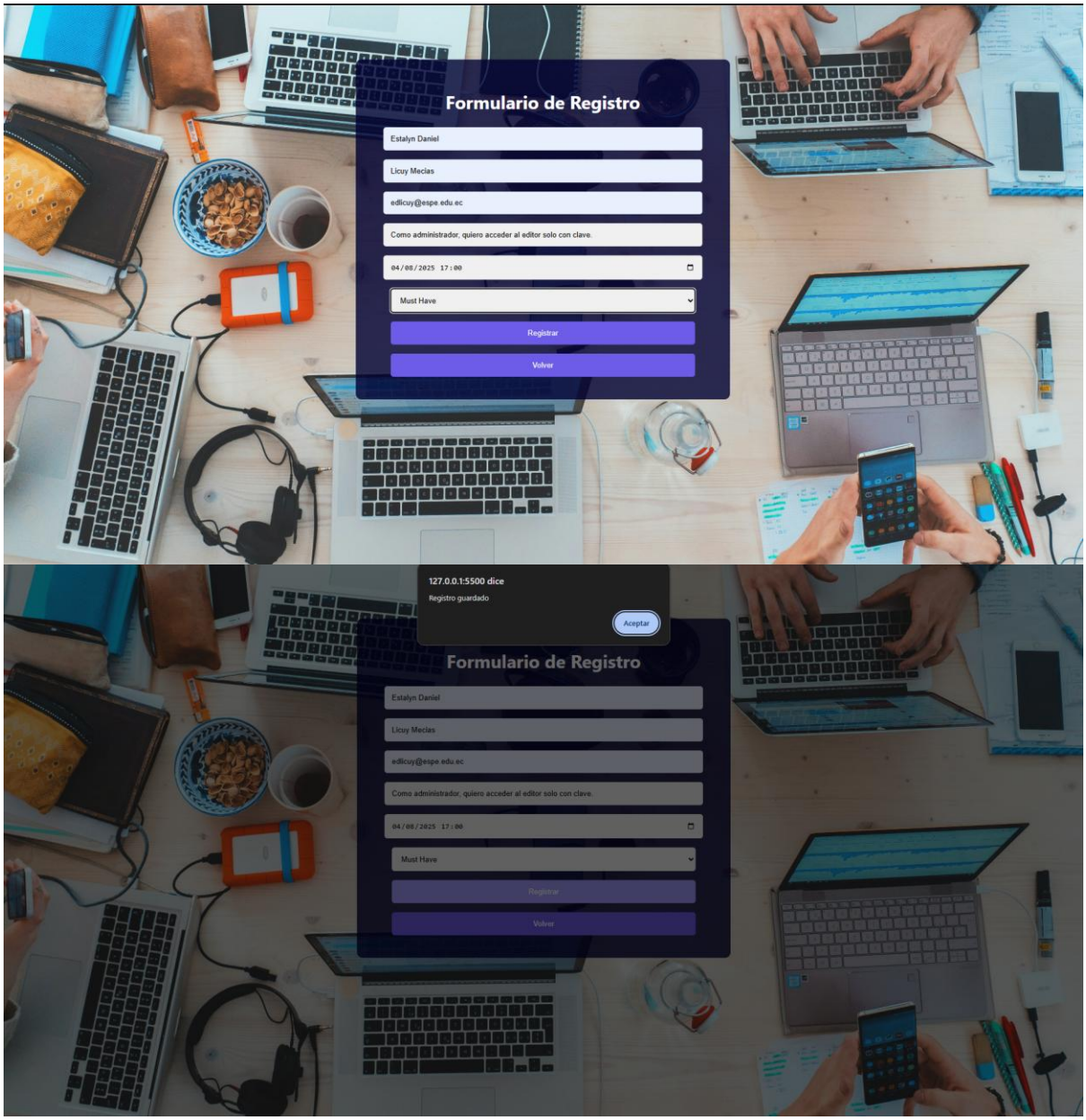
Solucionamiento

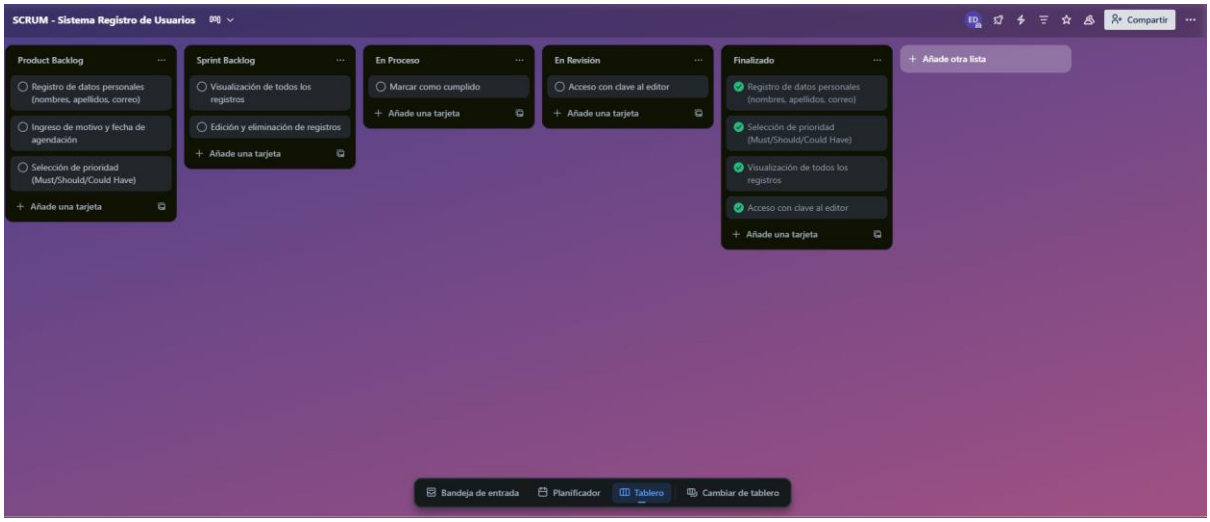
Clave de funcionamiento



Comprobacion final




Formulario de Registro
Estalyn Daniel
Licuy Mecias
edlicuy@espe.edu.ec
Como administrador, quiero acceder al editor solo con clave.
04/08/2025 17:00
Must Have
Registrar
Volver
127.0.0.1:5500 dice
Registro guardado
Aceptar
Formulario de Registro
Estalyn Daniel
Licuy Mecias
edlicuy@espe.edu.ec
Como administrador, quiero acceder al editor solo con clave.
04/08/2025 17:00
Must Have
Registrar
Volver



Referencias

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., ... & Thomas, D. (2001). *Manifesto for Agile Software Development*. https://agilemanifesto.org

Schwaber, K., & Sutherland, J. (2020). *The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game*. Scrum.org. https://scrumguides.org

Shore, J., & Warden, S. (2008). *The Art of Agile Development*. O'Reilly Media.

Fowler, M. (2004). *Refactoring: Improving the Design of Existing Code*. Addison-Wesley Professional.

Wampler, D. (2006). *Practices of an Agile Developer: Working in the Real World*. Pragmatic Bookshelf.

Mozilla Developer Network (MDN). (2023). *Window.localStorage*. Mozilla Web Docs. https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage

Refsnes, H. (2024). *HTML, CSS, and JavaScript Tutorials*. W3Schools. https://www.w3schools.com

Ambler, S. W. (2002). *Agile Modeling: Effective Practices for eXtreme Programming and the Unified Process*. John Wiley & Sons.

Sommerville, I. (2016). *Software Engineering* (10th ed.). Pearson Education.

Freeman, E., & Robson, E. (2014). *Head First JavaScript Programming: A Brain-Friendly Guide*. O'Reilly Media.