Programación 3

Docente: Kelyn Tejada Belliard

Ultimo Modulo: Proyecto Final

Tarea: Proyecto Final

Alexander Antonio Brito Soto(20230645)

Fecha de entrega:12/10/2025

Índice

1. Elabore una estrategia de trabajo donde pueda planificar los siguientes puntos

1. Nombre un proyecto de software.
2. Tecnología para aplicar.
3. Objetivo del proyecto.
4. Alcance del proyecto.
5. Cronograma del proyecto.
6. Definir claramente lo que el sistema en un primer Release va a poder hacer.
7. Definir requerimientos del sistema para el primer Release.

2. Metodología Scrum:

1. Definir tareas a ejecutar
2. Definir el equipo de trabajo (roles, habilidades, etc)
3. Herramientas que usarían
4. Definir las épicas

3. Acciones:

1. Definir las fechas para cada ceremonia de Scrum.
2. Afinar al menos 10 historias de usuarios (criterios de aceptación obligatorios).
3. Asignar los puntos de historia

4. Plan de Pruebas

1- Lista de requerimientos funcionales y no funcionales de acorde a las historias de usuarios. (mínimo de 10 HU).
2- Definir cuáles son los criterios de aceptación de las pruebas.
3- Definir cuáles son los criterios de rechazo en las pruebas.
4- Comentar sobre las herramientas de pruebas que estarían usando y justificar respuesta.
5- Estimar el tiempo que duraría la ejecución de pruebas. (elaborar cronograma de trabajo).
6- Elaborar plantillas para cada caso de pruebas.
7- Elaborar una plantilla con los equipos de pruebas y sus responsabilidades.
8- Elaborar un plan de automatización de pruebas.
9- Ejecutar y demostrar el plan de automación de pruebas.

1.1 Nombre un proyecto de software.

Blz Suite

"Blz Suite" es un nombre que combina varios conceptos importantes del proyecto:

Blz: Hace referencia a la tecnología principal utilizada en el front-end del proyecto. Blazor es un framework de Microsoft que permite construir aplicaciones web interactivas utilizando C# en lugar de JavaScript.

Suite: Implica que el proyecto es una colección o conjunto de herramientas y funcionalidades relacionadas. En este caso, sugiere que BlzSuite ofrece un conjunto completo de herramientas para trabajar con operaciones de datos en aplicaciones Blazor.

1.2 Tecnología para aplicar.

En un proyecto **Blz Suite** que involucre operaciones CRUD utilizando ASP.NET Web API, generalmente se emplearán varias tecnologías:

Blazor: Es la tecnología principal para el desarrollo del front-end. Blazor permite crear interfaces de usuario interactivas utilizando C# y Razor, lo que simplifica la experiencia de desarrollo al utilizar el mismo lenguaje en todo el stack.

ASP.NET Web API: Se utilizará para crear la API RESTful que gestionará las operaciones CRUD en el servidor. ASP.NET Web API es un marco que permite construir servicios web HTTP que son accesibles desde diversos clientes, incluyendo aplicaciones web Blazor.

Entity Framework Core (EF Core): Es probable que se utilice EF Core como ORM (Object-Relational Mapping) para interactuar con la base de datos desde la aplicación ASP.NET Web API. EF Core simplifica el acceso y manipulación de datos en la base de datos utilizando modelos de objetos en lugar de consultas SQL directas.

Base de datos: La base de datos puede variar dependiendo de las necesidades del proyecto y las preferencias del equipo de desarrollo. Algunas opciones comunes podrían incluir Microsoft SQL Server, PostgreSQL, MySQL, SQLite, etc.

Dependency Injection (DI): ASP.NET Core incluye un sistema de inyección de dependencias integrado que facilita la creación y administración de componentes y servicios. Se puede utilizar DI para manejar las dependencias entre los diferentes componentes de la aplicación.

Azure DevOps: Azure DevOps es una suite de herramientas de desarrollo que proporciona servicios para la planificación de proyectos, la colaboración en el código, la integración continua (CI), la entrega continua (CD) y mucho más. Puedes utilizar Azure DevOps para

gestionar el ciclo de vida completo del desarrollo de tu aplicación, desde la planificación inicial hasta el despliegue y la monitorización en producción.

GitHub: GitHub es una plataforma de desarrollo colaborativo que aloja repositorios de código fuente utilizando el sistema de control de versiones Git. Permite a los equipos de desarrollo colaborar en proyectos, realizar seguimiento de problemas, gestionar solicitudes de extracción (pull requests), y mucho más. GitHub es ampliamente utilizado por su facilidad de uso y su integración con una variedad de herramientas y servicios.

1.3 Objetivo del proyecto.

El proyecto **"Blz Suite"** tiene como objetivo principal desarrollar una solución integral para la gestión y operación de datos en aplicaciones web, utilizando tecnologías modernas basadas en .NET. Este proyecto busca automatizar las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) facilitando estas tareas a través de una interfaz de usuario intuitiva y eficiente en Blazor. Esta interfaz estará integrada con servicios backend mediante ASP.NET Web API.

Además de simplificar las operaciones CRUD, la suite tiene como objetivo mejorar la productividad del desarrollo al proporcionar herramientas y frameworks que aceleren la creación y mantenimiento de aplicaciones web. Se busca reducir el tiempo y esfuerzo necesarios para construir sistemas complejos, fomentando así una mayor eficiencia en el desarrollo.

La interoperabilidad y escalabilidad son elementos clave de este proyecto. Se asegurará de que la solución sea fácilmente escalable y pueda integrarse sin problemas con otras plataformas y servicios, especialmente en entornos cloud como Azure. Esto garantizará que la aplicación pueda crecer y adaptarse a las necesidades cambiantes del negocio.

La implementación de prácticas de DevOps será fundamental para el éxito del proyecto. Se integrarán procesos de desarrollo, pruebas y despliegue utilizando herramientas como Azure DevOps y GitHub. Esto mejorará la colaboración entre desarrolladores y permitirá un ciclo de vida de desarrollo de software más ágil y eficiente.

La seguridad y la conformidad son aspectos críticos que se abordarán en el proyecto. Se garantizará que la aplicación cumpla con los estándares de seguridad relevantes y maneje los datos de manera segura, especialmente al tratar con información sensible.

Finalmente, se pondrá un énfasis especial en la accesibilidad y la usabilidad. Se creará una interfaz de usuario que sea accesible y fácil de usar para todos los usuarios, aprovechando las capacidades de Blazor para crear experiencias ricas y responsivas. Esto garantizará que la aplicación sea accesible para una amplia gama de usuarios, independientemente de sus capacidades o dispositivos utilizados.

1.4 Alcance del proyecto.

El proyecto **"Blz Suite"** tiene un alcance empresarial que abarca varias áreas clave, cada una diseñada para tener un impacto directo en la operatividad y la estrategia de negocio. Establecer un alcance claro es fundamental para definir los límites del proyecto y las expectativas de todas las partes interesadas. Aquí se describen detalladamente algunos elementos esenciales del alcance en el plano empresarial:

Soluciones End-to-End: El proyecto se propone ofrecer una solución completa de gestión de datos, que abarca tanto el frontend (con Blazor) como el backend (con ASP.NET Web API). Desde la entrada de datos hasta su manipulación, almacenamiento y recuperación segura, la suite proporcionará una solución integral para todas las necesidades de gestión de datos.

Integración con Sistemas Existentes: Se adaptará la suite para integrarse sin problemas con los sistemas y plataformas de TI existentes en la empresa, como ERP, CRM y otros sistemas de bases de datos. Esto garantizará una transición suave y una interoperabilidad eficiente entre los sistemas existentes y la nueva suite.

Automatización de Procesos de Negocio: Se automatizarán tareas y procesos de negocio repetitivos relacionados con la gestión de datos, lo que reducirá el error humano y aumentará la eficiencia operativa. La automatización permitirá a los empleados concentrarse en tareas de mayor valor añadido, optimizando así los procesos empresariales.

Cumplimiento y Seguridad de Datos: Se garantizará que la suite cumpla con las regulaciones locales e internacionales pertinentes, como GDPR o HIPAA. Se implementarán prácticas de seguridad adecuadas para proteger los datos contra accesos no autorizados y brechas de seguridad, asegurando así la integridad y confidencialidad de los datos.

Capacidades de Reporting y Análisis: Se desarrollarán capacidades de generación de informes y análisis dentro de la suite, permitiendo a los usuarios finales crear visualizaciones de datos y obtener insights que apoyen la toma de decisiones. Esto facilitará la monitorización y evaluación del rendimiento de la empresa, así como la identificación de áreas de mejora.

Escalabilidad y Flexibilidad: La solución se diseñará para ser escalable y adaptable a las crecientes necesidades de la empresa, tanto en términos de volumen de datos como de número de usuarios. Esto garantizará que la suite pueda crecer junto con la empresa y adaptarse a cambios futuros en el entorno empresarial.

Formación y Soporte: Se incluirán programas de formación para los usuarios finales y equipos de TI, así como soporte continuo para resolver problemas y actualizar el sistema conforme evolucionan las necesidades y la tecnología. Esto asegurará que los usuarios estén correctamente capacitados para utilizar la suite y que cuenten con el apoyo necesario en caso de problemas o consultas.

Métricas de Éxito y ROI: Se definirán métricas claras de rendimiento y éxito que ayuden a medir el retorno de la inversión (ROI) y el impacto del proyecto en los objetivos de negocio. Esto permitirá evaluar la eficacia de la suite y justificar la inversión realizada en términos de beneficios tangibles para la empresa.

1.5 Cronograma del proyecto.

Crear un cronograma efectivo para el desarrollo del proyecto **"Blz Suite"** en un plazo de 30 días implica una planificación detallada para garantizar que todas las tareas clave sean cubiertas.

Semana 1: Planificación e Inicio

- *Día 1-2*: Reunión de lanzamiento del proyecto con todos los stakeholders para definir objetivos claros y expectativas.

- *Día 3*: Configuración de herramientas de desarrollo y entornos de trabajo (Azure DevOps, GitHub).

- *Día 4-5*: Diseño detallado de la arquitectura del sistema y planificación del sprint.

Semana 2: Desarrollo del Backend

- *Día 6-8*: Implementación de la API en ASP.NET Web API para operaciones CRUD básicas.

- *Día 9-10*: Configuración de la base de datos y pruebas de integración inicial.

Semana 3: Desarrollo del Frontend

- *Día 11-13*: Desarrollo de componentes Blazor para interfaz de usuario.

- *Día 14-15*: Integración de la interfaz de usuario con la API backend.

Semana 4: Testing y Mejoras

- *Día 16-18*: Testing intensivo de todo el sistema (pruebas unitarias, de integración y de usuario).

- *Día 19-20*: Corrección de errores y ajustes basados en el feedback de las pruebas.

Semana 5: Preparación para el Lanzamiento

- *Día 21-23*: Documentación del sistema y preparación de material de formación para usuarios.

- *Día 24-25*: Configuración final del entorno de producción y pruebas de estrés.

Semana 6: Lanzamiento y Monitorización

- *Día 26*: Lanzamiento oficial del sistema.

- *Día 27-28*: Monitorización del sistema en operación real para detectar problemas inmediatos.

- *Día 29-30*: Evaluación del proyecto, reunión de cierre y planificación de pasos futuros.

1.6 Definir claramente lo que el sistema en un primer Release va a poder hacer.

Para el primer lanzamiento del sistema **"Blz Suite",** es crucial tener una definición clara de sus capacidades, especialmente en la función de Gestión de Empleados en Blazor. Aquí detallo de manera más específica las características y funcionalidades de esta función:

Función de Gestión de Empleados en Blazor:

Menú:

Al acceder a la función 'Empleados' en Blazor, se presentará al usuario un menú claro y accesible que le permita navegar fácilmente por las diferentes opciones disponibles en la gestión de empleados.

Crear Nuevo Empleado:

El menú incluirá una opción claramente identificada para crear un nuevo empleado. Al seleccionar esta opción, se abrirá un formulario interactivo donde el usuario podrá ingresar los detalles del nuevo empleado, incluyendo su nombre completo, departamento, sueldo y fecha de contrato. Este formulario garantizará una experiencia de usuario intuitiva y eficiente, guiando al usuario a través de los campos necesarios para agregar un nuevo empleado con éxito.

Lista de Empleados:

La función de gestión de empleados en Blazor incluirá una lista dinámica y actualizada de todos los empleados creados en el sistema. Esta lista proporcionará una visión general de todos los empleados registrados y permitirá a los usuarios administrar fácilmente la información de cada empleado.

Campos de la Lista de Empleados:

Cada entrada en la lista de empleados incluirá detalles importantes, como el nombre completo del empleado, el departamento al que pertenece, su sueldo y la fecha de contrato. Estos campos clave ofrecerán una visión completa de la información de cada empleado de manera clara y organizada.

Opciones de Editar y Eliminar:

Cada entrada en la lista de empleados estará acompañada de opciones claras para editar o eliminar al empleado correspondiente. Estas opciones permitirán a los usuarios realizar

cambios o eliminar registros de empleados según sea necesario, brindando una gestión completa y flexible de los datos de los empleados.

Esta configuración garantiza que el sistema **"Blz Suite"** tenga las funcionalidades esenciales de recursos humanos desde su lanzamiento inicial. Proporciona una herramienta práctica y fácil de usar para la gestión efectiva de la información de los empleados, permitiendo a los usuarios registrar, visualizar, editar y eliminar registros de empleados de manera eficiente.

1.7 Definir requerimientos del sistema para el primer Release.

Requerimientos del Sistema:

UI/UX (Interfaz de Usuario/Experiencia de Usuario):

Diseño intuitivo y atractivo que permita a los usuarios navegar fácilmente por el sistema.

Menú claro y organizado que facilite el acceso a las diferentes funcionalidades, incluida la gestión de empleados.

Formularios de entrada de datos bien diseñados para agregar y editar empleados, con validaciones de datos para garantizar la integridad de la información.

Front-End (Cliente):

Desarrollo de las funcionalidades front-end para agregar, editar y eliminar empleados.

Implementación de interfaces de usuario interactivas y responsivas utilizando Blazor para una experiencia de usuario fluida.

Back-End y Base de Datos:

Diseño y desarrollo de la estructura de base de datos para almacenar la información de los empleados de manera segura y eficiente.

Implementación de la lógica de negocio en el back-end utilizando ASP.NET Web API para manejar las operaciones CRUD en la base de datos.

Integración adecuada entre el front-end y el back-end para garantizar la comunicación eficaz entre los componentes del sistema.

Seguridad:

Implementación de mecanismos de autenticación de usuarios para garantizar que solo usuarios autorizados puedan acceder al sistema.

Protección de datos sensibles mediante técnicas de encriptación y acceso controlado a recursos.

Pruebas:

Realización de pruebas exhaustivas para verificar la funcionalidad y la usabilidad del sistema.

Pruebas de integración para asegurar que todos los componentes del sistema trabajen de manera conjunta sin problemas.

Despliegue:

Configuración del entorno de producción para alojar el sistema de forma segura y accesible.

Automatización del proceso de despliegue para facilitar actualizaciones y mantenimiento del sistema.

Documentación:

Creación de documentación técnica detallada que describa la arquitectura, los componentes y el funcionamiento del sistema.

Elaboración de manuales de usuario para proporcionar instrucciones claras sobre cómo utilizar las diferentes funcionalidades del sistema.

Cumplimiento:

Asegurar que el sistema cumpla con las regulaciones pertinentes y los estándares de accesibilidad para garantizar la accesibilidad y la legalidad del sistema.

Estos requerimientos esenciales asegurarán que el sistema **"BlzSuite"** esté listo y sea eficiente para su primer lanzamiento, proporcionando una solución completa y funcional para la gestión de empleados.

1. Equipo Metodología Scrum:

2.1. Definir tareas a ejecutar

Semana 1: Planificación e Inicio

Reunión de Lanzamiento del Proyecto: Se llevará a cabo una reunión inicial con todas las partes interesadas para definir objetivos claros, alcance del proyecto y expectativas de entrega.

Configuración de Herramientas de Desarrollo: Configuración de las herramientas necesarias para el desarrollo del proyecto, incluyendo Azure DevOps, GitHub y entornos de desarrollo locales.

Diseño de Arquitectura del Sistema: Se realizará un diseño detallado de la arquitectura del sistema, incluyendo la estructura de la base de datos, la lógica de negocio y la interacción entre componentes.

Planificación del Sprint: Se realizará una planificación detallada del primer sprint, identificando las tareas específicas a realizar y asignando recursos y plazos.

Semana 2: Desarrollo del Backend

Implementación de API ASP.NET Web API para CRUD básico: Desarrollo de la API en ASP.NET Web API para gestionar las operaciones CRUD básicas de los empleados.

Configuración de Base de Datos: Configuración y diseño inicial de la base de datos que almacenará la información de los empleados.

Pruebas de Integración Inicial: Realización de pruebas de integración para asegurar que la API y la base de datos estén funcionando correctamente juntas.

Semana 3: Desarrollo del Frontend

Desarrollo de Componentes Blazor para Interfaz de Usuario: Creación de componentes Blazor que permitan a los usuarios interactuar con la aplicación y gestionar la información de los empleados.

Integración de Interfaz de Usuario con API Backend: Integración de la interfaz de usuario desarrollada con la API backend, permitiendo la visualización y manipulación de datos de empleados.

Semana 4: Testing y Mejoras

Pruebas Unitarias de Backend: Desarrollo y ejecución de pruebas unitarias para asegurar la correcta funcionalidad de la lógica de negocio en el backend.

Pruebas de Integración de Frontend y Backend: Realización de pruebas de integración para validar la comunicación entre el frontend y el backend.

Pruebas de Usuario: Pruebas de usuario para garantizar que la aplicación cumpla con los requisitos y expectativas de los usuarios finales.

Corrección de Errores y Ajustes: Corrección de cualquier error identificado durante las pruebas y realización de ajustes según sea necesario para mejorar la calidad del producto final.

Semana 5: Preparación para el Lanzamiento

Documentación del Sistema: Creación de documentación técnica y de usuario que describa la arquitectura, funcionamiento y uso de la aplicación.

Preparación de Material de Formación para Usuarios: Desarrollo de material de formación y entrenamiento para usuarios finales.

Configuración del Entorno de Producción: Configuración del entorno de producción en Azure u otro proveedor de servicios en la nube.

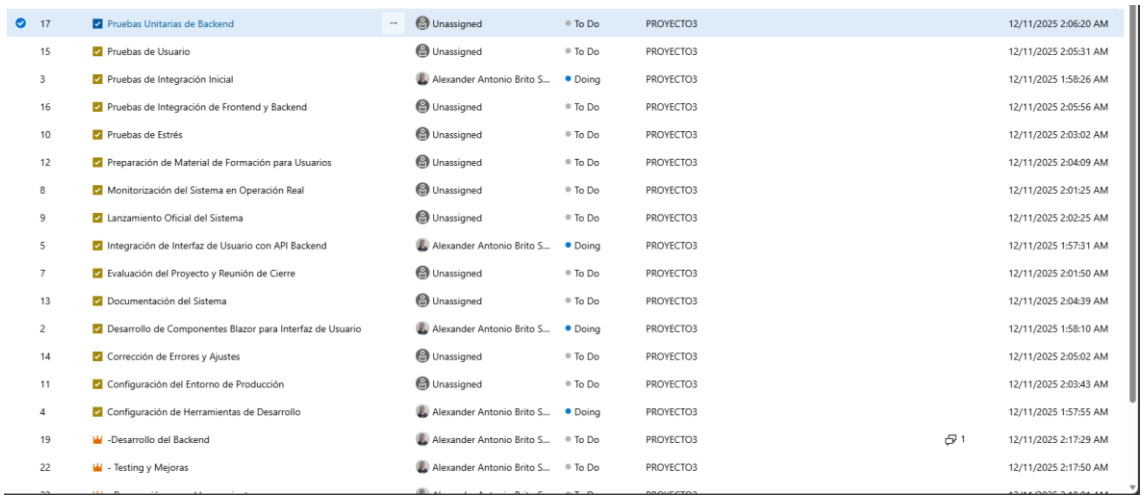
Pruebas de Estrés: Realización de pruebas de estrés para evaluar el rendimiento y la estabilidad del sistema bajo cargas de trabajo extremas.

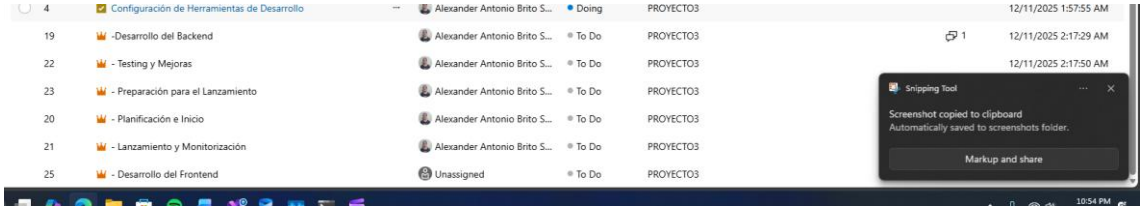
Semana 6: Lanzamiento y Monitorización

Lanzamiento Oficial del Sistema: Despliegue y lanzamiento oficial del sistema en el entorno de producción.

Monitorización del Sistema en Operación Real: Monitorización continua del sistema en operación real para identificar y resolver posibles problemas.

Evaluación del Proyecto y Reunión de Cierre: Evaluación del proyecto en su conjunto, identificación de lecciones aprendidas y planificación de futuras mejoras y proyectos.





2.2. Definir el equipo de trabajo (roles, habilidades, etc)

Líder de Proyecto / Scrum Master: Responsable de liderar el equipo, coordinar las actividades y garantizar que el proyecto avance de acuerdo al plan establecido.

Encargado: Alexander Brito 20230645@itla.edu.do

Desarrollador Backend (.NET): Encargado de desarrollar la API en ASP.NET Web API para gestionar las operaciones CRUD del sistema.

Encargado: Alexander Brito 20230645@itla.edu.do

Desarrollador Frontend (Blazor): Responsable de desarrollar la interfaz de usuario utilizando Blazor y de integrarla con la API backend.

Encargado: Alexander Brito 20230645@itla.edu.do

Diseñador UX/UI: Encargado de diseñar la interfaz de usuario para garantizar una experiencia de usuario intuitiva y atractiva.

Encargado: Alexander Brito 20230645@itla.edu.do

Especialista en Bases de Datos: Responsable de diseñar y configurar la base de datos que almacenará la información de los empleados.

Encargado: Alexander Brito 20230645@itla.edu.do

Tester / QA: Encargado de realizar pruebas exhaustivas para garantizar la calidad y funcionalidad del sistema.

Encargado: Alexander Brito 20230645@itla.edu.do

Documentador Técnico: Encargado de crear la documentación técnica del proyecto, incluyendo manuales de usuario y técnicos.

Encargado: Alexander Brito 20230645@itla.edu.do

Especialista en Despliegue y Operaciones (DevOps): Responsable de configurar el entorno de producción y automatizar el proceso de despliegue del sistema.

Encargado: Alexander Brito 20230645@itla.edu.do

Especialista en Formación y Soporte: Encargado de desarrollar material de formación para usuarios y proporcionar soporte continuo después del lanzamiento del sistema.

Encargado: Alexander Brito 20230645@itla.edu.do

2.3. Herramientas que usarían

Líder de Proyecto / Scrum Master:

- Herramientas de gestión de proyectos como Azure DevOps, Jira o Trello para planificar y seguir el progreso del proyecto.

- Plataformas de comunicación y colaboración como Microsoft Teams, Slack o Discord para coordinar al equipo y comunicarse con los stakeholders.

Desarrollador Backend (.NET):

- IDE (Entorno de Desarrollo Integrado) como Visual Studio o Visual Studio Code para escribir y depurar código en .NET.

- Git y GitHub para el control de versiones y colaboración en el código fuente.

- Postman para probar y depurar la API.

Desarrollador Frontend (Blazor):

- Visual Studio o Visual Studio Code para desarrollar componentes Blazor.

- Git y GitHub para control de versiones y colaboración.

- Herramientas de diseño como Adobe XD, Sketch o Figma para prototipado y diseño de la interfaz de usuario.

Diseñador UX/UI:

- Herramientas de diseño como Adobe XD, Sketch o Figma para crear prototipos y diseños de interfaz de usuario.

- Plataformas de colaboración como Zeplin o InVision para compartir diseños con el equipo de desarrollo.

Especialista en Bases de Datos:

- Gestor de bases de datos como SQL Server Management Studio o Azure Data Studio para diseñar y administrar la base de datos.

- Azure Portal u otras plataformas de nube para configurar y administrar servicios de base de datos en la nube.

Tester / QA:

- Herramientas de pruebas automatizadas como Selenium o SpecFlow para escribir y ejecutar pruebas automatizadas.

- Azure DevOps, Jira o similar para registrar y hacer seguimiento de los errores encontrados durante las pruebas.

Documentador Técnico:

- Microsoft Word, Google Docs o Markdown para escribir la documentación técnica.

- Azure DevOps Wiki, Confluence o similar para almacenar y compartir la documentación con el equipo.

Especialista en Despliegue y Operaciones (DevOps):

- Azure DevOps, GitHub Actions o Jenkins para automatizar el proceso de integración continua y despliegue continuo (CI/CD).

- Azure Portal o herramientas de infraestructura como código (Terraform, ARM Templates) para configurar y gestionar la infraestructura en la nube.

Especialista en Formación y Soporte:

- Plataformas de formación como Microsoft Learn, Udemy o Coursera para crear material de formación.

- Sistemas de gestión de tickets como Zendesk o Freshdesk para registrar y resolver consultas de soporte de usuarios.

2.4. Definir las épicas

Semana 1: Planificación e Inicio

Objetivo: Establecer una base sólida para el proyecto y definir claramente los objetivos y expectativas.

Criterios de Aceptación:

- Reunión de lanzamiento del proyecto realizada con todos los stakeholders para alinear expectativas.
- Configuración exitosa de herramientas de desarrollo y entornos de trabajo, incluyendo Azure DevOps y GitHub.
- Arquitectura del sistema diseñada y documentada, junto con un plan detallado para el primer sprint.

Semana 2: Desarrollo del Backend

Objetivo: Implementar la lógica del servidor necesaria para manejar las operaciones CRUD básicas.

Criterios de Aceptación:

- API desarrollada en ASP.NET Web API para realizar operaciones CRUD básicas.
- Base de datos configurada correctamente y pruebas de integración inicial exitosas.

Semana 3: Desarrollo del Frontend

Objetivo: Crear la interfaz de usuario utilizando componentes Blazor y conectarla con el backend.

Criterios de Aceptación:

- Componentes Blazor desarrollados para la interfaz de usuario de gestión de empleados.
- Integración exitosa de la interfaz de usuario con la API backend, permitiendo la visualización y manipulación de datos de empleados.

Semana 4: Testing y Mejoras

Objetivo: Realizar pruebas exhaustivas para garantizar la calidad y corregir cualquier problema identificado.

Criterios de Aceptación:

- Pruebas unitarias, de integración y de usuario realizadas y documentadas.

- Corrección de errores y ajustes basados en el feedback de las pruebas, garantizando un funcionamiento suave y sin errores del sistema.

Semana 5: Preparación para el Lanzamiento

Objetivo: Preparar el sistema y la documentación para el lanzamiento oficial.

Criterios de Aceptación:

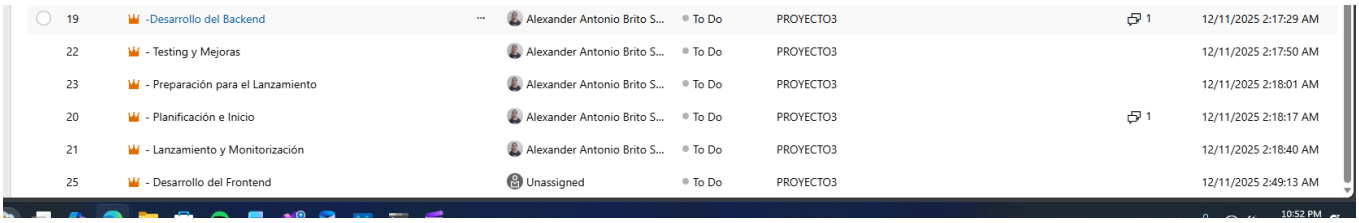
- Documentación del sistema completa y actualizada, incluyendo manuales de usuario y técnico.
- Entorno de producción configurado y listo para el despliegue final.
- Pruebas de estrés realizadas para garantizar la estabilidad y rendimiento del sistema en producción.

Semana 6: Lanzamiento y Monitorización

Objetivo: Lanzar oficialmente el sistema y monitorear su rendimiento en operación real.

Criterios de Aceptación:

- Lanzamiento oficial del sistema sin problemas, con una comunicación adecuada a todos los usuarios y partes interesadas.
- Monitorización del sistema durante los primeros días de operación para detectar y solucionar cualquier problema inmediato.
- Evaluación del proyecto realizada, incluyendo una reunión de cierre con todas las partes interesadas y la planificación de futuras mejoras y pasos a seguir.



2. Acciones:

3.1. Definir las fechas para cada ceremonia de Scrum. ´

Ceremonias Scrum:

Reunión de Planificación del Sprint:

Fecha: Al inicio de cada semana, preferiblemente el lunes por la mañana.

Duración: 1-2 horas, dependiendo de la duración del sprint y la complejidad de las tareas.

Primera reunión: 1/12/2025

Reuniones Diarias (Daily Standup):

Fecha: Todos los días laborables, de lunes a viernes.

Duración: 15 minutos.

Hora: Se sugiere realizarlas por la mañana, al inicio de la jornada laboral.

Revisión del Sprint:

Fecha: Al final de cada semana de trabajo, preferiblemente los viernes.

Duración: 1 hora.

Hora: Idealmente por la tarde, para permitir tiempo suficiente para completar las tareas planificadas.

Primera revisión: 8/12/2025

Retrospectiva del Sprint:

Fecha: Al final de cada semana de trabajo, después de la revisión del sprint(viernes).

Duración: 1 hora.

Hora: Inmediatamente después de la revisión del sprint, para permitir una reflexión inmediata sobre el trabajo realizado.

Reunión de Planificación de Release (Opcional, según necesidad):

Fecha: Al finalizar un ciclo de sprints, antes de iniciar el próximo.

Duración: 1-2 horas.

Hora: Se puede programar al final de una retrospectiva de sprint o en un día separado, dependiendo de la disponibilidad del equipo.

3.2. Afinar al menos 10 historias de usuarios (criterios de aceptación obligatorios).

Semana 1: Planificación e Inicio

1- Reunión de Lanzamiento del Proyecto

Como: miembro del equipo de desarrollo,

Quiero: participar en una reunión de lanzamiento del proyecto con todos los stakeholders,

Para: alinear expectativas y comprender claramente los objetivos del proyecto.

Criterios de Aceptación:

Todos los stakeholders están presentes en la reunión.

Se discuten y documentan los objetivos y expectativas del proyecto.

Se establece una agenda clara para las próximas etapas del proyecto.

2- Configuración de Herramientas de Desarrollo

Como: miembro del equipo de desarrollo,

Quiero: configurar con éxito las herramientas de desarrollo y los entornos de trabajo, incluyendo Azure DevOps y GitHub,

Para: asegurar un flujo de trabajo eficiente y colaborativo.

Criterios de Aceptación:

Las herramientas de desarrollo (Azure DevOps, GitHub) están correctamente configuradas.

Todos los miembros del equipo tienen acceso y están familiarizados con las herramientas.

Se establecen flujos de trabajo y se documentan las políticas de colaboración.

3- Diseño de Arquitectura del Sistema

Como: arquitecto de sistemas,

Quiero: diseñar y documentar la arquitectura del sistema, junto con un plan detallado para el primer sprint,

Para: establecer una base sólida para el desarrollo y garantizar la coherencia en el diseño.

Criterios de Aceptación:

Se crea un documento detallado que describe la arquitectura del sistema.

Se identifican los componentes clave y se documentan sus interacciones.

Se establece un plan detallado para el primer sprint, incluyendo tareas específicas y asignación de recursos.

Semana 2: Desarrollo del Backend

4- Desarrollo de API ASP.NET Web API

Como: desarrollador backend,

Quiero: desarrollar una API en ASP.NET Web API para realizar operaciones CRUD básicas,

Para: permitir la manipulación de datos de empleados desde el frontend.

Criterios de Aceptación:

Se implementan endpoints para crear, leer, actualizar y eliminar empleados.

La API es capaz de manejar solicitudes HTTP correctamente y devolver respuestas apropiadas.

5- Configuración de Base de Datos

Como: desarrollador backend,

Quiero: configurar correctamente la base de datos y realizar pruebas de integración inicial,

Para: asegurar que la API pueda interactuar adecuadamente con el almacenamiento de datos.

Criterios de Aceptación:

Se configura y se establece la conexión con la base de datos.

Se realizan pruebas de integración para asegurar que la API pueda leer y escribir datos en la base de datos correctamente.

Semana 3: Desarrollo del Frontend

6- Desarrollo de Componentes Blazor

Como: desarrollador frontend,

Quiero: desarrollar componentes Blazor para la interfaz de usuario de gestión de empleados,

Para: permitir a los usuarios interactuar de manera efectiva con el sistema.

Criterios de Aceptación:

Se desarrollan componentes para la visualización y manipulación de datos de empleados.

Los componentes son responsivos y se integran de manera coherente con el diseño general de la aplicación.

7- Integración de Interfaz de Usuario con API Backend

Como: desarrollador frontend,

Quiero: integrar la interfaz de usuario con la API backend,

Para: permitir la visualización y manipulación de datos de empleados desde la interfaz de usuario.

Criterios de Aceptación:

La interfaz de usuario puede enviar solicitudes HTTP a la API backend para realizar operaciones CRUD en los datos de empleados.

Los datos recuperados de la API se muestran correctamente en la interfaz de usuario.

Semana 4: Testing y Mejoras

8- Pruebas Unitarias y de Integración

Como: tester / desarrollador,

Quiero: realizar pruebas unitarias y de integración para garantizar la calidad del sistema,

Para: identificar y corregir errores antes del lanzamiento.

Criterios de Aceptación:

Se escriben y ejecutan pruebas unitarias para todas las funciones del backend y del frontend.

Se realizan pruebas de integración para validar la comunicación entre el frontend y el backend.

9- Pruebas de Usuario

Como: tester / desarrollador,

Quiero: realizar pruebas de usuario para validar la usabilidad y funcionalidad del sistema,

Para: asegurar que cumple con los requisitos y expectativas de los usuarios finales.

Criterios de Aceptación:

Se diseñan escenarios de prueba realistas que cubren las funcionalidades clave del sistema.

Los usuarios finales prueban el sistema y proporcionan retroalimentación sobre la experiencia de uso.

Semana 5: Preparación para el Lanzamiento

10- Documentación del Sistema

Como: documentador técnico,

Quiero: crear documentación completa y actualizada del sistema, incluyendo manuales de usuario y técnico,

Para: facilitar el entendimiento y el uso adecuado del sistema por parte de los usuarios y el equipo de desarrollo.

Criterios de Aceptación:

Se crea un manual de usuario que describe cómo utilizar las funciones principales del sistema.

Se genera documentación técnica que detalla la arquitectura del sistema y los procedimientos de mantenimiento.

3.3.Asignar los puntos de historia

1- Reunión de Lanzamiento del Proyecto: 2 puntos

Esta historia de usuario implica una reunión inicial para alinear expectativas y definir objetivos, lo cual es relativamente sencillo pero importante para el éxito del proyecto.

2- Configuración de Herramientas de Desarrollo: 3 puntos

Configurar herramientas como Azure DevOps y GitHub puede llevar algo de tiempo y esfuerzo, especialmente asegurando la correcta configuración y acceso para todos los miembros del equipo.

3- Diseño de Arquitectura del Sistema: 5 puntos

Esta historia de usuario implica una planificación detallada y la documentación de la arquitectura del sistema, lo cual puede ser complejo y requiere un análisis exhaustivo de los requisitos y componentes del sistema.

4- Desarrollo de API ASP.NET Web API: 8 puntos

Implementar una API para operaciones CRUD básicas implica diseñar y desarrollar endpoints funcionales, gestionar la validación de datos y garantizar la seguridad de la API, lo cual puede ser una tarea considerable.

5- Configuración de Base de Datos: 5 puntos

Configurar una base de datos correctamente y realizar pruebas de integración inicial puede ser complejo, especialmente al asegurar la correcta interacción entre la API y la base de datos.

6- Desarrollo de Componentes Blazor: 6 puntos

Desarrollar componentes Blazor para la interfaz de usuario implica crear funcionalidades interactivas y responsivas, lo cual puede requerir un diseño cuidadoso y la implementación de lógica compleja.

7- Integración de Interfaz de Usuario con API Backend: 4 puntos

Integrar la interfaz de usuario con la API backend implica asegurar una comunicación fluida y el intercambio de datos entre los componentes del frontend y del backend.

8- Pruebas Unitarias y de Integración: 7 puntos

Realizar pruebas unitarias y de integración exhaustivas puede ser una tarea laboriosa, ya que implica escribir y ejecutar casos de prueba para garantizar la funcionalidad y la calidad del sistema.

9- Pruebas de Usuario: 6 puntos

Realizar pruebas de usuario implica diseñar escenarios de prueba realistas y recopilar y analizar la retroalimentación de los usuarios finales, lo cual puede requerir tiempo y coordinación.

10- Documentación del Sistema: 4 puntos

Crear documentación completa y actualizada del sistema puede ser una tarea compleja, pero es esencial para garantizar que el sistema sea comprensible y mantenible a largo plazo.

4. Plan de Pruebas

4.1.Lista de requerimientos funcionales y no funcionales de acorde a las historias de usuarios. (mínimo de 10 HU).

Requerimientos Funcionales:

Crear Nuevo Empleado:

RF1: El sistema debe permitir al usuario acceder a la opción de crear un nuevo empleado desde la página de empleados.

RF2: Debe mostrarse un formulario con campos para Nombre Completo, Departamento, Sueldo y Fecha de Contrato al seleccionar la opción de nuevo empleado.

RF3: Debe ser posible ingresar valores válidos en cada campo del formulario para crear un nuevo empleado.

Validación de Datos:

RF4: El sistema debe validar que el campo "Nombre Completo" no esté vacío antes de permitir la creación de un nuevo empleado.

RF5: Debe ser obligatorio seleccionar un departamento válido de la lista desplegable antes de crear un nuevo empleado.

RF6: El sistema debe asegurarse de que el sueldo ingresado sea un valor numérico válido y no negativo.

Creación Exitosa:

RF7: Después de completar el formulario y hacer clic en "Guardar", el sistema debe crear un nuevo empleado y mostrar un mensaje de confirmación al usuario.

RF8: El nuevo empleado creado debe aparecer en la lista de empleados después de ser guardado exitosamente.

Requerimientos No Funcionales:

Interfaz de Usuario Responsiva:

RNF1: La página de creación de nuevo empleado debe ser responsiva y adaptable a diferentes tamaños de pantalla y dispositivos.

RNF2: Los campos del formulario deben ser fácilmente legibles y seleccionables en dispositivos móviles y tablets.

Velocidad de Carga:

RNF3: El formulario de creación de nuevo empleado debe cargarse rápidamente, sin demoras perceptibles para el usuario.

RNF4: La lista de departamentos y la opción de seleccionar el sueldo inicial deben cargarse de manera eficiente para una experiencia de usuario fluida.

Seguridad de Datos:

RNF5: Los datos ingresados por el usuario en el formulario de creación de nuevo empleado deben ser transmitidos de manera segura a través de HTTPS.

RNF6: Se debe implementar una política de privacidad y protección de datos para garantizar la confidencialidad de la información del empleado.

Compatibilidad del Navegador:

RNF7: El formulario de creación de nuevo empleado debe ser compatible con los principales navegadores web, como Chrome, Firefox, Edge y Safari.

RNF8: Debe ser posible completar y enviar el formulario sin problemas en diferentes navegadores sin afectar la funcionalidad.

Usabilidad:

RNF9: El formulario de creación de nuevo empleado debe ser intuitivo y fácil de usar, con etiquetas claras y ayuda contextual cuando sea necesario.

RNF10: Debe ser evidente para el usuario cómo completar y enviar el formulario sin la necesidad de instrucciones adicionales.

4.2- Definir cuáles son los criterios de aceptación de las pruebas.

Crear Nuevo Empleado:

CA1: Al seleccionar la opción "Nuevo Empleado" desde la página de empleados, se debe abrir un formulario de creación de empleado.

CA2: El formulario debe incluir campos para Nombre Completo, Departamento, Sueldo y Fecha de Contrato.

CA3: Todos los campos del formulario deben ser editables y visibles para el usuario.

Validación de Datos:

CA4: El sistema debe mostrar un mensaje de error si el campo "Nombre Completo" está vacío al intentar guardar el empleado.

CA5: Debe ser imposible guardar un empleado si no se ha seleccionado un departamento válido de la lista desplegable.

CA6: Se debe mostrar un mensaje de error si el sueldo ingresado es negativo o no es un valor numérico válido.

Creación Exitosa:

CA7: Después de completar y enviar el formulario con datos válidos, el sistema debe crear un nuevo empleado.

CA8: El sistema debe mostrar un mensaje de confirmación indicando que el empleado se ha creado correctamente.

CA9: El nuevo empleado creado debe aparecer en la lista de empleados después de haber sido guardado exitosamente.

Interfaz de Usuario Responsiva:

CA10: El formulario de creación de nuevo empleado debe ser completamente funcional y legible en diferentes tamaños de pantalla y dispositivos.

CA11: Todos los campos del formulario deben ser seleccionables y editables sin problemas en dispositivos móviles y tablets.

Velocidad de Carga:

CA12: El formulario de creación de nuevo empleado debe cargarse en menos de 3 segundos en cualquier dispositivo y conexión a Internet estándar.

CA13: La lista de departamentos y la opción de seleccionar el sueldo inicial deben aparecer instantáneamente sin retrasos perceptibles.

Seguridad de Datos:

CA14: Todos los datos ingresados por el usuario en el formulario de creación de nuevo empleado deben ser transmitidos de manera segura a través de HTTPS.

CA15: No debe haber ninguna vulnerabilidad de seguridad identificada durante la transmisión o almacenamiento de datos del empleado.

Compatibilidad del Navegador:

CA16: El formulario de creación de nuevo empleado debe funcionar correctamente en los principales navegadores web, incluyendo Chrome, Firefox, Edge y Safari.

CA17: No debe haber diferencias significativas en la funcionalidad del formulario entre diferentes navegadores.

Usabilidad:

CA18: Todos los campos del formulario deben tener etiquetas claras y descriptivas para guiar al usuario durante el proceso de creación del empleado.

CA19: Debe ser evidente para el usuario cómo completar y enviar el formulario sin la necesidad de instrucciones adicionales.

4.3- Definir cuáles son los criterios de rechazo en las pruebas.

Crear Nuevo Empleado:

CR1: El formulario de creación de nuevo empleado no se abre al seleccionar la opción "Nuevo Empleado" desde la página de empleados.

CR2: Algunos campos necesarios, como Nombre Completo o Departamento, no están presentes en el formulario de creación de empleado.

Validación de Datos:

CR3: No se muestra ningún mensaje de error cuando el campo "Nombre Completo" está vacío al intentar guardar el empleado.

CR4: El sistema permite guardar un empleado sin seleccionar un departamento válido de la lista desplegable.

Creación Exitosa:

CR5: Después de completar y enviar el formulario, no se muestra ningún mensaje de confirmación indicando que el empleado se ha creado correctamente.

CR6: El nuevo empleado creado no aparece en la lista de empleados después de haber sido guardado exitosamente.

Interfaz de Usuario Responsiva:

CR7: El formulario de creación de nuevo empleado no se muestra correctamente en dispositivos móviles o tablets, con campos superpuestos o cortados.

CR8: Algunos campos del formulario no son seleccionables o no se pueden editar en ciertos dispositivos.

Velocidad de Carga:

CR9: El formulario de creación de nuevo empleado tarda más de 3 segundos en cargarse en cualquier dispositivo o conexión a Internet estándar.

CR10: La lista de departamentos o la opción de seleccionar el sueldo inicial tardan mucho en aparecer, causando una experiencia de usuario lenta.

Seguridad de Datos:

CR11: Se identifican vulnerabilidades de seguridad durante la transmisión o almacenamiento de datos del empleado, como la exposición de información sensible.

Compatibilidad del Navegador:

CR12: El formulario de creación de nuevo empleado no funciona correctamente en uno o más navegadores principales, como Chrome o Firefox.

CR13: Se encuentran diferencias significativas en la funcionalidad o apariencia del formulario entre diferentes navegadores.

Usabilidad:

CR14: Los campos del formulario no tienen etiquetas descriptivas o claras, lo que dificulta para el usuario entender qué información se espera.

CR15: No es evidente para el usuario cómo completar y enviar el formulario, y se requieren instrucciones adicionales para realizar la acción.

4.4- Comentar sobre las herramientas de pruebas que estarían usando y justificar respuesta.

Selenium:

¿Qué es?

Selenium es un conjunto de herramientas de software de código abierto que se utiliza para automatizar pruebas en aplicaciones web. Permite interactuar con los elementos de una página web y simular acciones realizadas por usuarios reales, como hacer clic en botones, completar formularios y verificar contenido.

Características Principales:

- Automatización de pruebas en diferentes navegadores.
- Soporte para varios lenguajes de programación, incluyendo Java, C#, Python, etc.
- Capacidad para realizar pruebas de forma remota en diferentes entornos.
- Amplia comunidad de usuarios y documentación disponible.

C#:

¿Qué es?

C# (pronunciado "C sharp") es un lenguaje de programación orientado a objetos desarrollado por Microsoft. Es un componente fundamental de la plataforma .NET y se utiliza ampliamente para el desarrollo de aplicaciones de escritorio, web y móviles en el ecosistema de Microsoft.

Características Principales:

- Sintaxis similar a otros lenguajes de programación como Java y C++.
- Totalmente integrado con el ecosistema de desarrollo de Microsoft, incluyendo Visual Studio y .NET Framework.

- Amplia biblioteca estándar y soporte para desarrollo multiplataforma a través de .NET Core.
- Poderoso soporte para programación orientada a objetos, eventos, manejo de excepciones y más.

Blazor:

¿Qué es?

Blazor es un framework de desarrollo web de código abierto desarrollado por Microsoft que permite crear aplicaciones web interactivas y de una sola página (SPA) utilizando C# en lugar de JavaScript. Utiliza WebAssembly para ejecutar código .NET en el navegador web.

Características Principales:

- Desarrollo de aplicaciones web utilizando C# y .NET.
- Arquitectura de componentes reutilizables y composable.
- Interactividad en tiempo real utilizando SignalR.
- Soporte para routing, validación de formularios, inyección de dependencias y más.
- Amplia integración con herramientas y tecnologías de Microsoft, como Visual Studio y Azure.

El uso de **Selenium con C#** para realizar pruebas automatizadas en un programa desarrollado en **Blazor** proporciona una combinación poderosa de compatibilidad, funcionalidad, flexibilidad y soporte comunitario, lo que lo convierte en una opción sólida para garantizar la calidad y el rendimiento de la aplicación.

Justificación:

Compatibilidad con Blazor: Selenium es una herramienta muy versátil que puede ser utilizada para probar aplicaciones web desarrolladas en una amplia variedad de tecnologías, incluyendo **Blazor**. Dado que **Blazor** genera código **HTML** y **JavaScript** para la interfaz de usuario, **Selenium** puede interactuar con estos elementos web de manera efectiva.

Soporte para C#: Selenium cuenta con una API robusta que puede ser utilizada con varios lenguajes de programación, incluyendo C#. Dado que **Blazor** es un framework

basado en .NET, usar **C#** para escribir las pruebas automatizadas proporciona coherencia y facilidad de integración con el código existente del proyecto.

Amplia Funcionalidad: Selenium ofrece una amplia gama de funciones y capacidades para realizar diferentes tipos de pruebas, desde la interacción con elementos de la interfaz de usuario hasta la verificación de contenido y el manejo de ventanas emergentes. Esto permite crear pruebas exhaustivas que cubran todos los aspectos de la aplicación **Blazor**.

Flexibilidad y Escalabilidad: Selenium es altamente flexible y puede ser utilizado para escribir pruebas simples o complejas, dependiendo de las necesidades del proyecto. Además, es escalable, lo que significa que las pruebas pueden ser ampliadas y modificadas fácilmente a medida que la aplicación **Blazor** crece y evoluciona.

Comunidad y Documentación: Selenium es una herramienta muy popular y cuenta con una gran comunidad de usuarios y desarrolladores. Esto significa que hay una amplia gama de recursos, tutoriales y documentación disponibles para ayudar a los equipos a aprender y utilizar **Selenium** de manera efectiva en sus proyectos.

4.5- Estimar el tiempo que duraría la ejecución de pruebas. (elaborar cronograma de trabajo).

Pruebas de Creación de Empleados:

1. Prueba de Creación de Empleado con Datos Válidos.
2. Prueba de Creación de Empleado con Datos Inválidos.
3. Prueba de Creación de Empleado con Todos los Campos Opcionales.

Pruebas de Edición de Empleados:

4. Prueba de Edición de Empleado con Datos Válidos.
5. Prueba de Edición de Empleado con Datos Inválidos.
6. Prueba de Edición de Empleado con Cambio de Departamento.

Pruebas de Eliminación de Empleados:

7. Prueba de Eliminación de Empleado Existente.
8. Prueba de Eliminación de Empleado Inexistente.
9. Prueba de Confirmación de Eliminación de Empleado.
10. Prueba de Cancelación de Eliminación de Empleado.

Cronograma de Trabajo:

Día 1-2:

- Preparación del entorno de pruebas.
- Creación de scripts de pruebas automatizadas para las pruebas 1, 2 y 3.

Día 3-4:

- Ejecución de las pruebas 1, 2 y 3.
- Documentación de resultados y corrección de errores si es necesario.

Día 5-6:

- Creación de scripts de pruebas automatizadas para las pruebas 4, 5 y 6.
- Ejecución de las pruebas 4, 5 y 6.
- Documentación de resultados y corrección de errores si es necesario.

Día 7-8:

- Creación de scripts de pruebas automatizadas para las pruebas 7, 8 y 9.

Ejecución de las pruebas 7, 8 y 9.

- Documentación de resultados y corrección de errores si es necesario.

Día 9-10:

- Creación de scripts de pruebas automatizadas para la prueba 10.

Ejecución de la prueba 10.

- Documentación de resultados y corrección de errores si es necesario.

Tiempo Total de Ejecución:

La ejecución de las 10 pruebas tomará aproximadamente 10 días hábiles, con un total de 2 semanas de trabajo.

4.6- Elaborar plantillas para cada caso de pruebas.

Prueba de Creación de Empleado con Datos Válidos:

Nombre: Prueba de Creación de Empleado con Datos Válidos

Descripción: Verificar que un nuevo empleado pueda ser creado correctamente con datos válidos.

Pasos:

1. Abrir la página de gestión de empleados.
2. Seleccionar la opción para agregar un nuevo empleado.
3. Ingresar datos válidos en todos los campos del formulario.

4. Enviar el formulario para crear el nuevo empleado.
5. Verificar que el nuevo empleado aparezca en la lista de empleados.

Criterios de Aceptación:

- El nuevo empleado se crea correctamente y aparece en la lista de empleados.

- Todos los datos ingresados durante la creación del empleado son correctos y se muestran correctamente en la lista.

Prueba de Edición de Empleado con Cambio de Departamento:

Nombre: Prueba de Edición de Empleado con Cambio de Departamento

Descripción: Verificar que un empleado existente pueda ser editado correctamente, incluyendo un cambio de departamento.

Pasos:

1. Abrir la página de gestión de empleados.
2. Seleccionar un empleado existente para editar.
3. Cambiar el departamento del empleado seleccionado.
4. Guardar los cambios realizados.
5. Verificar que los cambios se reflejen correctamente en la lista de empleados.

Criterios de Aceptación:

- El departamento del empleado se actualiza correctamente con el valor especificado durante la edición.

- Los demás datos del empleado permanecen intactos después de la edición.

Prueba de Eliminación de Empleado Inexistente:

Nombre: Prueba de Eliminación de Empleado Inexistente

Descripción: Verificar que no se pueda eliminar un empleado que no existe en el sistema.

Pasos:

1. Abrir la página de gestión de empleados.
2. Intentar eliminar un empleado que no está presente en la lista.
3. Confirmar la eliminación del empleado inexistente.

4. Verificar que no se realice ninguna acción y no se muestre ningún mensaje de confirmación.

Criterios de Aceptación:

- No se permite la eliminación de un empleado inexistente.

- No se muestra ningún mensaje de confirmación o error durante el intento de eliminación.

4.7- Elaborar una plantilla con los equipos de pruebas y sus responsabilidades.

Equipo de Pruebas

Líder de Pruebas

Responsabilidades:

- Coordinar y supervisar todas las actividades de pruebas.

- Planificar y programar pruebas de acuerdo con los plazos del proyecto.

- Asignar tareas y responsabilidades a los miembros del equipo de pruebas.

- Revisar y aprobar casos de prueba y resultados de pruebas.

- Comunicar el progreso y los problemas del proyecto al equipo de desarrollo y a la gerencia.

- Garantizar la calidad y eficacia de las pruebas realizadas

Encargado: Alexander Brito 20230645@itla.edu.do

Ingeniero de Pruebas

Responsabilidades:

- Desarrollar casos de prueba detallados para cada funcionalidad del sistema.

- Ejecutar pruebas manuales y automatizadas según sea necesario.

- Registrar y documentar los resultados de las pruebas.
- Identificar y reportar problemas o defectos en el software.

- Colaborar con el equipo de desarrollo para resolver problemas identificados.
- Mantener y mejorar continuamente los procesos de pruebas.

Encargado: Alexander Brito 20230645@itla.edu.do

Ingeniero de Pruebas Automatizadas

Responsabilidades:

- Desarrollar scripts de pruebas automatizadas utilizando herramientas como Selenium y C#.
- Ejecutar pruebas automatizadas y analizar los resultados.

- Identificar oportunidades para la automatización de pruebas y proponer mejoras.

- Mantener y actualizar los scripts de pruebas automatizadas según sea necesario.

- Colaborar con el equipo de desarrollo para integrar pruebas automatizadas en el flujo de trabajo de desarrollo continuo.

- Participar en la revisión de código y procesos para garantizar la calidad del software.

Encargado: Alexander Brito 20230645@itla.edu.do

Analista de Calidad

Responsabilidades:

- Analizar los requisitos del sistema y definir criterios de aceptación para las pruebas.
- Revisar la documentación del sistema y los casos de uso para identificar posibles áreas de prueba.

- Diseñar estrategias de pruebas efectivas que aborden todos los aspectos críticos del sistema.

- Realizar pruebas de usabilidad y accesibilidad para garantizar una experiencia de usuario óptima.

- Generar informes de calidad y métricas de pruebas para informar a la gerencia sobre el estado del proyecto.

- Colaborar estrechamente con el equipo de desarrollo para garantizar la entrega de un producto de alta calidad.

Encargado: Alexander Brito 20230645@itla.edu.do

4.8- Elaborar un plan de automatización de pruebas.

Plan de Automatización de Pruebas

1. Objetivos de Automatización:

- Automatizar pruebas funcionales y de regresión para mejorar la eficiencia y la calidad del software.
- Reducir el tiempo y los recursos necesarios para realizar pruebas repetitivas.
- Aumentar la cobertura de pruebas y la detección temprana de defectos en el ciclo de desarrollo.

2. Alcance de Automatización:

- Automatizar casos de prueba críticos y de alto impacto.
- Incluir pruebas de integración, funcionales y de usuario en el alcance de la automatización.
- Priorizar la automatización de pruebas que se ejecuten con frecuencia durante el ciclo de desarrollo.

3. Herramientas y Tecnologías:

- Utilizar Selenium WebDriver con C# para la automatización de pruebas de interfaz de usuario.
- Integrar NUnit como framework de pruebas unitarias para la ejecución de pruebas automatizadas.
- Emplear Azure DevOps o herramientas similares para la gestión de pruebas y la integración continua.

4. Estrategia de Automatización:

- Identificar casos de prueba adecuados para la automatización basados en su frecuencia de ejecución, complejidad y criticidad.
- Desarrollar scripts de pruebas automatizadas para cubrir escenarios críticos de uso y funcionalidades clave del sistema.
- Implementar pruebas de regresión automatizadas para garantizar que los cambios no introduzcan nuevos errores en áreas previamente probadas.
- Ejecutar pruebas automatizadas de manera regular como parte del proceso de integración continua.

5. Proceso de Automatización:

- **Identificación de Casos de Prueba:** Revisar la suite de pruebas existente y seleccionar casos de prueba adecuados para la automatización.
- **Diseño de Scripts de Pruebas:** Desarrollar scripts de pruebas automatizadas utilizando Selenium WebDriver y C# para replicar los casos de prueba identificados.
- **Implementación de Pruebas:** Escribir código robusto y modular para las pruebas, teniendo en cuenta la reutilización y la mantenibilidad.
- **Ejecución de Pruebas:** Ejecutar las pruebas automatizadas en diferentes navegadores y entornos para garantizar la compatibilidad y consistencia.
- **Análisis de Resultados:** Analizar los resultados de las pruebas automatizadas y documentar cualquier defecto encontrado.
- **Integración Continua:** Integrar las pruebas automatizadas en el flujo de trabajo de integración continua para ejecutarlas automáticamente después de cada cambio de código.
- **Monitoreo y Mantenimiento:** Supervisar y mantener los scripts de pruebas automatizadas para asegurar su eficacia y relevancia continuas.

6. Métricas de Automatización:

- Porcentaje de casos de prueba automatizados en comparación con el total de casos de prueba.
- Tiempo ahorrado en la ejecución de pruebas manuales debido a la automatización.
- Reducción en el número de defectos introducidos en producción después de la

implementación de pruebas automatizadas.

- Tiempo medio entre la detección y resolución de defectos identificados por pruebas automatizadas.

7. Plan de Implementación:

- Asignar recursos y tiempo suficientes para el desarrollo y la implementación de pruebas automatizadas.
- Establecer un cronograma claro para la creación y ejecución de pruebas automatizadas.
- Capacitar al equipo en el uso de herramientas y tecnologías de automatización seleccionadas.
- Monitorear regularmente el progreso y los resultados de la automatización y realizar ajustes según sea necesario.

Lista de pruebas

1. Prueba de Creación de Empleado con Datos Válidos

2. Prueba de Creación de Empleado con Datos Inválidos

3. Prueba de Creación de Empleado con Todos los Campos Opcionales

4. Prueba de Edición de Empleado con Datos Válidos

5. Prueba de Edición de Empleado con Datos Inválidos

6. Prueba de Edición de Empleado con Cambio de Departamento

7. Prueba de Eliminación de Empleado Existente

8. Prueba de Eliminación de Empleado Inexistente

9. Prueba de Eliminación de Empleado Cancelada

10. Prueba de Eliminación de Empleado con Confirmación

4.9- Ejecutar y demostrar el plan de automación de pruebas.

Video adjunto de pruebas también lo tendrá en git

```
        ChromeOptions options = new ChromeOptions();
        options.AddArgument("--start-maximized");
        IWebDriver driver = new ChromeDriver(options);
        System.Threading.Thread.Sleep(3000);

        // Abrir página de gestión de empleados
        driver.Navigate().GoToUrl("http://localhost:5257/empleados");
        System.Threading.Thread.Sleep(2000);

        //1.Prueba de Creación de Empleado con Datos Válidos:
        // Hacer clic en el botón de Nuevo Empleado
        driver.Navigate().GoToUrl("http://localhost:5257/empleado");
        System.Threading.Thread.Sleep(2000);
        // Ingresar datos válidos en el formulario de creación de empleado
        driver.FindElement(By.CssSelector("input.form-control.valid")).SendKeys("Juan Perez");
        System.Threading.Thread.Sleep(1000);
        driver.FindElement(By.CssSelector("select.form-select.valid")).SendKeys("a");
        driver.FindElement(By.CssSelector("input[type='number'].form-control.valid")).SendKeys("2000");
        System.Threading.Thread.Sleep(1000);
        // Hacer clic en el botón de Guardar
        driver.FindElement(By.CssSelector("button.btn.btn-primary")).Click();
        System.Threading.Thread.Sleep(2000);

        //2.Prueba de Creación de Empleado con Datos Inválidos:
        // Hacer clic en el botón de Nuevo Empleado
        driver.Navigate().GoToUrl("http://localhost:5257/empleado");
        // Ingresar datos inválidos en el formulario de creación de empleado
        System.Threading.Thread.Sleep(1000);
        driver.FindElement(By.CssSelector("select.form-select.valid")).SendKeys("IT");
        driver.FindElement(By.CssSelector("input[type='number'].form-control.valid")).SendKeys("-100");
        System.Threading.Thread.Sleep(1000);
        // Hacer clic en el botón de Guardar
        System.Threading.Thread.Sleep(1000);
        // Hacer clic en el botón de Guardar
        driver.FindElement(By.CssSelector("button.btn.btn-primary")).Click();
        System.Threading.Thread.Sleep(2000);


        //3.Prueba de Creación de Empleado con Todos los Campos Opcionales:
        // Abrir página de gestión de empleados
        driver.Navigate().GoToUrl("http://localhost:5257/empleados");
        System.Threading.Thread.Sleep(2000);
        // Hacer clic en el botón de Nuevo Empleado
        driver.Navigate().GoToUrl("http://localhost:5257/empleado");
        System.Threading.Thread.Sleep(2000);
        // Ingresar solo el nombre del empleado
        driver.FindElement(By.CssSelector("input.form-control.valid")).SendKeys("Maria Garcia");
        System.Threading.Thread.Sleep(1000);
        // Hacer clic en el botón de Guardar
        driver.FindElement(By.CssSelector("button.btn.btn-primary")).Click();
        System.Threading.Thread.Sleep(2000);

        //4.Pruebas de Edición de Empleados:
        driver.Navigate().GoToUrl("http://localhost:5257/empleados");
        System.Threading.Thread.Sleep(2000);
        // Hacer clic en el botón de Editar del primer empleado en la lista
        driver.FindElement(By.CssSelector("i.oi.oi-pencil")).Click();
        System.Threading.Thread.Sleep(2000);
        // Modificar el departamento del empleado
        driver.FindElement(By.CssSelector("select.form-select.valid")).SendKeys("m");
        System.Threading.Thread.Sleep(2000);
        // Hacer clic en el botón de Guardar
        driver.FindElement(By.CssSelector("button.btn.btn-primary")).Click();
        System.Threading.Thread.Sleep(2000);

        //5.Prueba de Edición de Empleado con Datos Inválidos:
        // Hacer clic en el botón de Editar del primer empleado en la lista
```

```
    //6.Prueba de Edición de Empleado con Cambio de Departamento:
    driver.Navigate().GoToUrl("http://localhost:5257/empleados");
    System.Threading.Thread.Sleep(2000);
    // Hacer clic en el botón de Editar del primer empleado en la lista
    driver.FindElement(By.CssSelector("i.oi.oi-pencil")).Click();
    System.Threading.Thread.Sleep(2000);
    // Modificar el departamento del empleado
    driver.FindElement(By.CssSelector("select.form-select.valid")).SendKeys("m");
    System.Threading.Thread.Sleep(2000);
    // Hacer clic en el botón de Guardar
    driver.FindElement(By.CssSelector("button.btn.btn-primary")).Click();
    System.Threading.Thread.Sleep(2000);

    //7.Pruebas de Eliminación de Empleados:
    // Hacer clic en el botón de Eliminar del primer empleado en la lista
    driver.FindElement(By.CssSelector("i.oi.oi-trash")).Click();
    System.Threading.Thread.Sleep(2000);
    // Confirmar la eliminación del empleado
    driver.FindElement(By.CssSelector("button.swal2-confirm.swal2-styled")).Click();
    System.Threading.Thread.Sleep(2000);

    //8.Prueba de Eliminación de Empleado Inexistente:

    //9.Prueba de Eliminación de Empleado Cancelada:
    //7.Pruebas de Eliminación de Empleados:
    // Hacer clic en el botón de Eliminar del primer empleado en la lista
    driver.FindElement(By.CssSelector("i.oi.oi-trash")).Click();
    System.Threading.Thread.Sleep(2000);
    // Confirmar la eliminación del empleado
    driver.FindElement(By.CssSelector("button.swal2-cancel.swal2-styled")).Click();
    System.Threading.Thread.Sleep(2000);
```

```
//9.Prueba de Eliminación de Empleado Cancelada:
//7.Pruebas de Eliminación de Empleados:
// Hacer clic en el botón de Eliminar del primer empleado en la lista
driver.FindElement(By.CssSelector("i.oi.oi-trash")).Click();
System.Threading.Thread.Sleep(2000);
// Confirmar la eliminación del empleado
driver.FindElement(By.CssSelector("button.swal2-cancel.swal2-styled")).Click();
System.Threading.Thread.Sleep(2000);

//10.Prueba de Eliminación de Empleado con Confirmación:
//7.Pruebas de Eliminación de Empleados:
// Hacer clic en el botón de Eliminar del primer empleado en la lista
driver.FindElement(By.CssSelector("i.oi.oi-trash")).Click();
System.Threading.Thread.Sleep(2000);
// Confirmar la eliminación del empleado
driver.FindElement(By.CssSelector("button.swal2-confirm.swal2-styled")).Click();
System.Threading.Thread.Sleep(2000);
```

Conclusión

El proyecto de desarrollo de software "**BlzSuite**" ha sido una empresa ambiciosa y emocionante que ha abordado la creación de una solución integral para la gestión y operación de datos en aplicaciones web, utilizando tecnologías modernas basadas en .NET. A lo largo de este proceso, se han enfrentado desafíos significativos y se han alcanzado importantes hitos que han marcado el camino hacia el éxito.

Desde la fase inicial de planificación e inicio, donde se establecieron los objetivos del proyecto y se definieron claramente las expectativas de todas las partes interesadas, hasta el desarrollo del backend y frontend, donde se implementaron las funcionalidades esenciales para la manipulación de datos y la creación de una interfaz de usuario intuitiva, cada etapa del proyecto ha sido crucial para su realización.

La integración con sistemas existentes, la automatización de procesos de negocio, la garantía de cumplimiento y seguridad de datos, y la preparación meticulosa para el lanzamiento y monitorización del sistema en operación real, han sido aspectos fundamentales que se han abordado con dedicación y atención al detalle.

Además, la inclusión de pruebas automatizadas ha asegurado la calidad del producto final, identificando y corrigiendo posibles problemas antes de su implementación. La colaboración estrecha entre los diferentes equipos de trabajo, cada uno con sus responsabilidades y especialidades, ha sido clave para la ejecución exitosa del proyecto.

Link del github:

Link del AzureDevOps: