Software de display para monitoreo de línea

Proyecto de desarrollo tecnológico

Contenido

[Resumen 4](#_Toc57325954)

[Abstract 5](#_Toc57325955)

[I. Introducción 6](#_Toc57325956)

[II. Planteamiento del problema 7](#_Toc57325957)

[III. Objetivos 8](#_Toc57325958)

[III.1. General 8](#_Toc57325959)

[III.2. Específicos 8](#_Toc57325960)

[IV. Justificación 9](#_Toc57325961)

[V. Delimitación 10](#_Toc57325962)

[VI. Marco de referencia 12](#_Toc57325963)

[VI.1 Marco teórico 12](#_Toc57325964)

[VI.1.1 Teoría general de sistemas (TGS) 12](#_Toc57325965)

[VI.1.2 Ingeniería de sistemas computacionales 12](#_Toc57325966)

[VI.1.3 Ingeniería de software 13](#_Toc57325967)

[VI.1.4 Metodología ágil 13](#_Toc57325968)

[VI.1.5 Sistemas basados en transacciones 13](#_Toc57325969)

[VI.1.6 N-tier 14](#_Toc57325970)

[VI.1.7 Clean code 14](#_Toc57325971)

[VI.1.8 Sistema de control de versiones 15](#_Toc57325972)

[VI.2 Antecedentes 15](#_Toc57325973)

[VI.2.1 Desarrollo e implementación de un sistema de escritorio 15](#_Toc57325974)

[VI.2.2 Diseño de sistema de control y gestión con SQL Server y .NET (uso de LINQ) 16](#_Toc57325975)

[VI.2.3 Desarrollo de software utilizando metodologías ágiles 16](#_Toc57325976)

[VI.2.4 Refactorización y técnicas de clean code 16](#_Toc57325977)

[VI.2.5 Control de versiones 17](#_Toc57325978)

[VI.3 Marco conceptual 17](#_Toc57325979)

[VI.3.1 SAP ERP 17](#_Toc57325980)

[VI.3.2 Escalabilidad 18](#_Toc57325981)

[VI.3.3 Refactorización 18](#_Toc57325982)

[VI.3.4 Procedimiento almacenado 18](#_Toc57325983)

[VI.3.5 Colección de datos 19](#_Toc57325984)

[VI.3.6 CRUD 19](#_Toc57325985)

[VI.3.7 Interfaz de usuario 19](#_Toc57325986)

[VI.3.8 Programación orientada a objetos 19](#_Toc57325987)

[VI.3.9 IDE 20](#_Toc57325988)

[VI.3.10 Visual Basic 20](#_Toc57325989)

[VI.3.11 C# 20](#_Toc57325990)

[VI.3.12 SQL Server 21](#_Toc57325991)

[VI.3.13 ADO.NET 21](#_Toc57325992)

[VI.3.14 DLL 22](#_Toc57325993)

[VI.3.15 XML 22](#_Toc57325994)

[VI.3.16 LINQ 23](#_Toc57325995)

[VII. Metodología 25](#_Toc57325996)

[VII.1 Viabilidad 30](#_Toc57325997)

[VII.1.1 Técnica 30](#_Toc57325998)

[VI.1.2 Económica 31](#_Toc57325999)

[VI.1.3 Operativa 31](#_Toc57326000)

[VIII. Resultados 32](#_Toc57326001)

[VIII.1 Objetivo específico 1. 32](#_Toc57326002)

[VIII.2 Objetivo específico 2 33](#_Toc57326003)

[VIII.3 Objetivo específico 3 34](#_Toc57326004)

[VIII.4 Objetivo específico 4 34](#_Toc57326005)

[VIII.5 Objetivo específico 5 35](#_Toc57326006)

[VIII.6 Objetivo específico 6 36](#_Toc57326007)

[VIII.7 Objetivo específico 7 37](#_Toc57326008)

[VIII.8 Objetivo específico 8 38](#_Toc57326009)

[VIII.9 Objetivo específico 9 39](#_Toc57326010)

[VIII.10 Objetivo específico 10 39](#_Toc57326011)

[VIII.11 Conclusiones 39](#_Toc57326012)

[VIII.12 Recomendaciones 40](#_Toc57326013)

[VII.13 Bibliografía 40](#_Toc57326014)

[VIII.14 Anexos 43](#_Toc57326015)

[VIII.14.1 Manual de usuario 43](#_Toc57326016)

# Índice de figuras

[Ilustración 1. Cronograma de actividades. 13](#_Toc57326017)

[Ilustración 2. Proceso de ejecución con ADO.NET (gráfico propio). 24](#_Toc57326018)

[Ilustración 3. Ejemplo de documento XML (obtenido de Dykes, L., & Tittel, E. (2005). Ejemplo de clasificación con XML. Recuperado de XML for Dummies. Hoboken: Wiley Publishing). 25](#_Toc57326019)

[Ilustración 4. Linq para objetos (gráfico propio). 26](#_Toc57326020)

[Ilustración 5. Linq to XML bajo escritura funcional (gráfico propio). 26](#_Toc57326021)

[Ilustración 6. Ejemplo de tablero kanban (extraído de Anderson & Carmichel, 2016). 29](#_Toc57326022)

[Ilustración 7. Tablero kanban para el proyecto (ilustración propia). 30](#_Toc57326023)

[Ilustración 8. Mini tablero Kanban (gráfico propio). 31](#_Toc57326024)

[Ilustración 9. Comparativa entre patrones de diseño (derechos reservados a ACC MEX S.A. de C.V.). 34](#_Toc57326025)

[Ilustración 10. Programa original (derechos reservados a ACC MEX S.A. de C.V.) 35](#_Toc57326026)

[Ilustración 11. Programa refactorizado, se conservaron muchos aspectos funcionales (derechos reservados a ACC MEX S.A. de C.V.). 35](#_Toc57326027)

[Ilustración 12. Marco de trabajo general (gráfico propio). 36](#_Toc57326028)

[Ilustración 13. Aspecto gráfico original (derechos reservados a ACC MEX S.A. de C.V.). 37](#_Toc57326029)

[Ilustración 14. Apartado gráfico actualizado (derechos reservados a ACC MEX S.A. de C.V.). 37](#_Toc57326030)

[Ilustración 15. Segmentación de los datos (gráfico propio). 38](#_Toc57326031)

[Ilustración 16. Aportaciones al DLL SPEngineCSharp (gráfico propio). 38](#_Toc57326032)

[Ilustración 17. Ejemplo de código con LINQ to XML (derechos reservados a ACC MEX S.A. de C.V.). 39](#_Toc57326033)

[Ilustración 18. Contribuciones a los repositorios del proyecto (gráfico propio). 40](#_Toc57326034)

[Ilustración 19. Operaciones CRUD (derechos reservados a ACC MEX S.A. de C.V.). 41](#_Toc57326035)

[Ilustración 20. Inicio de la aplicación. 45](#_Toc57326036)

[Ilustración 21. Alertas y cargas. 46](#_Toc57326037)

[Ilustración 22. Menú principal. 46](#_Toc57326038)

[Ilustración 23. Botón flotante del menú principal. 47](#_Toc57326039)

[Ilustración 24. Ventana para editar datos. 47](#_Toc57326040)

[Ilustración 25. Datos actualizados con ayuda de las pestañas superiores. 48](#_Toc57326041)

[Ilustración 26. Campos estáticos de la pestaña Datos. 49](#_Toc57326042)

[Ilustración 27. Campos dinámicos y controles para operaciones de alta, baja y modificaciones. 49](#_Toc57326043)

[Ilustración 28. Pestaña Materiales con datos precargados. 50](#_Toc57326044)

[Ilustración 29. Pestaña Materiales sin datos. 50](#_Toc57326045)

[Ilustración 30. Edición de datos en Materiales. 50](#_Toc57326046)

[Ilustración 31. Tabla para registrar las actividades. 51](#_Toc57326047)

[Ilustración 32. Tabla de meta semanal 51](#_Toc57326048)

[Ilustración 33. Tabla de documentos relacionados. 52](#_Toc57326049)

[Ilustración 34. Botones inferiores del a ventana editar datos. 52](#_Toc57326050)

[Ilustración 35. Menú flotante. 52](#_Toc57326051)

[Ilustración 36. Pantalla dinámica. 53](#_Toc57326052)

[Ilustración 37. Menú lateral izquierdo. 53](#_Toc57326053)

[Ilustración 38. Sección estática. 54](#_Toc57326054)

[Ilustración 39. Parte dinámica 1. 54](#_Toc57326055)

[Ilustración 40. Parte dinámica 2. 55](#_Toc57326056)

[Ilustración 41. Controles de tiempo. 55](#_Toc57326057)

# Índice de tablas

[Tabla 1. Principios fundamentales de la ingeniería de software (elaboración propia). 28](#_Toc57326074)

[Tabla 2. Pasos para llevar a cabo el desarrollo de elementos específicos (gráfico propio). 32](#_Toc57326075)

# Resumen

El presente trabajo tiene como objetivo presentar al lector las implicaciones para desarrollar e implementar el software que requiere la empresa ACC MEX S.A de C.V., la presentación de la información se manifiesta desde una perspectiva operacional pues es el principal ámbito en el que un proyecto de desarrollo tecnológico logra tener impacto. El software que se planea liberar tiene módulos para editar la información en base a que parte de la producción se desea controlar, la posibilidad de modificar dicha información e incluso eliminarla y por último cuenta con un apartado para mostrar los datos de forma amena.

El desarrollo de este proyecto fue posible gracias a que se siguieron principios para el desarrollo de sistemas basados en la ingeniería de software, implementación de una metodología ágil, conocimiento y desarrollo de un sistema basado en transacciones, buenas prácticas de programación, manejo de control de versiones y un buen manejo del tiempo para administrar las pruebas del software.

**Palabras clave:** ADO.NET, C#, LINQ to XML, SQL SERVER, Git.

# Abstract

The objective of this paper is to present to the reader the implications for the development and implementation of software required by ACC MEX S.A. de C.V., the presentation of the information is manifested in a operational perspective because is this the main scope where a technollogical development project has impact. The software’s release has modules for information managment based on what producction’s part needs to control, the possibility to modify the information and deleted too and the last part is related to show information in a pleasant way.

The develpment of this project was posible because the development team follow principles to design and mantaine software engenieer proyects, implementing an agile methodology, knowledgment and development of a transactional system, good programming practices, versión control management and a good time administration to make tests.

**Keywords:** ADO.NET, C#, LINQ to XML, SQL SERVER, Git.

# Introducción

Debido a que la tecnología relacionada al desarrollo de software se ve cada vez más envuelta en las empresas, se ha notado una creciente necesidad por automatizar procesos relacionados al área industrial en donde la gestión de recursos es sumamente importante. El tiempo es un recurso que las empresas valoran por encima de cualquier otro, inclusive en algunos casos se puede valorar más que el dinero.

Gracias a la alta demanda en la automatización de procesos algunas empresas han optado por desarrollar subsistemas para llevar el control de algunos recursos, el problema con este tipo de soluciones surge cuando se piensa a futuro, pues generalmente el desarrollo de una solución barata y rápida se traduce en falta de escalabilidad y un aumento considerable de la deuda técnica, gracias a esto surgió la alternativa de crear un sistema monolítico en donde se puedan obtener recursos que a su vez serán consumidos por otras aplicaciones y así aumentar el rendimiento.

Si bien es cierto que actualmente este tipo de arquitecturas pueden carecer de escalabilidad cuando se pretende migrar a la nube, no deja de ser necesario comprender modelos básicos de arquitecturas y tratar de adaptarlas al crecimiento en la nube parece una vertiente que gana un mayor auge de practicantes, pues la migración a la nube puede ser un camino fácil o difícil dependiendo de la calidad con la que se escribió el código.

Dentro de este trabajo se podrán encontrar los detalles para elaborar una aplicación para la línea de producción en ACC MEX S.A de C.V. la cual pasará por un proceso de análisis detallado mientras se trate de definir el planteamiento que tendrá el problema por resolver, seguido de una serie de especificaciones generales que dan pauta a los objetivos de realizar un proyecto, para reafirmar la importancia del mismo se podrá consultar el apartado de Justificación y delimitación.

Un punto muy importante para comprender los problemas de desarrollo es seguir un camino de aprendizaje, en este caso particular se mencionan las teorías en las que se basa el proyecto y que pueden presentarse como casos de oportunidad para seguir desarrollándose los temas que se mencionan en el marco de referencia van relacionados a la teoría general de sistemas, ingeniería del software, metodologías ágiles, sistemas basados en transacciones, aplicación de una arquitectura de N capas (N-tier), aplicación de principios de código limpio y conocimiento sobre los sistemas de control de versiones. Además se hace una mención especial de todas las tecnologías que sirven de apoyo en el desarrollo de este aplicativo entre las que destacan: ADO.NET y el uso de LINQ to XML.

En cuanto a la forma de resolver el problema se puede consultar acerca de ello en el apartado de metodología en donde se detalla cómo fue el proceso de construcción del software y las implicaciones técnicas, operativas y económicas para llevar a cabo el software. En el apartado final se incluyen las conclusiones y resultados obtenidos además de biografía y anexos.

# Planteamiento del problema

En la empresa ACC MEX S.A. De C.V ubicada en Libramiento Sur-Poniente #808 Campestre Italiana, Santiago de Querétaro, Qro. Se llevan a cabo distintos tipos de proyectos relacionados a la ingeniería en automatización y sistemas de control industrial llevando a cabo proyectos en forma de soluciones en robótica, soluciones para visión computacional, desarrollo de software y/o mantenimiento. La empresa cuenta con varias áreas para poder llevar a cabo el proyecto y existe una en específico con el mismo nombre encargada de gestionar de la mejor manera los recursos que se disponen.

Uno de los recursos clave para la empresa es el tiempo, pues un proyecto suele demorar más o menos en demorarse si los procesos se gestionan adecuadamente, dicha gestión es un factor que se suele manejar por medio de programas propios o de terceros dentro de la empresa, uno de los sectores donde menos visibilidad del tiempo invertido sobre un proyecto se tiene es precisamente en el área de producción.

Hasta este momento la empresa cuenta con un software que permite visualizar los procesos en línea de producción en tres pantallas distintas para el contexto temporal en que se desarrollo pudo resolver adecuadamente sus necesidades pues principalmente los cliente cuando se dirigían a revisar el estado de sus proyectos podían notar una avance sin dejar la incertidumbre de por medio, este programa fue desarrollado en Visual Basic con el funcionamiento adecuado, sin embargo durante el inicio del año 2020 el gerente se dio cuenta de la falta de actualización de este y otros programas dentro de las áreas de la empresa por lo que se encuentra en un proceso de actualización e integración de un sistema SAP para controlar eficientemente sus datos.

Dentro de los problemas que se han encontrado esta la necesidad de integrar sus aplicaciones al SAP, cambiar la estructura interna a un patrón de diseño escalable, delimitar los datos que entran o salen del origen de datos y llevar a cabo la implementación de las aplicaciones en producción.

# Objetivos

## General

Desarrollar e implementar una aplicación de escritorio para la línea de producción de ACC MEX S.A. De C.V.

## Específicos

* Reestructurar el patrón de diseño de la aplicación para aumentar su escalabilidad utilizando un modelo de n-capas (presentación, negocio, entidades y datos).
* Analizar los requerimientos respecto al sistema actual, conservando el objetivo principal de la aplicación (informar sobre el estado en producción).
* Conocer el entorno de desarrollo en C# para realizar una refactorización de código desde VB.
* Implementar un diseño orientado a UI/UX para facilitar la comprensión de la información
* Segmentar correctamente el origen de datos local con archivos de configuración y de base de datos utilizando ADO.NET para evitar redundancia.
* Crear e implementar un DLL para manipular procedimientos almacenados de una manera más amena.
* Implementar librerías específicas de C# para manipular componentes y colecciones de datos de XML.
* Manejar un control de versiones con Git para manipular los cambios y el trabajo remoto.
* Implementar operaciones CRUD a los módulos correspondientes tanto para componentes como para XML.
* Documentar constantemente los cambios que se realicen para conservar un código más limpio.

# Justificación

Anterior a la fecha de residencia profesional el procedimiento para saber en qué estatus se encontraban los proyectos, por parte de trabajadores, era a partir de tres pantallas dispuestas de tal forma que en cada línea de producción se observara con claridad que procesos se estaban trabajando, sin embargo debido a la falta de diseño de interfaces era muy fácil perderse y optar por no consultar la información en la pantalla a menos de que se suscitara un paro en la producción.

Por otra parte los clientes que iban a revisar el estado de los proyectos, e inversionistas que planeaban desarrollar sus ideas con la empresa se encontraban descontentos por que la información era difícil de procesar, pues las ventanas no estaban dispuestas de la mejor manera y no existía un gráfico que mostrara de forma rápida que estaba sucediendo.

Este grado de incertidumbre se verá reducido de forma gradual una vez que se implementen las mejoras correspondientes, de las cuales se destacan:

* Una interfaz visualmente atractiva, intuitiva y llamativa, gracias a la selección de colores y componentes que permiten una mejor interpretación aún para personas poco experimentadas con los proyectos que se desarrollan.
* Una mayor fluidez a la hora de navegar por las ventanas en el momento que se desee editar la información, lo que significa un ahorro de tiempo en las pestañas de carga y un mejor control para realizar dichas ediciones
* Segmentación de la información gracias al método de guardado local por medio de un archivo XML evitando contar con datos redundantes en base de datos y a su vez liberar la carga de consultas al servidor, lo que significa un mejor rendimiento del SAP.

Todas estas mejoras se notarán gradualmente en la medida en que se satisfagan las pruebas propuestas por el equipo de desarrollo, el CTO y por supuesto los asesores a cargo. Con esta actualización además es posible que la empresa pueda calificar con los certificados que la abalen ante autoridades importantes en el rubro tecnológico.

Pensando acerca del futuro de esta y otro tipo de aplicaciones, se tiene en consideración la escalabilidad, pues en el código fuente se han realizado acciones adecuadas como:

* Programación orientada a objetos
* Selección de un patrón de diseño escalable y no monolítico.
* Uso de buenas prácticas de programación
* Reutilización de código en la medida de lo posible
* Documentación constante de cada elemento del código.

# Delimitación

Para poder llevar a cabo este proyecto es necesario conocer el entorno de trabajo de Visual Studio y poder entender cómo se integran las herramientas para el trabajo colaborativo, en el caso de este proyecto la herramienta colaborativa utilizada es Git (para gestionar archivos locales) y GitHub (como el almacenamiento remoto o repositorio).

A su vez es muy importante comprender que el lenguaje de programación principal es C# del cual se necesitan conceptos teóricos - prácticos de programación orientada a objetos, comprensión de clases estáticas y abstractas, además se requieren algunos fundamentos de programación funcional. C# contiene una variedad de herramientas para integrar base de datos en sus aplicaciones, por este motivo es necesario conocer el funcionamiento de ADO.Net y comunicaciones con puertos seriales. Para optimizar procesos de carga es necesario implementar programación asíncrona.

En cuestión de diseño de arquitectura de software es necesario comprender que el proyecto deberá seguir una estructura de 4 capas:

• Datos: Comunicación con base de datos.

• Entidades: Abstracción de los elementos físicos en propiedades.

• Negocios: Lógica de negocios del programa donde se realizan operaciones y se preparan los datos para ser mostrados.

• Presentación: Manipulación de los elementos visuales.

Para llevar a cabo la implementación del software será necesario realizar una serie de pruebas funcionales y considerar la retroalimentación de los usuarios para adaptar el programa a sus necesidades, este proyecto solamente será utilizado por la línea de producción y su funcionamiento está limitado a la visualización de información.

En cuanto a aspectos socio-administrativos se sabe que la gerencia no necesitará incurrir en gastos extras para la implementación, pues el esfuerzo más grande recae en cómo se coordinarán las pruebas con el personal.

Los proyectos que se tienen planeados implementar están delimitados por el tiempo en que se cuente con personal de residencia profesional para elaborar el software, en la ilustración 1 se muestra detalle cómo se tiene organizado el tiempo para desarrollar los productos.



Ilustración 1. Cronograma de actividades.

# Marco de referencia

En este apartado se encuentra la base de información que sustenta el desarrollo tecnológico de este proyecto, pasando por bases teóricas relacionadas a (…), posteriormente se enlistarán algunas referencias a proyectos que antecedan al presente sirviendo de base para la selección de tecnologías y prácticas tanto en el desarrollo como la gestión. Finalmente se hará la respectiva mención de cada concepto para ayudar a mejorar la compresión al lector.

## VI.1 Marco teórico

En este apartado se encontrarán los detalles acerca de las teorías en donde se fundamente la investigación para la realización del presente proyecto, se comenzará en detallar aspectos generales hasta ir a conceptos puntuales.

### VI.1.1 Teoría general de sistemas (TGS)

La observación de un concepto fundamental en un área determinada varía de acuerdo a los objetivos de la misma, es decir que cada objeto de estudio derivará de un elemento en particular por ejemplo. Una célula es la base para la biología, un dato para la informática e incluso el pensamiento es la base para estudiar la lógica y el razonamiento. Para representar algunos tópicos complejos de áreas especializadas de la ciencia se volvió indispensable el uso de un modelo de construcción teórica que pueda establecer un nivel intermedio entre un contenido específico (que aporte poco o nulo significado general) y la parte más general (que pueda carecer de contenido) (Johansen Bertoglio, 1993). La presente investigación se sustenta en la TGS debido a que el aplicativo construido formará parte de un sistema de información, formalmente un TGS pretende ser una forma sistemática y científica de aproximación y representación de la realidad, al mismo tiempo, como una orientación hacia una práctica estimulante para formas de trabajo transdisciplinarias (Arnold Cathalifaud & Osorio, 1998).

### VI.1.2 Ingeniería de sistemas computacionales

La TGS funge como un esqueleto para el estudio de las ramas de la ciencia y dentro de estas se encuentran diversas tendencias que apuntan a un contexto aplicativo combinando el pensamiento ingenieril se dio origen a dicha rama interdisciplinar, hace referencia a la planeación, diseño, evaluación y construcción científica de sistemas hombre – máquina (Johansen Bertoglio, 1993). Debido al cambio generacional y a la aparición de los primeros ordenadores surgió una vertiente en la rama de la ingeniería de sistemas que involucraba precisamente las ciencias de la computación adquiriendo los principios de creación hacia elementos no tangibles pero claramente visibles, este apartado teórico es la base fundamental para la creación ya sea de un sistema o parte del mismo basado en los elementos de las ciencias de la computación como lo son la programación, paradigmas, estructuras de datos, arquitectura, entre otros elementos detallados más adelante.

### VI.1.3 Ingeniería de software

Esta rama de las ciencias de la computación aporta un enfoque orientado a la construcción de aplicaciones, es el establecimiento y uso de principios fundamentales de la ingeniería con objeto de desarrollar en forma económica software que sea confiable y que trabaje con eficiencia en máquinas reales (Pressman, 2010). Otra definición que abarca más aspectos es aquella desarrollada por la IEEE en la que se dice la ingeniería de software es la aplicación de un enfoque sistémico, disciplinado y cuantificable al desarrollo, operación y mantenimiento de software; es decir la aplicación de la ingeniería al software (The Institute of Electrical and Electronics Engineers, 1990), se puede deducir que la repercusión de este campo de la ingeniería impacta en la forma de gestionar los recursos para llevar a cabo la concepción del software.

### VI.1.4 Metodología ágil

La metodología de trabaja que se emplea en el desarrollo de software robusto está orientado a modelos estructurados y bien documentados, que en un principio funcionan para equipos expertos y muy bien coordinados, pero cuando se trata de ciclos de desarrollo cortos estas estructuras clásicas de desarrollo presentan ciertos problemas, las metodologías ágiles provienen del manifiesto ágil y pretende optimizar el proceso de creación de software, se valora entonces:

* Al individuo y sus interacciones del equipo de desarrollo sobre el proceso y las herramientas.
* Desarrollar software que funcione más que la documentación del mismo.
* La colaboración con el cliente más que la negociación de su contrato.
* Responde a los cambios más que seguir con el plan establecido.

Además se siguen doce principios para seguir realmente una metodología ágil (López Menéndez de Jiménez, 2015). Dentro de las metodologías ágiles se encuentran Kanban y LEAN como metodologías menos específicas y por tanto más fáciles de adaptar a un equipo de trabajo con relativa facilidad, dentro de este proyecto se pretende adoptar principios de ambas metodologías, especialmente de Kanban, pero sin seguir pautas rígidas (Project Management Institute, 2017).

### VI.1.5 Sistemas basados en transacciones

Son sistemas generalmente localizados en un ordenador específico el cual permite llevar registrar y llevar a cabo transacciones (consultas, altas y bajas) que son necesarias para el desarrollo de una empresa, el objetivo principal de este tipo de sistemas es automatizar el procesamiento de los volúmenes de información mismos que se almacenan en una base de datos o archivos específicos de colecciones (Pítsica Marques, 2001). El proyecto que se pretende llevar a cabo está basado en este tipo de sistema de información debido a que ayuda a automatizar un proceso de visualización de datos con ayuda de un servidor para realizar operaciones de tipo CRUD en un archivo específico evitando la redundancia del guardado de información.

### VI.1.6 N-tier

El desarrollo de un producto de software tiene una implicación muy importante en la estructura o patrón de diseño para implementar una solución, según Refactoring Guru (2014-2020) estos patrones de diseño son considerados como planos prefabricados que se pueden personalizar para resolver un problema un problema de diseño del código de una aplicación, los patrones de diseño cuentan con una serie de secciones específicas como lo son:

* Un propósito
* La motivación
* Estructura
* Ejemplo de código

Partiendo de esta premisa Mark Richards (Richards, 2015) explica que los patrones de diseño de arquitecturas albergan una serie de elementos que las convierten en más o menos escalables, más o menos difíciles de implementar y más o menos difíciles de comprender, la arquitectura de N capas o N-tier tiene la característica de ser un buen elemento de partida a la hora de diseñar software de poco impacto, el autor incluso expresa que es un buen punto de partida para cualquiera que incursione llevando el desarrollo de aplicaciones a un nivel más organizado.

El patrón de N capas no define un límite en específico para organizar la estructura del proyecto, sin embargo es importante mencionar que es idóneo usar 4 capas que pueden tener nombres específicos o se puede optar por nombrarlos como:

* Presentación (presentation).
* Negocios (bussiness).
* Entidades (entities).
* Base de datos (database)

En el presente proyecto se optó por nombrarlas de la misma manera a excepción de la capa de base de datos a la que se le recortó el nombre a “Datos”, además se siguió el principio de nivel de aislamiento que determina que algunas capas pueden estar abiertas para saltar un nivel de capa e ir a otro, este diseño de capas se caracteriza por ser bueno en desarrollo y pruebas.

### VI.1.7 Clean code

Es una serie de pensamientos que tratan de dar un estándar adecuado para generar una alta mantenibilidad del código que se desarrolle, esta manera de desarrollar código permite obtener beneficios como la facilidad de desarrollar pruebas sean del tipo que sean, si se adoptan este tipo de medidas además será mucho más sencillo que cualquier otra persona pueda comprender que está pasando en el software (Suryanarayana, Samarthyam, & Sharma, 2015). Las prácticas de clean code no solo abordan como evitar malas prácticas si no que a su vez se abordan temas de deuda técnica o código hediondo que pudieran comprometer el desarrollo futuro del código, dentro de la aplicación que se trabajó se detectó una deuda técnica orientada a: el diseño, pues carecía de un estándar y todo parecía ejecutarse en una misma clase que combinaba la parte de la interfaz de usuario, el código, debido a que se encontraban múltiples carencias sobre el conocimiento del lenguaje, estándares para nombrar elementos del código y carencia de estructuras de datos para manipular la información, en cuanto a la documentación se encontró un gran déficit, pues no existía una manera de conocer cómo funcionaba cada parte importante del código ya que se incluían únicamente comentarios relacionados a partes fáciles de comprender e incluso un tanto obvias.

### VI.1.8 Sistema de control de versiones

Una parte crucial para llevar un mejor control sobre el desarrollo del proyecto es cuando se implementa un sistema de control de versiones, pues este tipo de sistema puede registrar los cambios realizados en un archivo o conjunto de archivos a lo largo del tiempo (Git, 2014) si se realiza un análisis más profundo sobre cómo funciona el control de versiones, se puede notar que existen diversos tipos de sistemas de entre los cuales destaca uno principalmente por su alta disponibilidad y capacidad de colaboración este es denominado como *sistema de control de versiones distribuido* en el cual se basa Git el cuál destaca debido a que cuenta con una alta demanda por la comunidad de desarrolladores alrededor del mundo debido a que no controla cada cambio específico sino que más bien hace una especie de captura de todos los archivos cada vez que se guardan se optó por utilizar esta tecnología para llevar el control del proyecto debido a que la colaboración aunada con GitHub puede ser altamente eficiente para trabajar en equipo.

## VI.2 Antecedentes

Para conocer de una mejor manera como se aborda el desarrollo de software para empresas, es necesario investigar acerca de caso en los que se haya intentado implementar algún tipo de sistema especialmente un sistema basado en el desarrollo hacia el escritorio, herramientas basadas en Microsoft (SQL Server, ADO.NET/.NET Core), metodologías ágiles, basados en refactorización y control de versiones.

### VI.2.1 Desarrollo e implementación de un sistema de escritorio

Como primer caso que sirve de base para comprender los problemas dentro de las organizaciones relacionados al software se encuentra la tesis llamada “Desarrollo e implementación de un sistema de escritorio para el control y producción” (Merizalde Andrade & Toledo Vallejo, 2013) en donde se menciona que los procesos de control y producción no se llevan a cabo de una manera adecuada a raíz de esto surge cierto nivel de desconfianza por parte de los clientes, un grado de poco crecimiento a nivel empresarial y bajo rendimiento en las ganancias, gracias a la investigación para identificar estos problemas se generó un diagrama de negocios en el que se pudiera definir claramente que elementos debería de contener el sistema en cuestión, para el caso de la arquitectura se hizo uso de una arquitectura por capas en donde se destacan la capa de acceso a datos, capa lógica de negocios, capa de interfaz de usuario, este proyecto utiliza tecnologías de Microsoft para desarrollar el sistemas como SQL Server como RDBMS, Visual Basic como principal lenguaje de programación y el IDE de Visual Studio para desarrollar cada funcionalidad.

### VI.2.2 Diseño de sistema de control y gestión con SQL Server y .NET (uso de LINQ)

Atendiendo a las necesidades tecnológicas de implementar sistemas de información rápidos en el presente trabajo de referencia el autor optó por desarrollar un sistema que pueda recolectar información a través de distintas vistas, reduciendo en gran medida el tiempo de captura de datos, las tecnologías empleadas son SQL Server, ASP.NET y Entity Framework lo que permite el uso de consultas avanzadas para gestionar el flujo de información de una forma más sencilla y rápida que se traduce en un programa que cumple con los requisitos de ser rápido y optimizado (Escudero Cisneros, 2017).

### VI.2.3 Desarrollo de software utilizando metodologías ágiles

El proyecto titulado como “Aplicación de la metodología Kanban en el desarrollo del software para generación, validación y actualización de reactivos, integrado al sistema informático de control académico UNACH (Universidad Nacional de Chimborazo)” (Yépez Llerena & Armijos Guillen, 2020), habla sobre una mala gestión de los reactivos, la palabra reactivo hace referencia a una serie de preguntas que dan la base para otorgar un número a la evaluación de los estudiantes, dado que el enfoque de este trabajo se orienta a los procesos, la metodología Kanban resulta de gran ayuda para eliminar retrasos en el tiempo, se menciona especial atención a los cuellos de botella que pueden surgir a la hora de llevar la supervisión de los reactivos de forma manual, el sistema posee una base de operaciones CRUD y a su vez utiliza la tecnología de Microsoft (ASP.NET), para llevar a cabo este proyecto fue necesario medir el grado de satisfacción con el uso de encuestas un punto importante que podría considerarse para el desarrollo del proyecto de display de pantallas.

### VI.2.4 Refactorización y técnicas de clean code

En el trabajo consultado denominado “Evoluciones y refactorizaciones de una aplicación Android para alfabetización digital” (Pérez Filoche, 2020), se hace la mención sobre una aplicación ya consolidada llamada “TechPeopleCare Android App” la cual tiene una función que permite la descarga de cursos de alfabetización digital y la fácil reproducción de estos, algunos conceptos clave para establecer una buena refactorización como lo es el clean code, bibliotecas de descarga, inyección de dependencias e inclusive un patrón de diseño que difiere con el patrón de diseño MVP (Modelo Vista Presentador). Un enfoque muy importante para esta aplicación es el código limpio, escalable y fácil de comprender pues la misión principal de la refactorización es que el código pueda mantenerse y evitar cambios grandes o un desarrollo desde cero de la aplicación en cuestión, esta práctica es bastante común en el desarrollo a nivel profesional y resulta de gran ayuda que algunas universidades conozcan de su importancia.

### VI.2.5 Control de versiones

Cuando se trata de implementar el trabajo colaborativo existen diversas maneras de implementarlo como es el caso del trabajo denominado “Estudio de sistema de control de versiones y uso de los mismos proyectos de integración continua” (Mompó Camarasa, 2017) este estudio se centra en adaptar una nueva forma de pensamiento frente al software para que se pueda llevar a cabo la implantación de un software más eficiente y con mejores características para llevar los procesos de actualización, este análisis lo llevo a cabo el autor del trabajo gracias a la experiencia personal que le brindaron el servicio social, para facilitar el flujo de trabajo el autor expreso su deseo de apoyarse de un control de versiones, la base de esto fue trabajar con Git un conocido sistema de control de versiones distribuido que funciona a modo local y tiene una facilidad de integrarse con servicios de repositorios en la nube, en el caso del trabajo de este autor incorporó BitBucket, el trabajo se dividió en una rama de funcionalidad, rama de entregas y rama de hotfix (manejo de errores), este trabajo es de ayuda especial para conocer como otros ingenieros resuelven los problemas de control de versiones.

## VI.3 Marco conceptual

Dentro de este apartado el lector podrá encontrar los conceptos puntuales que dan origen al desarrollo del sistema y que son realmente esenciales para comprender el objetivo de llevar a cabo la implementación del software, pasando de lo más genérico a conceptos muy específicos.

### VI.3.1 SAP ERP

La relación del uso de sistemas informáticos para entornos empresariales ha estado presente desde el principio de la era digital ya que cuenta con un factor de innovación y captación del público impresionantes, existen diversas corrientes del pensamiento, teorías y principalmente estrategias de marketing que apuntan a la comercialización con las empresas pudiendo traer valor agregado a elementos que ya existen (como la regulación del tiempo, gestión de recursos, control de personal, etc.) es por este motivo que algunos equipos de trabajo en el área de desarrollo han optado por desarrollar software específico para empresas. Cuando se habla de SAP (System Analysis and Program Development) se hace referencia a la compañía que en 1972, cinco trabajadores de IBM dieron a conocer al mundo gracias a la idea de crear software con estándares empresariales que integraran todos los procesos de negocio y permitieran procesar datos en tiempo real (SAP, 2020). Por su parte la abreviatura ERP (Enterprise Resource Planning) hace alusión al software que tiene por objetivo facilitar la gestión de todos los recursos de la empresa a través de la integración de la información de los distintos departamentos y áreas funcionales (Suárez Rey, 2010).

### VI.3.2 Escalabilidad

Es importante recalcar que una aplicación sea pequeña o grande en el momento que se despliega pasados algunos años pueden surgir más necesidades para resolver y es el deber del equipo de desarrollo dejar un sistema escalable. La escalabilidad es la habilidad que tiene un sistema para para adaptarse al crecimiento sin perder calidad en los servicios que se ofrecen, es de suma importancia mencionar que la capacidad de crecimiento puede darse de dos maneras: scale in (escalado vertical) que hace referencia al crecimiento en un solo nodo, por otro lado el scale out (escalado horizontal) que se enfoca en el aumento de nodos para liberar la carga del sistema (Gonzalez Godino, 2013).

### VI.3.3 Refactorización

Cuando se diseña el software normalmente se debería pensar en crearlo orientado hacia la mantenibilidad del código fuente, una de las principales reglas para llevar a cabo este cometido se trata de escribir partes simples para lograr un cambio en la estructura interna para lograr que este sea más sencillo de comprender y sobre todo fácil de modificar sin alterar el comportamiento normal. Generalmente las técnicas de refactorización de código se aplican después de que el funcionamiento de un módulo ha sido desarrollado, pasa el tiempo y al revisar el código se encuentra una difícil comprensión del funcionamiento o simplemente el producto ha crecido y necesita una mejor forma de escalar (Dooley, 2017).

### VI.3.4 Procedimiento almacenado

Es un grupo de una o más instrucciones en lenguaje SQL los procedimientos se asemejan a las construcciones de otros lenguajes de programación, porque pueden: aceptar parámetros de entrada y devolver varios valores en forma de parámetros de salida al programa que realiza la llamada, contener instrucciones de programación que realicen operaciones en la base de datos, inclusive pueden contener llamadas a otros procedimientos, devolver un valor de estado a un programa que realiza una llamada para indicar si la operación se ha realizado correctamente o se han producido errores, y el motivo de estos, el uso de procedimientos almacenados acarra una serie de beneficios entre los cuales destacan el bajo uso del tráfico de red, una mayor seguridad ante inyecciones SQL (además es posible cifrar un procedimiento almacenado), mejora el mantenimiento y la usabilidad debido a que un solo procedimiento almacenado puede ser utilizado múltiples veces (Microsoft, 2020).

### VI.3.5 Colección de datos

Para que un programa cuente con una alta eficiencia debe incluirse un buen manejo de la memoria aplicando estructuras de datos o colecciones de datos que no son más que un conjunto de técnicas que aumentan considerablemente la productividad del programa, reduciendo en elevado grado, el tiempo requerido para escribir, verificar, depurar y mantener los programas. El término estructura de datos hace referencia a un conjunto de datos que, por medio de un nombre, identifican un espacio en memoria, teniendo ciertas características como la organización y estructuración, permitiendo realizar operaciones definidas en ellas. Las estructuras de datos pueden ser de dos tipos: estructuras de datos estáticas (aquellas que cuentan con un tamaño definido), estructuras de datos dinámicas (aquellas en las que el tamaño puede variar en tiempo de ejecución) (Luna López, 2012).

### VI.3.6 CRUD

Es un acrónimo para las maneras en las que se puede operar sobre información almacenada. Es un nemónico para las cuatro funciones del almacenamiento persistente. CRUD usualmente se refiere a operaciones llevadas a cabo en una base de datos o un almacén de datos este acrónimo deriva de las palabras: Create (Crear), Read (Leer), Update (Actualizar) y Delete (Eliminar) (Mozilla and contribuitors, 2019).

### VI.3.7 Interfaz de usuario

Para que un usuario ordinario se pueda comunicar con un ordenador o aplicación del ordenador hace falta contar con un conjunto de elementos de hardware y software que presenten la información de una forma entendible apoyándose de gráficos, colores, tipografía presentados en forma de ayuda, manuales, tutoriales, etc. Si la interfaz de usuario está perfectamente diseñada el usuario podrá navegar por la aplicación u ordenador sin problema alguno encontrando las respuestas a cada acción que realice (Sebastián Gómez, 2000).

### VI.3.8 Programación orientada a objetos

Conocido como un paradigma de programación, son un conjunto de técnicas que nos permiten incrementar enormemente nuestro proceso de producción de software; aumentando drásticamente nuestra productividad por un lado y permitiéndonos abordar proyectos de mucha mayor envergadura (Morero, 2000), la POO se encarga de definir un objeto como una serie de métodos y atributos que se encontrarán definidos en una clase (abstracción de la realidad, sirve como una especie de plano) al utilizar un paradigma de programación orientado a objetos es posible observar un grado de reusabilidad en el código que van de la mano con otra serie de conceptos que son:

* **Herencia:** Destaca la cualidad de contar con clases padre e hijo para reutilizar elementos de las clases derivadas.
* **Encapsulación:** Cuando se reúnen todos los elementos pertenecientes a una misma entidad.
* **Polimorfismo:** Comportamientos similares asociados a objetos distintos que pueden compartir este funcionamiento con elementos distintivos de cada uno.

### VI.3.9 IDE

Un entorno de desarrollo integrado (IDE) es un sistema de software para el diseño de aplicaciones que combina herramientas del desarrollador comunes en una sola interfaz gráfica de usuario (GUI) (Red Hat Inc., 2020). Generalmente, un IDE cuenta con las siguientes características:

* **Editor de código fuente:** editor de texto que ayuda a escribir el código de software con funciones como el resaltado de la sintaxis con indicaciones visuales, el relleno automático específico del lenguaje y la comprobación de errores a medida que se escribe el código.
* **Automatización de compilación local:** herramientas que automatizan tareas sencillas e iterativas como parte de la creación de una compilación local del software para su uso por parte del desarrollador, como la compilación del código fuente de la computadora en un código binario, el empaquetado del código binario y la ejecución de pruebas automatizadas.
* **Depurador:** programa que sirve para probar otros programas y mostrar la ubicación de un error en el código original de forma gráfica.

### VI.3.10 Visual Basic

Es un lenguaje de programación orientado a objetos desarrollado por Microsoft, utilizar visual basic puede aumentarla velocidad de creación de aplicaciones con .NET, la palabra Visual hace referencia al método de creación de la GUI (método que otros lenguajes ya han optado gracias a los IDEs), por otra parte la palabra BASIC es una referencia al lenguaje del mismo nombre que significa “Código de instrucciones simbólicas multipropósito para novatos” (Microsoft, 2020). Esta forma de desarrollar aplicaciones tiene algunas limitantes como la mantenibilidad del código que aumenta en relación a la envergadura de los proyectos y temas de escalabilidad previamente mencionados, cabe mencionar que este lenguaje fue la base para crear la lógica del primer programa para la empresa.

### VI.3.11 C#

Es el lenguaje de programación por excelencia desarrollado por Microsoft que corre tanto en .NET core como en .NET framework, está orientado al paradigma de programación orientado a objetos pero comparte una clara similitud con C y Java, este lenguaje ha sido utilizado para desarrollar aplicaciones web, móviles, juegos, etc. (Microsoft, 2020). Este lenguaje es ampliamente utilizado junto al IDE Visual Studio un entorno de desarrollo en donde es posible aplicar todas las técnicas de estructura de un patrón de diseño y a su vez buenas prácticas de programación, el IDE cuenta también con herramientas de depuración y testeo lo que lleva a pensar en un desarrollo bastante completo.

### VI.3.12 SQL Server

A medida que las aplicaciones informáticas comenzaron a popularizarse, se observó la vertiente de contar con datos redundantes para un conjunto de aplicaciones específico, por ejemplo un nombre de producto podría estar en el inventario y al mismo tiempo en el detalle de venta desde ese momento fue necesario comenzar a utilizar conjuntos de ficheros interrelacionados que implementaban estructuras complejas que a su vez, podían ser compartidos por varios procesos de manera simultánea a este descubrimiento se le conoce como bases de datos, de una manera más formal se puede decir que una base de datos es la representación integrada de los conjuntos de entidades instancia correspondientes a las diferentes entidades tipo del sistema de información y de sus interrelaciones. Esta representación informática (o conjunto estructurado de datos) debe poder ser utilizada de forma compartida por muchos usuarios de distintos tipos (Camps Paré, y otros, 2005). Para llevar a cabo el manejo de una base de datos más tarde surgió el concepto de DBMS (Data Base Management System) que básicamente es un gestor de base de datos percibido por el usuario como tablas que contienen dos dimensiones en donde se representa la información concepto al que posteriormente se le agrego una parte más complejo pero altamente eficiente para trabajar con un gran conjunto de datos al que se le aplican conceptos más avanzados sobre atomicidad, consistencia, aislamiento y durabilidad (Gómez Fuentes, 2013). SQL Server es un RDBMS (Relational Data Base Management System) desarrollado por Microsoft que a partir de 2016 ha dado soporte no solo para Windows sino que también para Linux y contenedores lo que permite que pueda escalar de una forma natural, la propia empresa optó por utilizar este RDBMS debido a sus buena integración con C# y su facilidad para crear conexiones hacia el servidor (Gorman, y otros, 2019).

### VI.3.13 ADO.NET

Para llevar a cabo la conexión entre una aplicación y una base de datos se han implementado diversos métodos de conexión de acuerdo a las necesidades del desarrollador, para seguir el estándar ya establecido en la empresa se optó por utilizar conexión de tipo ADO.NET, es importante mencionar que este tipo de conexión es la forma más simple de conectarse a una base de datos, una conexión simple puede utilizar una sola referencia a System.Data.SqlClient y satisfacer todas sus necesidades (Stack Overflow Comunity, 2017), el modo de acceso para una aplicación se puede apreciar en la figura posterior (ver ilustración 2).

Ilustración 2. Proceso de ejecución con ADO.NET (gráfico propio).

### VI.3.14 DLL

Es una librería que contiene datos que pueden ser usados por más de un programa al mismo tiempo, las siglas significan biblioteca de enlace dinámico (Dynamic Link Library) esta forma de crear pequeños fragmentos de código que puede ser reutilizable aumenta en gran medida la eficiencia de la memoria debido a que el DLL carga únicamente cuando se necesita la funcionalidad, otro de los beneficios está en que se promueve una arquitectura modular que permite que los cambios se hagan con mayor facilidad, los desarrolladores que implementen nuevas funcionalidades además tendrán un mejor control sobre que implementar y la característica principal por la cual se destaca es su facilidad de implementación ya que basta con tener la referencia para ocupar la funcionalidad del DLL (Microsoft, 2020).

### VI.3.15 XML

El Lenguaje de Marcado Extendido es un tipo de especial de lenguaje utilizado principalmente para guardar información en un modo más sencillo (Dykes & Tittel, 2005), se compone principalmente de etiquetas propias con nombres en de cadena o tokens especiales, cada tag o etiqueta debe contener un cierre y puede contener cero o más atributos que identifiquen a la etiqueta con un valor único. Otra característica importante del lenguaje de marcado es que puede albergar comentarios para facilitar su comprensión y una característica notoria es que una etiqueta puede tener el mismo nombre y dentro contener elementos similares o distintos, es el mismo caso para el contenido de cada etiqueta (Landa Cosio N. A., 2018) (ver ilustración 3).

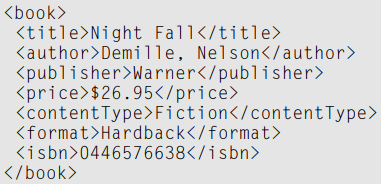


Ilustración 3. Ejemplo de documento XML (obtenido de Dykes, L., & Tittel, E. (2005). Ejemplo de clasificación con XML. Recuperado de XML for Dummies. Hoboken: Wiley Publishing).

### VI.3.16 LINQ

Para comprender de qué se trata este concepto es necesario explicar en primer lugar lo que es el mapeo Objeto/Relacional (ORM), se refiere a la persistencia automatizada y transparente de las tablas en una base de datos relacional, usando metadatos que definen el mapeo entre los objetos y la base de datos, dichos objetos pueden estar orientados únicamente a una estructura de datos (arreglo, matriz, etc.), un origen de datos de BD o inclusive utilizarse con XML, este último apartado requiere de algunos elementos especiales para funcionar listados a continuación:

* **XObject:** Clase abstracta que es la base para todo el contenido XML.
* **XNode:** Clase base para la mayoría del contenido XML, excepto atributos.
* **XContainer:** Define miembros para trabajar con sus hijos y es la clase padre de XElement y XDocument
* **XElement:** Define a un elemento XML.
* **XDocument:** Representa la raíz de un árbol XML, en realidad envuelve a un XElement que actúa como raíz y lo podemos usar para adicionar la declaración e instrucciones de procesamiento.

Es importante mencionar además que es necesario utilizar System.Xml.Linq y no System.Linq (Landa Cosio N. , 2018), Linq es la base lógica del negocio por lo que es importante comprenderlo si se requieren realizar modificaciones a ese nivel de comprensión, a continuación se listan ejemplos de escritura (ver ilustración 4 y 5).

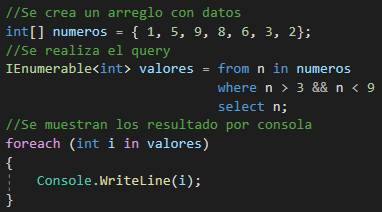


Ilustración 4. Linq para objetos (gráfico propio).

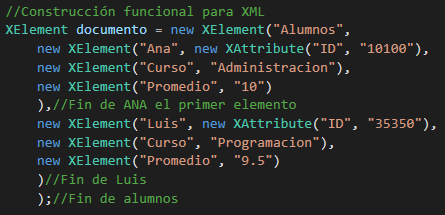


Ilustración 5. Linq to XML bajo escritura funcional (gráfico propio).

# Metodología

En este aparatado se explica a detalle la organización de recursos, requerimientos y procedimientos que permiten realizar el presente proyecto de desarrollo de software, en un principio se mencionará sobre el tipo de metodología ágil, que beneficios trae al desarrollo que hacen optimo su uso en contra de una metodología tradicional, posteriormente se enlistarán algunas metodologías de las cuales se han obtenido elementos específicos y porque no se ha optado por seguir una metodología al pie de la letra, en el apartado final se hará mención del proceso que se ha seguido con ilustraciones que ayuden a corroborar este proceso y además se incluye un apartado sobre la viabilidad técnica, económica y operativa.

Cuando se piensa en comenzar el desarrollo de un proyecto, existen diversas maneras de atacar el problema, generalmente un programador optaría por desarrollar una solución lo antes posible para realizar cambios y ajustarse a las necesidades del cliente esto quiere decir que se enfoca a los requerimientos únicamente e ignora por completo aspectos como el tiempo, costo y procedimientos externos involucrados. En caso de que una persona que gestiona proyectos que no son precisamente de software pretenda llevarlo a cabo se notará una mejora en el control de recursos y sabrá diferenciar que procesos externos intervienen, sin embargo la implementación de cambios en los requerimientos le supondrá un reto enorme pues no está acostumbrado a los cambios repentinos y su gestión del proyecto podría tomar el doble de tiempo si comenzara a realizar esos cambios desde un inicio.

De acuerdo con Pressman (2010), la ingeniería de software proporciona los elementos necesarios para llevar un enfoque no solo eficaz, sino que también eficiente pues se centra en aportar principios fundamentales divididos en principios que guían el proceso y principios que guían la práctica (ver tabla 1).

Tabla 1. Principios fundamentales de la ingeniería de software (elaboración propia).

|  |  |  |
| --- | --- | --- |
|  | Principios que guían el proceso | Principios que guían la práctica |
| Significado | Establece un fundamento filosófico para guiar actividades estructurales | Definen un conjunto de valores y reglas que sirven para analizar un problema, diseñar una solución, implementar, probar y entregar. |
| Principios | **Ser ágil.** Mantener un enfoque técnico sencillo tanto como sea posible, generar productos concisos, tomar decisiones locales en la medida de lo posible | **Divide y vencerás.** Enfatizar la separación de entidades, un problema grande se vuelve más sencillo si se divide en pequeñas entidades. |
| **Centrarse en la calidad.** Aplica para toda actividad, acción y/o tarea del proceso | **Entender el uso de la abstracción.** Conocer primero los detalles antes de utilizar la abstracción en un nivel alto o bajo. |
| **Estar listo para adaptar.** Es normal cambiar el enfoque a las restricciones impuestas, no hay una fórmula para hacer las cosas | **Buscar coherencia.** Aplicar un contexto familiar hace que el software sea más fácil de usar. |
| **Formar un equipo eficaz.** Un equipo de confianza, organizado y comunicado. | **Centrarse en la transferencia de información.** Poner atención especial al flujo de la información dentro del análisis, diseño, construcción y prueba. |
| **Establecer mecanismos para la comunicación y coordinación.** Es de vital importancia para que el proyecto no falle | **Construir software con modularidad eficaz.** Centrar cada módulo en un aspecto delimitad del sistema y su conexión debe ser sencilla. |
| **Administrar el cambio.** Deben establecerse mecanismos para administrar la forma en que los cambios se solicitan, evalúan, aprueban e implementan. | **Buscar patrones.** Ayudan a formar un cúmulo de información para futuras referencias. |
| **Evaluar el riesgo.** Es necesario establecer planes de contingencia. | **Cuando sea posible representar el problema y su solución desde varias perspectivas.** Es más probable que se tenga mayor visión, lo que ayuda a detectar errores y omisiones. |
| **Crear productos con valor para otras actividades.** El producto de trabajo imparte la información necesaria sin omisiones ni ambigüedades. | **Tener en mente que alguien dará mantenimiento al software.** Es importante aplicar técnicas de ingeniería de software para asegurar que el software tenga mantenimiento y no una refactorización completa. |

Independientemente si se utiliza una metodología ágil o tradicional, estos principios deben estar parcialmente cubiertos especialmente cuando se combinan aspectos de distintas metodologías.

La metodología de la que parte el proyecto es una serie de elementos de kanban, una metodología ágil caracterizada por contar con una rápida adopción en proyectos en donde el equipo de desarrollo no supera los 10 integrantes (incluso puede ser usada para una sola persona), kanban generalmente está orientado a mejorar el control del flujo de trabajo, según Anderson & Carmichel (2016) Kanban sigue el modeo de pensamiento de “*Comienza por donde estés*” siendo así una metodología centrada en el cambio rápido y focalizado.

Para hacer uso de esta metodología se utiliza un sistema kanban el cual permite controlar el flujo de entrega limitando la cantidad de trabajo en progreso con ayuda de una serie de tarjetas denominadas kanbans distribuidas en un tablero kanban (ver ilustración 6) que no es más que una pizarra digital o física con una serie de divisiones que no están fuertemente definidas siempre y cuando existan los elementos esenciales de Kanban.

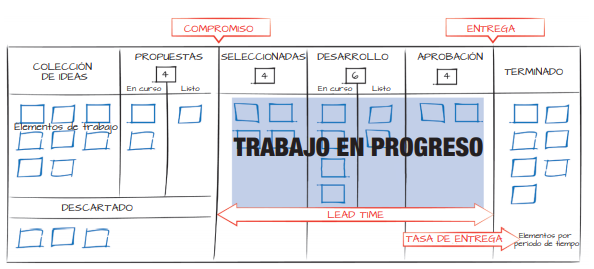


Ilustración 6. Ejemplo de tablero kanban (extraído de Anderson & Carmichel, 2016).

* **Reconocer el trabajo a futuro:** Que requerimientos son los solicitados, como serán implementados y si existen casos en los que no se atenderán esos requerimientos por diversos motivos.
* **Identificar el trabajo en progreso:** Una serie de columnas en donde se expresa que trabajo está realizando cada miembro del equipo.
* **Establecer políticas:** En este apartado se incluye la delimitación del WiP (Work in Progress), nivelación de carga, definición de columnas y políticas de retroalimentación.

Se adoptaron estos principios para la gestión del proyecto debido al enfoque iterativo, incremental y orientado a la mentalidad del manifesto ágil se optó por tomar elementos importantes de Kanban sin seguir una pauta estricta del mismo, a continuación se puede observar la implementación de un tablero kanban tradicional con ayuda de la herramienta Trello (ver ilustración 7).

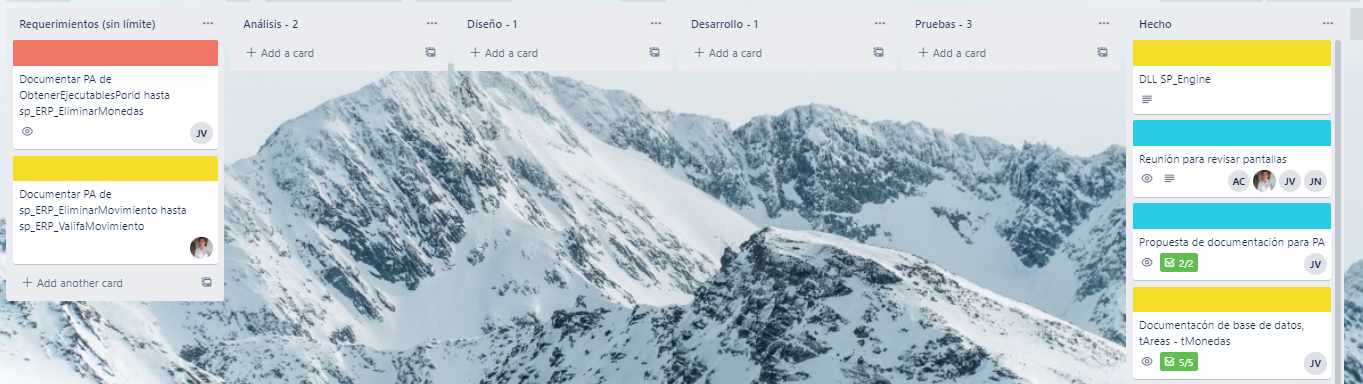


Ilustración 7. Tablero kanban para el proyecto (ilustración propia).

Los elementos de las columnas están organizados de la siguiente manera:

* **Requerimientos (sin límite):** En este apartado se colocan los nuevos requerimientos que vayan surgiendo y se analiza que tan viable es implementarlos o posponerlos.
* **Análisis (2):** Para esta columna solo se pueden colocar dos elementos en progreso, debido a que como el equipo está conformado por dos desarrolladores supervisados por un CTO el análisis puede estar dirigido únicamente por alguno de los dos desarrolladores apoyado en el CTO.
* **Diseño (1):** Uno de los desarrolladores está enfocado mayormente en el tema de frontend (aplicaciones del lado del servidor) y cuenta con las habilidades necesarias sobre desarrollo de interfaces de usuario.
* **Desarrollo (1):** Debido a que el miembro del equipo restante basa su experiencia en aplicaciones para el backend solo puede existir un trabajo para la columna del desarrollo en cada momento de tiempo.
* **Pruebas (3):** En esta sección se coloca un mayor WiP debido a que además de los desarrolladores el CTO puede apoyar en la tarea de pruebas e incluso llevar a cabo pruebas de integración y no solo unitarias.
* **Hecho:** Esta última columna está dedicada a colocar las secciones terminadas del proyecto.

Además se implementó un mini tablero Kanban por cada miembro del equipo (en caso de que fuera necesario) al cual se le agregaron las subcolumnas de proceso y listo para el diseño, desarrollo y pruebas (QA) (ver ilustración 8). Para detallar más el proceso de desarrollo de la metodología se mostrará en un gráfico que contemple la creación de un único módulo con una o varias subtareas (ver tabla 9).



Ilustración 8. Mini tablero Kanban (gráfico propio).

Tabla 2. Pasos para llevar a cabo el desarrollo de elementos específicos (gráfico propio).

|  |  |  |
| --- | --- | --- |
| Paso | Elemento unitario | Elementos integrados o módulos |
| Verificar requerimiento | Cuando un requerimiento llaga se pone en la pila de trabajo y se abordan en ese orden, no debe pasar mucho tiempo para decidir si la tarjeta kanban se utilizará o no. | Se verifica que una funcionalidad pueda ser segmentada en más partes para resolver cada una de estas de forma particular. |
| Analizar el tiempo de desarrollo | De acuerdo con el nivel de prioridad que el CTO haga conocer se decide que requerimiento trabajar primero y cuáles son los elementos necesarios por conocer antes de comenzar a desarrollar. | Analizar el conjunto de requerimientos unitarios para estimar un tiempo de trabajo total sumando el tiempo de capacitación en caso de requerir una capacitación |
| Diseñar parte gráfica o lógica | El diseño gráfico incluye la selección de colores, formatos de imagen, iconos, distribución del contenido, medidas y selección de cada componente gráfico para la muestra de información. Si se trata de diseño lógico se hace referencia a la estructura del código, carpetas o métodos, el proceso de funcionamiento se detalla generalmente mediante un dibujo o gráfico para apreciar la idea. | Corroborar que el diseño total cuadre con los diseños específicos. |
| Desarrollar funcionalidad | Esta puede ser de tipo frontend o backend de acuerdo al tipo se trabajara con la capa del patrón de diseño que le corresponda. | Si el conjunto de un módulo desempeña sus funcionalidades individuales este apartado está completo |
| Probar elementos | Este apartado verifica que el requerimiento haya sido cumplido satisfactoriamente, de lo contrario se hace regresa al comienzo para revalorar la solución, se suele emplear la depuración de código. | La prueba de integración o de endpoint suele ser la más usada para verificar que un módulo efectivamente cumpla su función. |

## VII.1 Viabilidad

Se mostrarán algunos puntos que permiten demostrar la viabilidad del proyecto basado en la experiencia de la empresa.

### VII.1.1 Técnica

Al desarrollar este proyecto se hizo hincapié en algunos aspectos técnicos, como la organización e implementación de una estructura bien definida (conocida como de n capas o n-tier) lo que se puede traducir como un trabajo que debería desarrollar unas mejores características de escalabilidad, pues el uso de este tipo de software le dan mayores oportunidades de crecimiento no solo dentro de un nivel operativo, sino que también dentro de una mejor calidad como profesionistas que conocen y saben manipular herramientas de desarrollo. Se visualiza un panorama favorable dentro del uso de este proyecto, pues sin duda era una necesidad que el software tenía que ser actualizado y no es la única vez que ha trabajado con este tipo de equipos de trabajo, por lo que la comunicación no se mostró como un problema.

### VI.1.2 Económica

El uso de software empresarial a medida trae consigo una serie de gastos económicos mismos que pueden ser muy rentables dependiendo de la cantidad de demanda que se quiera cubrir, sin embargo con los sucesos del año 2020 se empezó a dar a conocer una nueva vertiente sobre la capacidad de llegar de forma remota hacia el público objetivo, es por ello que el desarrollo de software para empresas ayuda en gran medida a invertir los recursos económicos en herramientas que faciliten el trabajo a distancia, dado que el software que se implementó contiene código de versiones recientes, se estima que el momento en que se necesite de una actualización sea menor a 4 años dependiendo del crecimiento de la demanda de productos y que este tipo de actualización puede resultar bastante rentable en el futuro pudiendo llegar a durar hasta 10 años sin realizar una modificación de todo el código fuente representando un ahorro significativo para la empresa.

### VI.1.3 Operativa

Debido a que los sistemas que se solían utilizar en línea de producción dentro de la empresa se encuentran en paro, sería ampliamente recomendable realizar actualizaciones y mejoras dentro de las instalaciones, pues el periodo de prueba es adecuado para detectar errores antes de ingresar a producción, aunado a esto la capacitación para el uso del software puede ser más productiva pues al no tener un espacio tan concurrido es más sencillo que los empleados expresen sus inquietudes. No se observa un punto en el que el software deje de ser utilizado, sino por el contrario es muy probable que con el paso del tiempo se requiera mayores capacidades que agilicen la tarea visual de los trabajadores, el único detalle que podría limitar su uso es en respecto a la capacidad de producción con la que cuenta la empresa debido a la pandemia global por COVID-19.

# Resultados

En el presente apartado se hará mención de los comentarios finales que dejo como resultado el desarrollo del proyecto, se mencionan algunas conclusiones relacionadas con los objetivos planteados al inicio del presente documento, posteriormente se darán a conocer algunas recomendaciones para aquellas personas que deseen retomar el desarrollo para la empresa, se muestra la bibliografía consultada y que hace referencia a algunos temas específicos de la investigación relacionada con la parte teórica, se encuentra además un glosario para identificar aquellos conceptos que no son tan sencillos de interpretar para el público general, en el apartado final se muestra material que sirve de referencia al proyecto, pero que no puede estar descrito en ninguno de los apartados principales, como es el caso del manual de usuario, descripción de clases y arquitectura.

A continuación se enlistará el requerimiento específico y la acción que se toma en cuanta, posteriormente se mostrará una imagen en donde se visualice el cambio en el desarrollo del programa.

## VIII.1 Objetivo específico 1.

Reestructurar el patrón de diseño de la aplicación para aumentar su escalabilidad utilizando un modelo de n-capas (presentación, negocio, entidades y datos). Con ayuda de Visual Studio se realizó una estructura basada en bibliotecas de clases en las que se siguiera dicho patrón de clases, esta fue la base del desarrollo básicamente sin un patrón de diseño la refactorización de la aplicación no hubiese tenido sentido (ver ilustración 9)

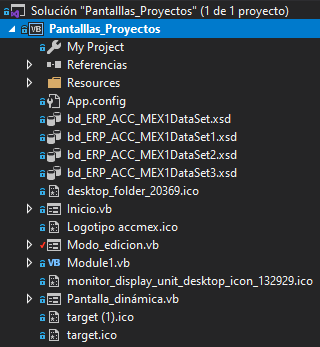
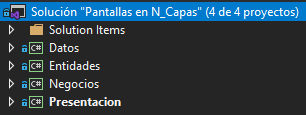
 

Ilustración 9. Comparativa entre patrones de diseño (derechos reservados a ACC MEX S.A. de C.V.).

## VIII.2 Objetivo específico 2

Analizar los requerimientos respecto al sistema actual, conservando el objetivo principal de la aplicación (informar sobre el estado en producción). Se conservó gran parte de la funcionalidad del programa, tratando de visualizarlo como una estructura de bloques para abstraer cada parte de la funcionalidad y volver a crearla (ver ilustración 10 y 11).

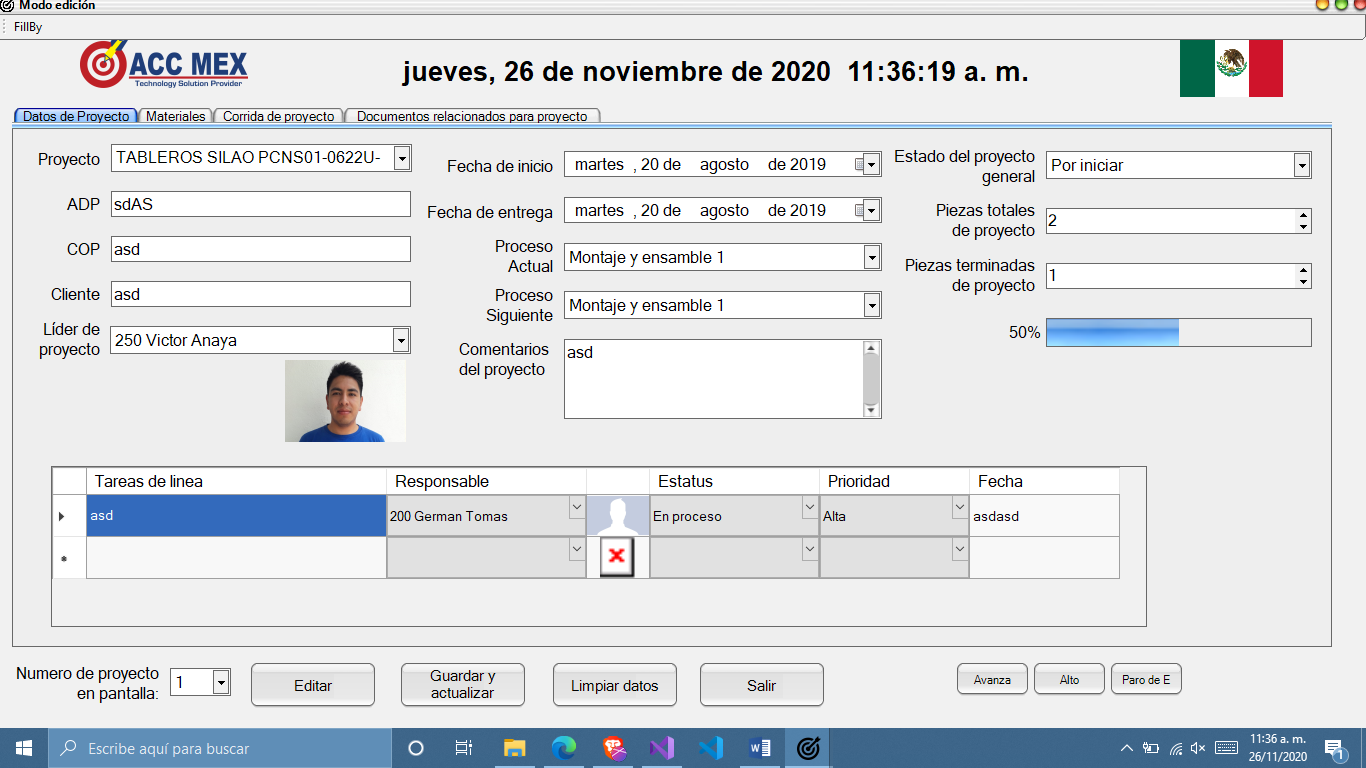


Ilustración 10. Programa original (derechos reservados a ACC MEX S.A. de C.V.)

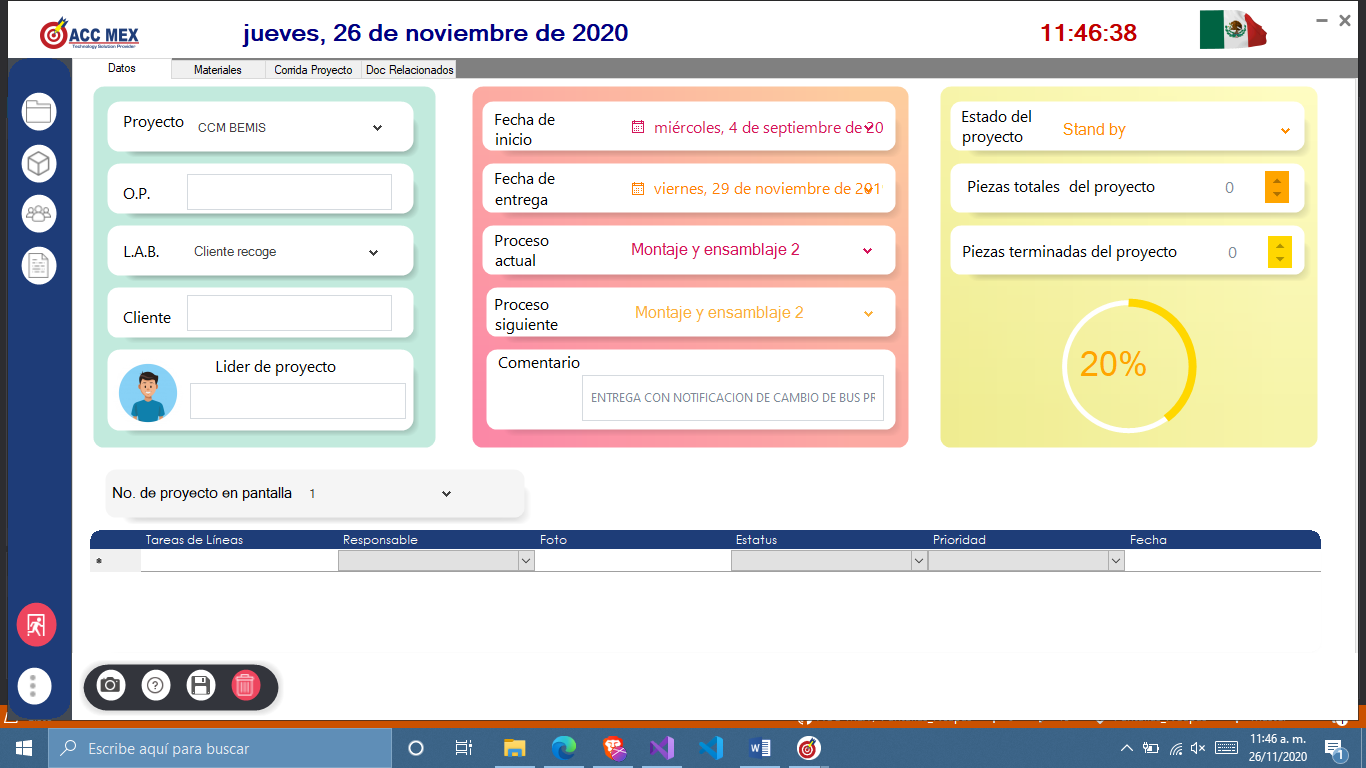


Ilustración 11. Programa refactorizado, se conservaron muchos aspectos funcionales (derechos reservados a ACC MEX S.A. de C.V.).

## VIII.3 Objetivo específico 3

Conocer el entorno de desarrollo en C# para realizar una refactorización de código desde VB. Fue necesario realizar una capacitación por cuenta propia de cada miembro del equipo de trabajo en el entorno de desarrollo de Visual Studio y las tecnologías que utiliza para el desarrollo de aplicaciones de escritorio, el lenguaje de Visual Basic solo sirvió de referencia para conocer como estaba construida la programación (ver ilustración ).

Ilustración 12. Marco de trabajo general (gráfico propio).

## VIII.4 Objetivo específico 4

Implementar un diseño orientado a UI/UX para facilitar la comprensión de la información El diseño orientado hacia una experiencia que los usuarios encuentren familiar e intuitiva facilita muchos temas de comprensión es por ello que la interfaz tiene una paleta de olores que facilite encontrar los elementos dispuestos en pantalla, controles con algunas animaciones para resaltar elementos y uso de programación asíncrona (ver ilustración 13 y 14).

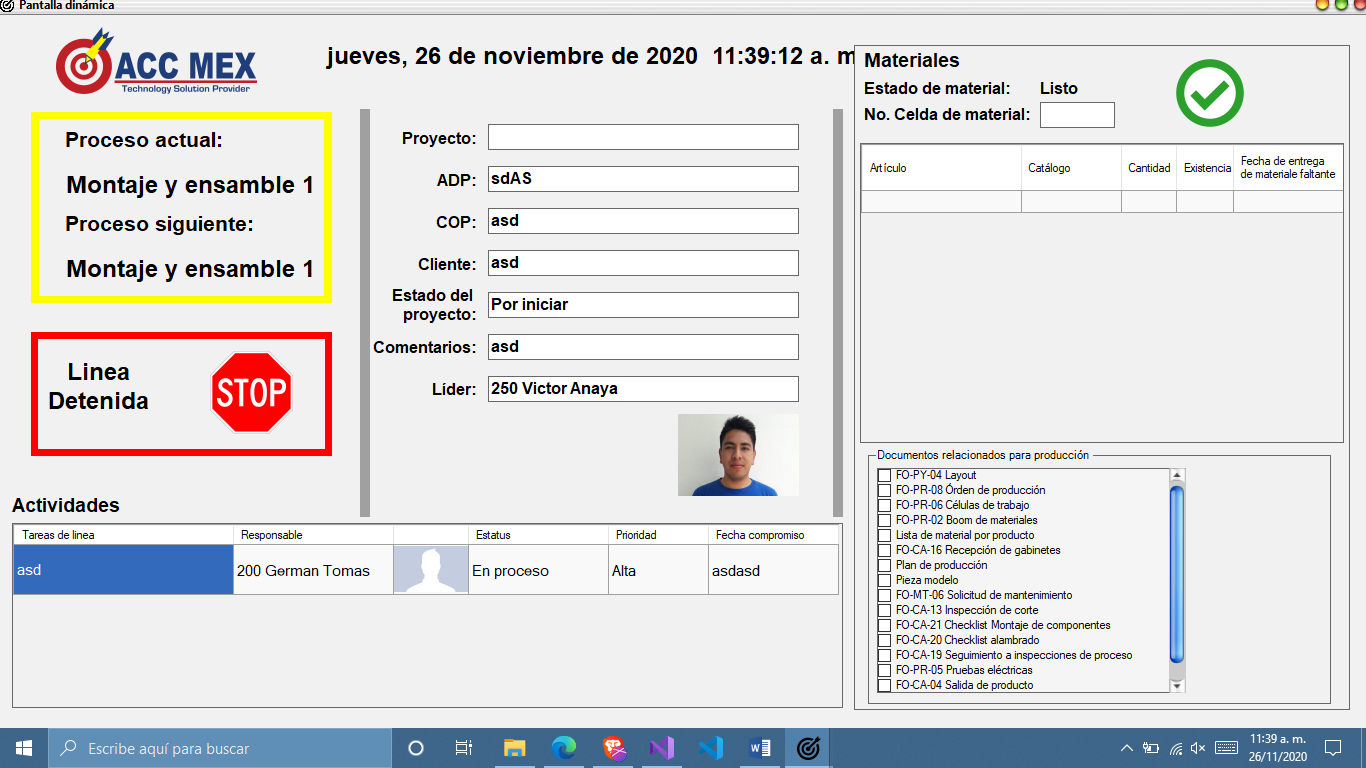
 

Ilustración 13. Aspecto gráfico original (derechos reservados a ACC MEX S.A. de C.V.).

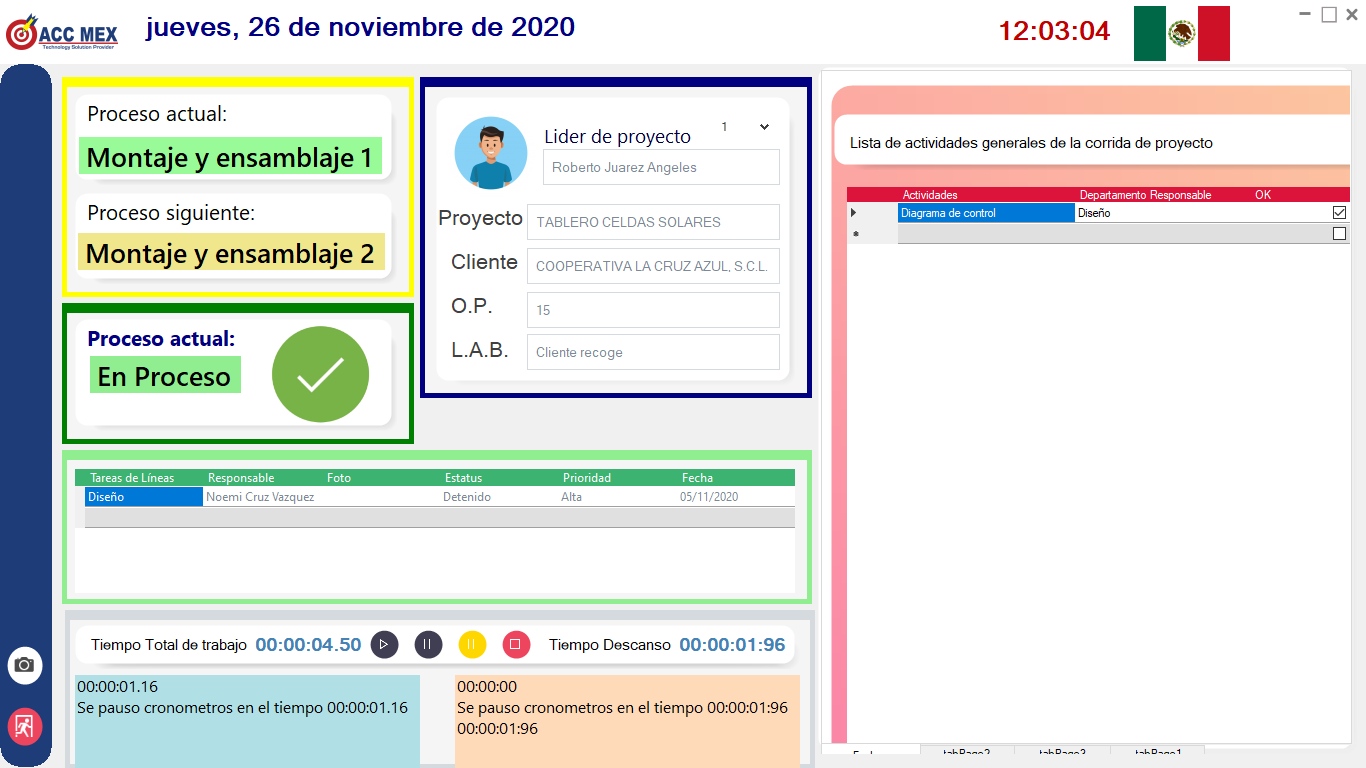


Ilustración 14. Apartado gráfico actualizado (derechos reservados a ACC MEX S.A. de C.V.).

## VIII.5 Objetivo específico 5

Segmentar correctamente el origen de datos local con archivos de configuración y de base de datos utilizando ADO.NET para evitar redundancia. Existe una pequeña diferencia entre el origen de datos y el lugar donde se almacenarán los datos, en este proyecto particular la primera parte se realiza con datos de base de datos y archivos de configuración que contienen datos predeterminados (ver ilustración 15).

Origen de datos

Almacenamiento de datos

Ilustración 15. Segmentación de los datos (gráfico propio).

## VIII.6 Objetivo específico 6

Crear e implementar un DLL para manipular procedimientos almacenados de una manera más amena. El uso de procedimientos almacenados demanda una conexión activa al servidor que posteriormente se cierra, utilizar ADO.NET como método de conexión puede llegar a mantener viva la sesión en más de una vez si no se manipulan correctamente los recursos, es por ello que se desarrolló un DLL que pudiera manipular ese acceso a datos de forma más sencilla y sobre todo asegurándose de que se elimina la conexión a SQL Server, un punto importante es que gracias al DLL es posible implementar otro tipo de conexión hacia otra base de datos en caso de que se quisiera migrar (ver ilustración 16).

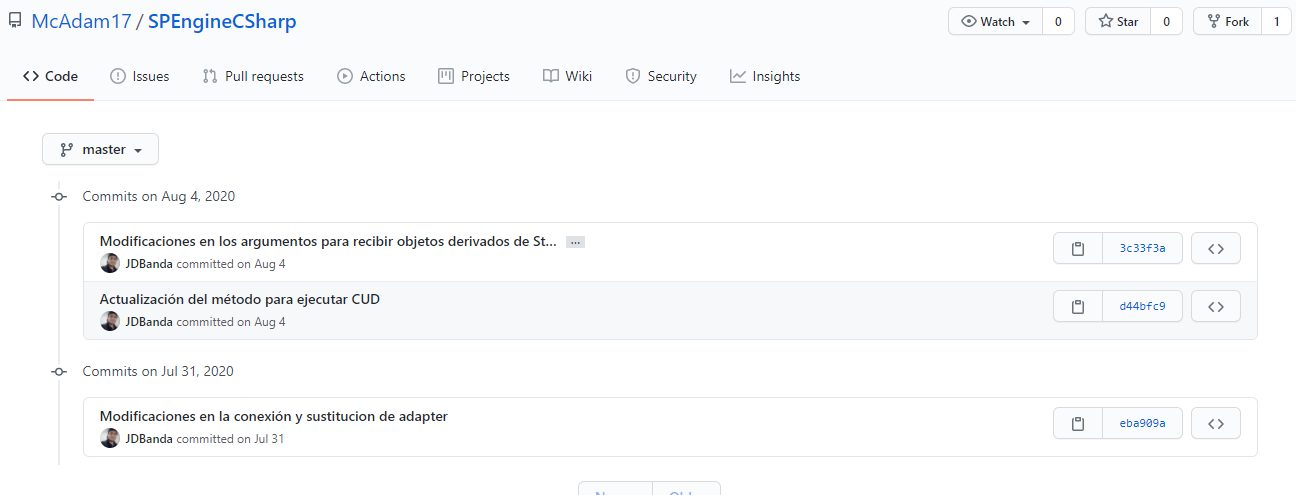


Ilustración 16. Aportaciones al DLL SPEngineCSharp (gráfico propio).

## VIII.7 Objetivo específico 7

Implementar librerías específicas de C# para manipular componentes y colecciones de datos de XML. Se utilizaron las librerías que permiten llevar a cabo LINQ to XML, programación asíncrona, componentes de la GUI, entre otras librerías (ver ilustración 17).

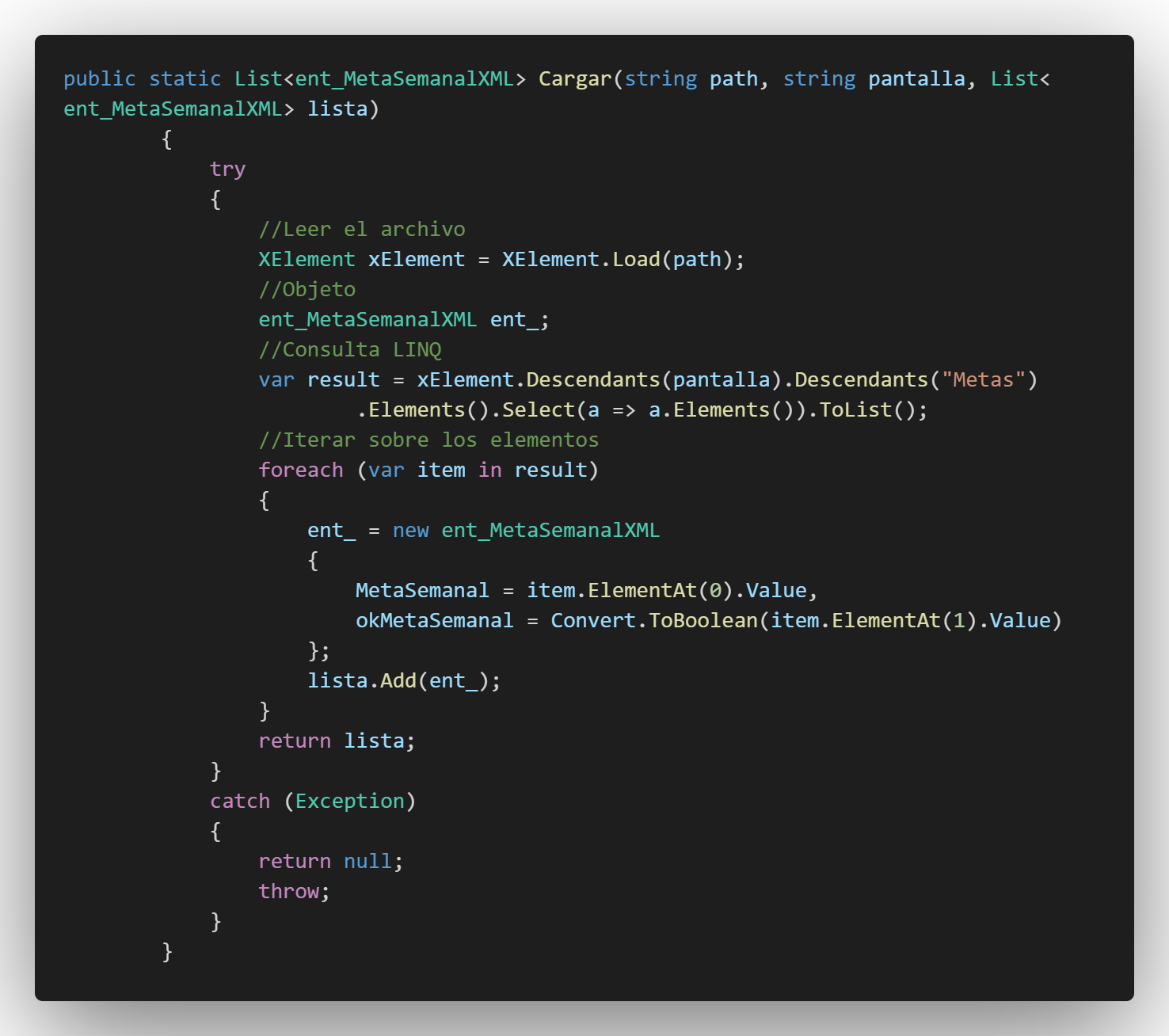


Ilustración 17. Ejemplo de código con LINQ to XML (derechos reservados a ACC MEX S.A. de C.V.).

## VIII.8 Objetivo específico 8

Manejar un control de versiones con Git para manipular los cambios y el trabajo remoto. Git y Github se convirtieron en herramientas indispensables para trabajar debido a la facilidad para manipular los cambios, mostrarlos a otras personas relacionadas con el entorno de trabajo, realizar revisiones de código y detectar errores (ver ilustración 18).

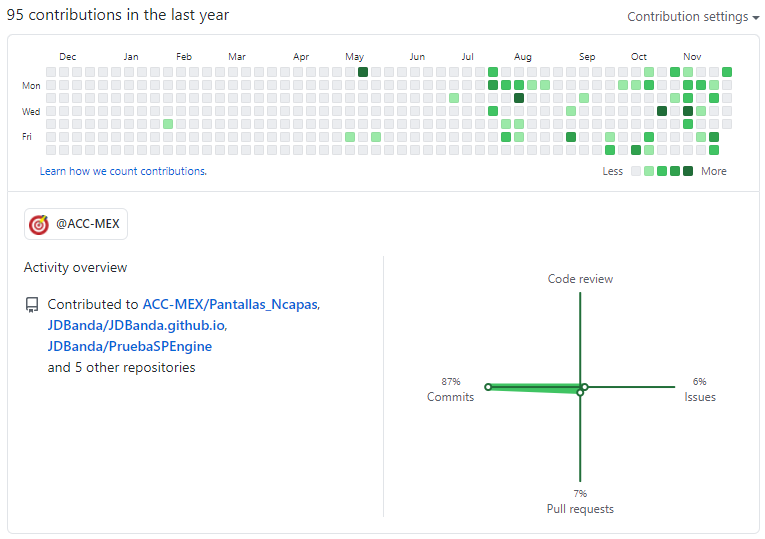


Ilustración 18. Contribuciones a los repositorios del proyecto (gráfico propio).

## VIII.9 Objetivo específico 9

Implementar operaciones CRUD a los módulos correspondientes tanto para componentes como para XML. Las operaciones denominadas CRUD ya descritas en el documento, se realizan a nivel de frontend y de backend, al cumplir con este objetivo se cuenta con la mayor parte de la funcionalidad del proyecto (ver ilustración 19).

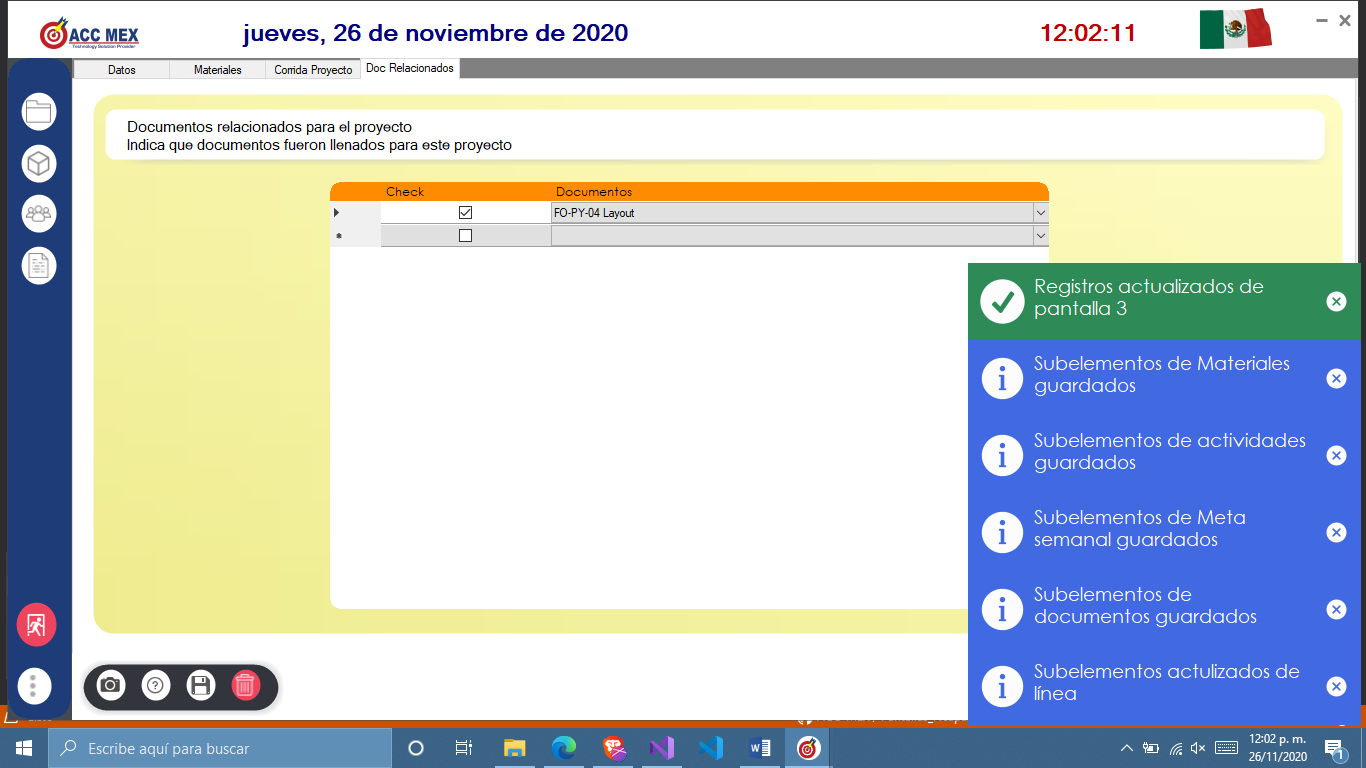


Ilustración 19. Operaciones CRUD (derechos reservados a ACC MEX S.A. de C.V.).

## VIII.10 Objetivo específico 10

Documentar constantemente los cambios que se realicen para conservar un código más limpio. La documentación de código además de realizarse en el manual de usuario (consultar anexos VIII.14.1), también se realiza en dentro del propio código explicando cual es la funcionalidad clave de los métodos y a su vez en GitHub se pueden notar como se llevaron a cabo los cambios y de que se trataron.

## VIII.11 Conclusiones

Se puede determinar que el proyecto termino adecuadamente en una fase de pruebas, lo cual dio la pauta a las personas encargadas del área de proyectos de la empresa para implementar el sistema en producción, se desconoce con exactitud qué impacto positivo pueda tener frente a los inversionistas, debido a temas de contingencia sanitaria que acontecen en el periodo de tiempo que se presenta el proyecto, sin embargo se puede mencionar que se ha cumplido correctamente con el desarrollo de una aplicación para la línea de producción en ACC MEX S.A De C.V. y que cuenta con características que la vuelven competente y sencilla de mantener.

## VIII.12 Recomendaciones

Se sugiere tener un buen dominio sobre el control de versiones, pues este podría ser el origen de un problema o denominado “code smell”, además conocer el entorno de desarrollo en Visual Studio puede no ser suficiente en caso de requerir una segunda actualización, pues se requerirá de un personal capacitado en algún estándar para programar como podría ser el caso de los principios SOLID. Por último, se puede llegar a considerar cambiar el uso de ADO.NET por Entity Framework una manera de conectar que implica mayor seguridad y a su vez mayor responsabilidad técnica.

## VII.13 Bibliografía

Anderson, D. J., & Carmichel, A. (2016). *Essential Kanban Condensed.* Seattle: LEAN Kanban University Press.

Arnold Cathalifaud, M., & Osorio, F. (1998). Introducción a los Conceptos Básicos de la Teoría General de Sistemas. *Cinta de Moebio*(3). Obtenido de https://www.redalyc.org/pdf/101/10100306.pdf

Camps Paré, R., Casillas Santillán, L., Costal Costa, D., Gibert Ginesta, M., Martín Escofet, C., & Pérez Mora, O. (2005). *Bases de datos.* Barcelona: UOC.

Dooley, J. F. (2017). *Software development, design and coding. with patterns, debugging, unit testing and refactoring.* Galesburg, Illinois: Apress.

Dykes, L., & Tittel, E. (2005). *XML for Dummies.* Hoboken: Wiley Publishing.

Escudero Cisneros, J. (2017). *Diseño de sistema de control y gestión de proveedores y agentes para inmobiliaria.* Obtenido de Universidad Autónoma del Estado de Hidalgo: https://www.uaeh.edu.mx/docencia/Tesis/huejutla/licenciatura/

Git. (2014). *Documentation: 1.1 Inicio - Sobre el control de versiones*. Obtenido de Git: https://git-scm.com/book/es/v2/Inicio---Sobre-el-Control-de-Versiones-Acerca-del-Control-de-Versiones

Gómez Fuentes, M. d. (2013). *Notas del curso BASES DE DATOS.* CDMX: Casa abierta al tiempo UAM Cuajimalpa.

Gonzalez Godino, F. (2013). *Diseño de una arquitectura escalable y de alta disponibilidad para un sistema middleware.* Obtenido de Universidad Politécnica de Madrid: http://oa.upm.es/32647/1/PFC\_FERNANDO\_GODINO\_GONZALEZ.pdf

Gorman, K., Hirt, A., Noderer, D., Rowland-Jones, J., Sirpal, A., Ryan, D., & Woody, B. (2019). *Introducing Microsoft SQL Server 2019. Reliability, scalability, and security both on premises and in the cloud.* Birmingham: Packt Publishing.

Johansen Bertoglio, O. (1993). *Introducción general a la teoría general de sistemas.* Santiago: LIMUSA GRUPO NORIEGA EDITORES.

Landa Cosio, N. (25 de Febrero de 2018). LINQ C# en español, básico. *Este es un curso básico de LINQ con C#, nos enfocamos principalmente en LINQ para objetos y XML.* Obtenido de https://www.youtube.com/playlist?list=PLM-p96nOrGcYGyfYpStN3jnaTuIosO3V5

Landa Cosio, N. A. (2018). Tutorial XML en español. *Curso básico de XML para entenderlo de forma rápida y sencilla*. Obtenido de https://www.youtube.com/playlist?list=PLM-p96nOrGcYb96AMy3VdUN8fo-sVAx7K

López Menéndez de Jiménez, R. E. (Diciembre de 2015). Metodologías Ágiles de Desarrollo de Software Aplicadas a la Gestión de Proyectos Empresariales. *ITCA-FEPADE*, 6-11. Obtenido de https://core.ac.uk/download/pdf/80296686.pdf

Luna López, C. A. (2012). *Estructura de datos.* Estado de México: Red tercer milenio.

Merizalde Andrade, C. E., & Toledo Vallejo, A. F. (2013). *Repositorio Institutcional de la Universidad Politécnica Salesiana.* Obtenido de Desarrollo e implementación de un sistema de escritorio para el control de producción: https://dspace.ups.edu.ec/bitstream/123456789/4229/1/UPS-GT000382.pdf

Microsoft. (2020). *.NET*. Obtenido de Microsoft Documentation: https://docs.microsoft.com/es-es

Mompó Camarasa, Á. (2017). *Repositorio Institucional Universitat Politècnica de València.* Obtenido de Estudio de sistema de control de versiones y uso de los mismos en proyectos de integración continua: https://riunet.upv.es/bitstream/handle/10251/87104/MOMP%C3%93%20-%20Estudio%20de%20sistema%20de%20control%20de%20versiones%20y%20uso%20de%20los%20mismos%20en%20proyectos%20de%20integraci%C3%B3....pdf?sequence=1

Morero, F. (2000). *Introducción a la OOP.* Obtenido de Laboratorio de computación de la universidad Magallanes: https://kataix.umag.cl/~ruribe/Utilidades/Introduccion%20a%20la%20Programacion%20Orientada%20a%20Objetos.pdf

Mozilla and contribuitors. (23 de Marzo de 2019). *CRUD*. Obtenido de MDN web docs: https://developer.mozilla.org/es/docs/Glossary/CRUD

Pérez Filoche, Á. (Enero de 2020). *Archivo Digital Universidad Politecnica de Madrid.* Obtenido de Evoluciones y refactorizaciones de una aplicación Android para alfabetización digital: http://oa.upm.es/58085/1/TFG\_ALVARO\_PEREZ\_FILOCHE.pdf

Pítsica Marques, M. (2001). *SISTEMA DE INFORMACIÓN PARA LA GESTIÓN APLICADO EN LAS ENTIDADES FINANCIERAS: ESTUDIO EMPIRICO SANTA CATARINA.* Obtenido de Universidad Complutense Madrid: http://webs.ucm.es/BUCM/tesis//cee/ucm-t25237.pdf

Pressman, R. (2010). *Ingeniería del software. Un enfoque práctico.* Ciudad de México: McGrawll Hill.

Project Management Institute. (2017). *AGILE PRACTICE GUIDE.* Pennsylvania: PMI Book Service.

Red Hat Inc. (2020). *El concepto de IDE*. Obtenido de MiddleWare: https://www.redhat.com/es/topics/middleware/what-is-ide

Refactoring Guru. (2014-2020). *Patrones de diseño*. Obtenido de Refactoring Guru: https://refactoring.guru/es/design-patterns/what-is-pattern

Richards, M. (2015). *Software architecture patterns.* Sebastopol, CA: O’Reilly Media, Inc.

SAP. (2020). *SAP History*. Obtenido de SAP: A 48-year history of success: https://www.sap.com/corporate/en/company/history.html

Sebastián Gómez, L. (2000). *Diseño de Interfaces de Usuario Principios, Prototipos y Heurísticas para Evaluación.* Obtenido de ResearchGate: https://www.researchgate.net/publication/228877430\_Diseno\_de\_Interfaces\_de\_Usuario\_Principios\_Prototipos\_y\_Heuristicas\_para\_Evaluacion

Stack Overflow Comunity. (07 de Marzo de 2017). *C# Notes for professionals*. Obtenido de GoalKicker.com: https://goalkicker.com/CSharpBook/

Suárez Rey, C. (2010). *Sistemas integrados de gestión.* Obtenido de Grupo Jean Monet de Competencia y desarrollo regional: http://www.gcd.udc.es/subido/catedra/presentaciones/economia\_competencia\_ii/nota\_tecnica\_sistemas\_de\_gestion\_erp\_carlos\_suarez\_rey\_17-03-2010.pdf

Suryanarayana, G., Samarthyam, G., & Sharma, T. (2015). *Refactoring for software design smells. Managing technical debt.* Waltham, MA: Elsevier.

The Institute of Electrical and Electronics Engineers. (28 de Septiembre de 1990). IEEE Standard Glossary of Software Engineering Terminology. *IEEE Std 610.12-1990*. New York, New York, United States of America: IEEE Standards.

Yépez Llerena, E. D., & Armijos Guillen, K. F. (2020). *Repositorio de la UNACH.* Obtenido de APLICACIÓN DE LA METODOLOGÍA KANBAN EN EL DESARROLLO DEL SOFTWARE PARA GENERACIÓN, VALIDACIÓN Y ACTUALIZACIÓN DE REACTIVOS, INTEGRADO AL SISTEMA INFORMÁTICO DE CONTROL ACADÉMICO UNACH.: http://dspace.unach.edu.ec/bitstream/51000/6457/1/APLICACI%C3%93N%20DE%20LA%20METODOLOG%C3%8DA%20KANBAN%20EN%20EL%20DESARROLLO%20DEL%20SOFTWARE%20PARA%20GENERACI%C3%93N.pdf

De miguel Adoración., Piattini Mario., Marcos Esperanza. (2000). Diseño de base de datos relacionales. Alfaomega

Eguíluz Pérez Javier. (Junio de 2008). Introducción a JavaScript. Recuperado de https://goo.gl/y0cWEG

Gómez, I., Pegenaute, X., Reig, G., & Torres, J. (2009). Cloud Computing una visión desde un aula de la UPC. Sevilla, España: Lulú.

Heurtel, O. (2014). PHP y MySQL: domine el desarrollo de un sitio web dinámico e interactivo. Ediciones ENI.

Kendall, J. E. (2005). Análisis y diseño de sistemas. Pearson Educación. Lafosse, J. (s.f.). Struts 2: El framework de desarrolo de aplicaciones Java EE 2014. Cornellà de Llobregat (Barcelona): ENI.

Laudon, K., & Laudon, J. (2004). Sistemas de información gerencial (Octava ed.). México: Pearson

Lineamientos para la Operación y Acreditación de la Residencia Profesional versión 1.0 Planes de Estudio 2009-2010, Fecha de creación 27 de octubre de 2011

Lineamiento para la Operación y Acreditación del Servicio Social versión 1.0 Planes de Estudio 2009- 2010, Fecha de creación 2 de septiembre de 2011.

Luján Mora Sergio. Programación de aplicaciones web: historia, principios básicos y clientes web. Editorial Club Universitario, Alicante, 2002. ISBN: 978-84-8454-206-3. (PI'2002) Libro / Book

Nevado, M. (2010). Introducción a Las Bases de Datos Relacionales. España: Visión Libros

Pressman, R., & Murrieta, J. (2006). Ingeniería del software un enfoque práctico (6ª Edición ed.). McGraw-Hill.

Sommerville, Ian M. I. (2005). Ingeniería del software. Pearson Educación.

Zaefferer Jörn. (s.f). Validación de formularios con jQuery. Recuperado de https://Jqueryvalidation.org/

## VIII.14 Anexos

### VIII.14.1 Manual de usuario

En este anexo se desarrolla el manual técnico para comprender el funcionamiento adecuado del software.

Una vez que el proyecto ha sido iniciado desde la solución de Visual Studio o como un archivo ejecutable, se mostrará la siguiente pantalla (ver ilustración 20), el tamaño de la ventana varía de acuerdo a la resolución del equipo.

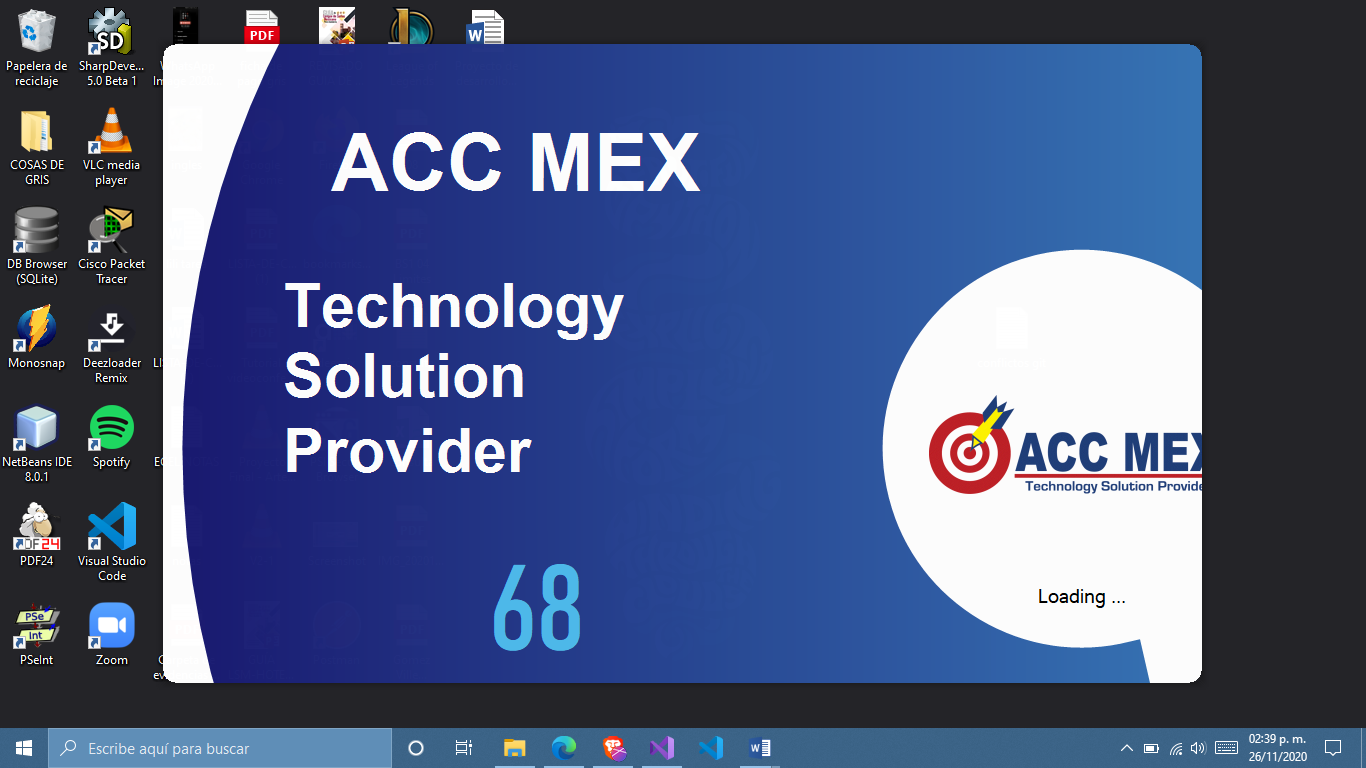


Ilustración 20. Inicio de la aplicación.

En algunas ocasiones se muestran cuadros de dialogo para visualizar la carga entre interfaces, si el usuario da clic en aceptar, simplemente se eliminarán lo mismo sucede con las notificaciones en la parte inferior derecha de la pantalla (ver ilustración 21).

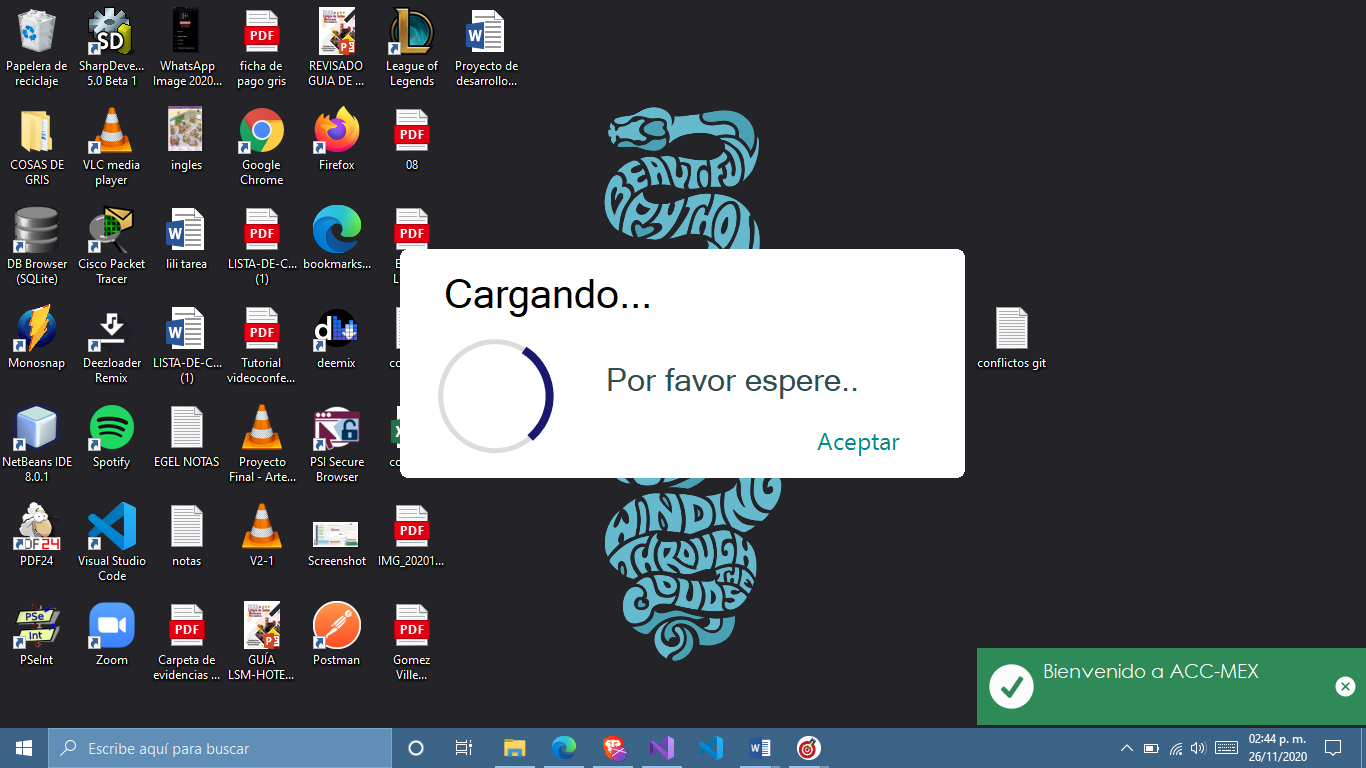


Ilustración 21. Alertas y cargas.

El menú principal cuenta con tres botones grandes de color azul en donde se llevan a cabo las principales operaciones del programa, además de existir un cuarto botón flotante para activar algunas funciones (ver ilustración 22).

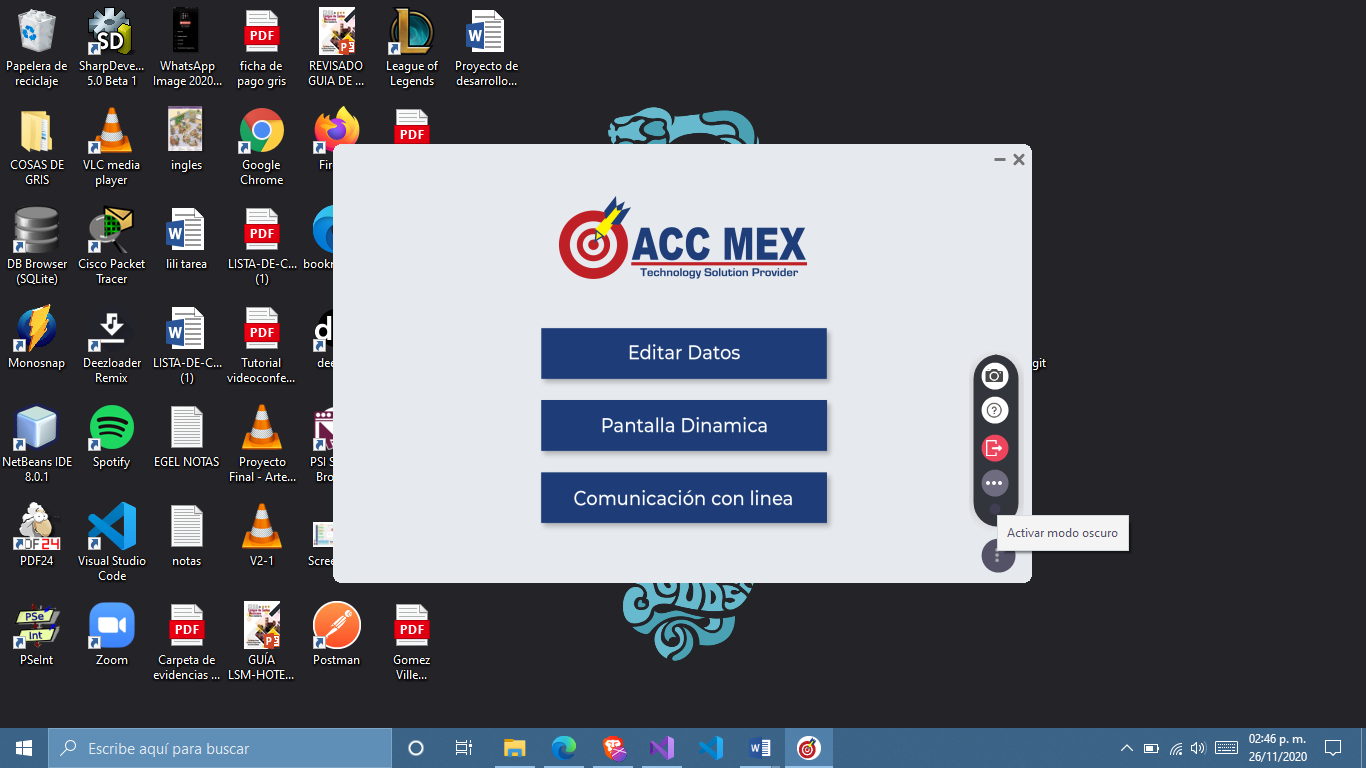


Ilustración 22. Menú principal.

El botón flotante puede ser activado o desactivado pulsando en los tres puntos verticales (E), para cambiar a un tema de color obscuro se utiliza el botón con los tres puntos dispuesto de manera horizontal (D), si se desea salir de la aplicación sin presionar la pequeña letra x en la parte superior derecha es posible realizarlo con el botón rojo que tiene el icono de puerta (C), cuando se requiera solicitar ayuda se puede presionar el botón con signo de interrogación (B) y el último botón colocado en la parte superior realiza una captura de pantalla (A) estos botones se pueden utilizar en la pantalla principal únicamente, en otras pantallas existen los mismos elementos solo que con distintas funcionalidades (ver ilustración 23).

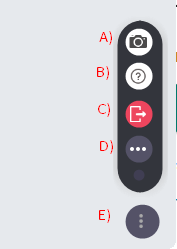


Ilustración 23. Botón flotante del menú principal.

El apartado para editar datos cuenta con una serie de pestañas (ver ilustración 24) que realizan funcionalidades específicas para cada caso, existen dos formas de cambiar entre pantallas, la primera de ellas actualiza el contenido de la pantalla (localizado en la parte superior) y el segundo no actualiza el contenido, lo que quiere decir que si se desean conservar datos antes de guardarlos esta podría ser la mejor alternativa (ver ilustración 25).

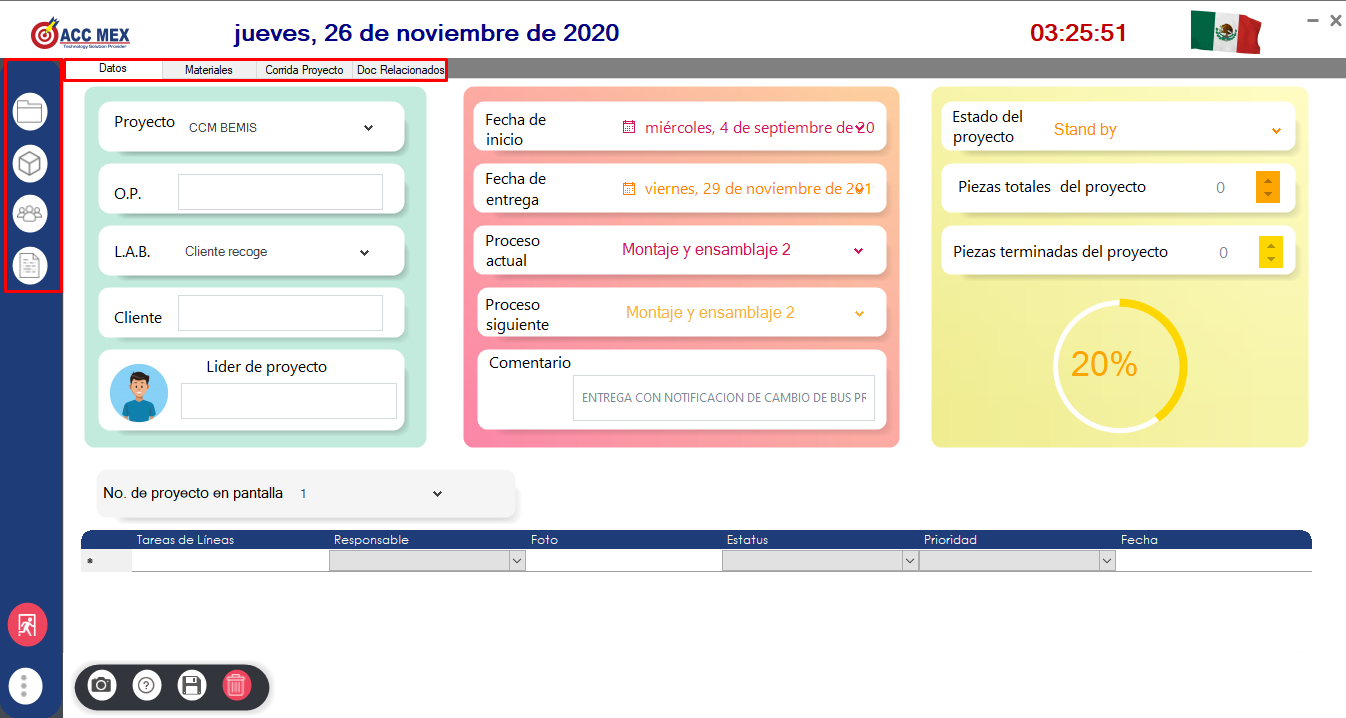


Ilustración 24. Ventana para editar datos.

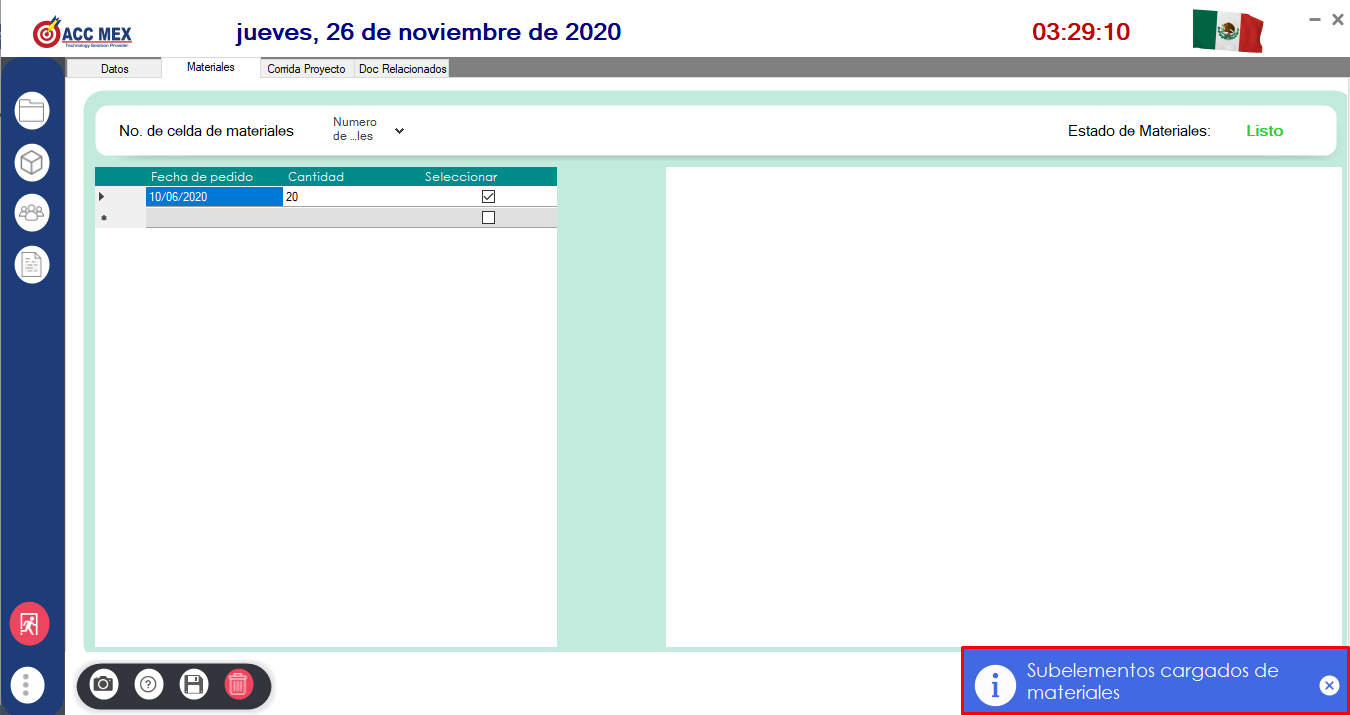


Ilustración 25. Datos actualizados con ayuda de las pestañas superiores.

El primer apartado contiene varios campos por llenar (ver ilustración 26), algunos de ellos son datos precargados, lo que quiere decir que ya han sido registrados en base de datos y solo se están mostrando, si se realiza una edición solo se verá reflejada en el archivo local de almacenamiento, los datos mostrados se listan a continuación:

1. Combo box en el que se selecciona el proyecto en el cual se quiera trabajar, algunos datos se precargan con esta selección, su origen es la base la datos lo que significa que no pueden ser borrados.
2. (Orden de producción) hace referencia al Id asignado para producción.
3. (Libre a bordo) existen tres opciones predeterminadas, Cliente recoge, ACC MEX envía y ACC MEX paquetería.
4. (por motivos de confidencialidad no se coloca el nombre), este dato esta precargado.
5. Dato precargado para el nombre del líder
6. Dato precargado para la fecha de inicio
7. Dato precargado para la fecha entrega
8. Combo box para seleccionar el proceso actual
9. Combo box para seleccionar el proceso siguiente
10. Dato precargado en el que se muestra un comentario
11. Combo box para seleccionar el estado del proyecto
12. Selector de número para indicar el total de piezas
13. Selector de número para indicar el número de piezas terminadas

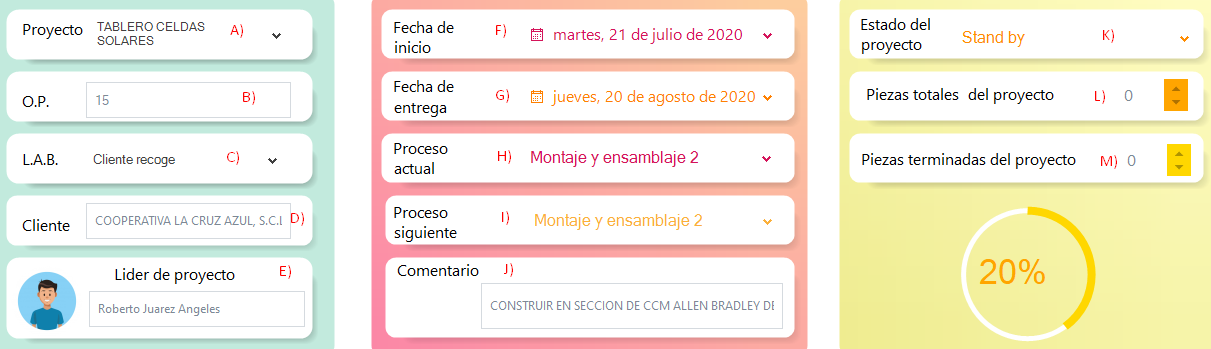


Ilustración 26. Campos estáticos de la pestaña Datos.

En el apartado posterior del programa (ver ilustración 27) se puede observar una serie de componentes que controlan otro tipo de acciones y datos explicados a continuación:

1. Combo box en donde se selecciona el número de pantalla a presentar, al cambiar el número los datos se actualizará automáticamente al de la pantalla seleccionada, en caso de estar vacío mostrará un mensaje y en caso de estar lleno mostrará otro mensaje, es importante mencionar que si se colocan datos y se cambia el índice estos no se guardarán y cambiará a otra pantalla.
2. Las tareas de línea son tareas asignadas por el personal, este campo es de texto.
3. Combo box para seleccionar el responsable contiene a todo el personal de ACC MEX registrado en el ERP.
4. El apartado de foto no se guarda, es una funcionalidad que no se implementó.
5. Combo box estatus del proyecto cambia el valor del selector por detenido, en proceso, esperando revisión y listo.
6. Combo box que almacena datos de un archivo de configuración y tiene los valores alta, media y baja.
7. Campo de texto en el que se puede almacenar la fecha en la que se asignó la tarea de línea.

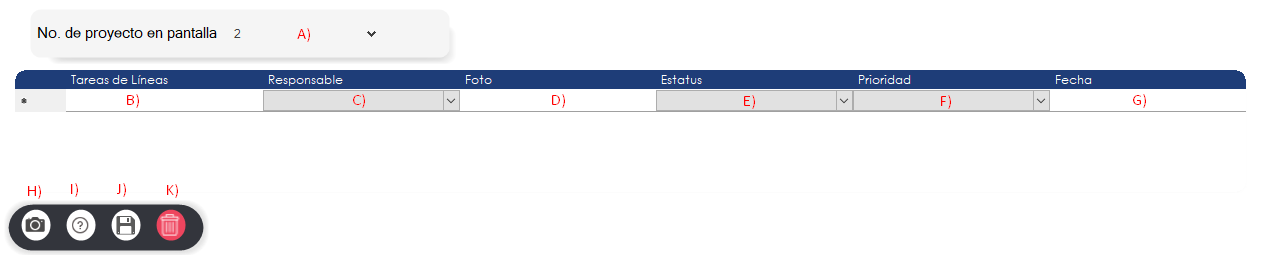


Ilustración 27. Campos dinámicos y controles para operaciones de alta, baja y modificaciones.

La pestaña siguiente tiene por nombre Materiales, en esta se muestran los materiales disponibles del proyecto que se seleccionó en la pestaña datos (ver ilustración 28), en caso de no tener materiales relacionados mostrará una ventana en blanco del lado derecho (ver ilustración 29)

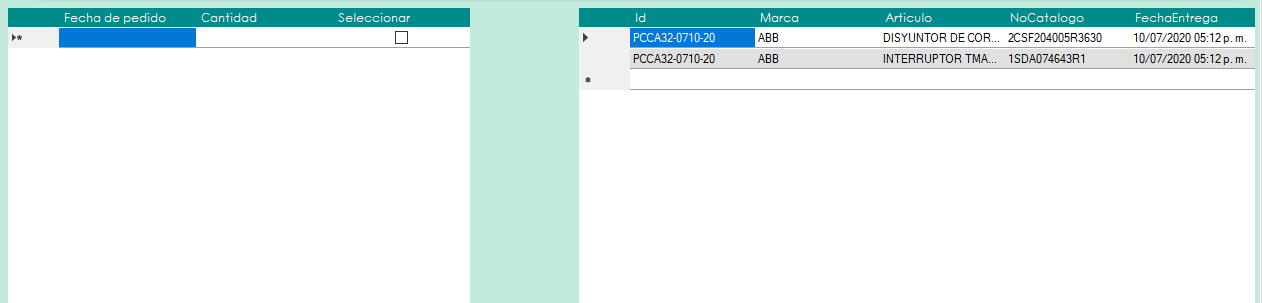


Ilustración 28. Pestaña Materiales con datos precargados.

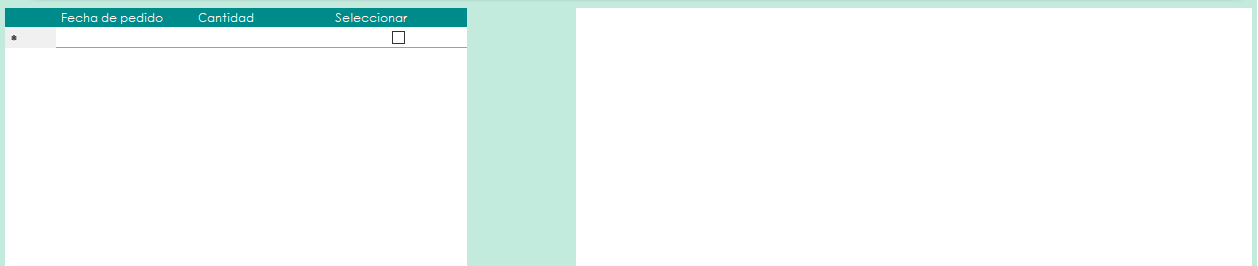


Ilustración 29. Pestaña Materiales sin datos.

La tabla en donde se pueden editar datos será la del lado izquierdo (ver ilustración 30), se guardarán únicamente las filas que sean iguales o menores al número de filas precargadas en la tabla derecha, al utilizar el programa se puede observar que no hay un límite para agregar las filas, sin embargo las que superen el número no se guardarán, los campos guardados son:

1. Campo de texto denominado fecha de pedido puede almacenar una fecha o cadena para indicar cuando se entregará.
2. Campo de texto denominado cantidad hace referencia al número de piezas que se pedirán.
3. Campo seleccionable para indicar si se realizará o no la selección de materiales.

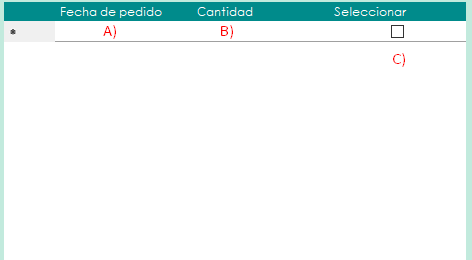


Ilustración 30. Edición de datos en Materiales.

Dentro de la pestaña corrida del proyecto existen dos tablas, en la primera (ver ilustración 31) se especifican los campos que se almacenan relacionados a las actividades de la corrida del proyecto:

1. Combo box precargado de un archivo de configuración en el cual se lista una serie de actividades que podría desempeñar un departamento.
2. Combo box precargado gracias a la base de datos que muestra los departamentos existentes.
3. El selector indica que está listo.
4. Este otro selector indica que no está listo.

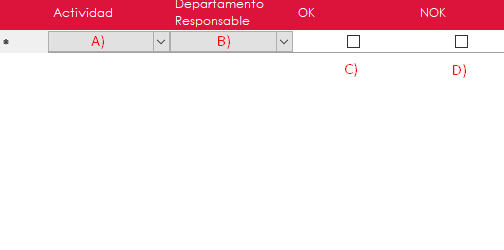


Ilustración 31. Tabla para registrar las actividades.

La segunda tabla hace referencia al producto que estará marcado como meta semanal para entregar en la línea de producción, únicamente se puede seleccionar el nombre del elemento y si está listo o no (ver ilustración 32).



Ilustración 32. Tabla de meta semanal

Por último en la pestaña de documentos relacionados (ver ilustración 33) se encuentra una única tabla que contiene dos tipos de datos a almacenar, el primero de ellos es el control de selección que define si está o no seleccionado el documento, el otro tipo de dato a guardar es el relacionado con los documentos los cuales tienen como origen un archivo de configuración.



Ilustración 33. Tabla de documentos relacionados.

Si el usuario ya ha cargado todos los elementos pertinentes dentro de la ventana para editar datos, puede hacer uso de los botones inferiores (ver ilustración 34) el primer botón tiene la función de salir y el segundo botón puede desplegar un menú flotante (ver ilustración 35).



Ilustración 34. Botones inferiores del a ventana editar datos.



Ilustración 35. Menú flotante.

Dicho menú flotante tiene la función de capturar una imagen, mostrar ayuda, guardar o editar datos y eliminarlos en ese orden.

El apartado siguiente hace referencia a la ventana dinámica (ver ilustración 36), para que esta ventana funcione correctamente es necesario dejar que cargue los componentes por algunos segundos, pues debido a que se cargan los recursos de forma asíncrona, la ventana está repartida en cuatro secciones.



Ilustración 36. Pantalla dinámica.

La primera sección es el menú lateral izquierdo (ver ilustración 37) en la cual se puede capturar la imagen y guardarse, enviarse por correo o cancelar dicho captura.



Ilustración 37. Menú lateral izquierdo.

La siguiente parte es un espacio estático en donde únicamente se podrá cambiar la pantalla cambiando el valor en el combo box indicado con color rojo (ver ilustración 38).



Ilustración 38. Sección estática.

La parte siguiente esta segmentada entre la o las tablas que van cambiando cada 5 segundos (ver ilustración 39), se puede notar fácilmente que los datos aquí mostrados son aquellos que se guardaron en la ventana de editar datos.

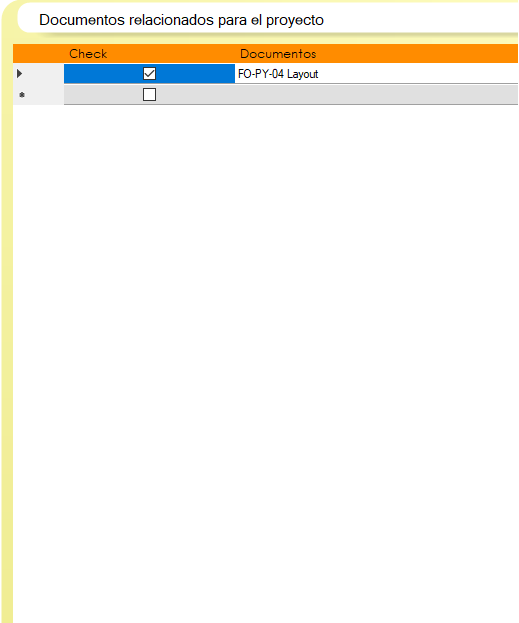


Ilustración 39. Parte dinámica 1.

Y la otra parte es la tabla que se obtiene de la pestaña datos del menú editar datos, no puede ser editado (ver ilustración 40).

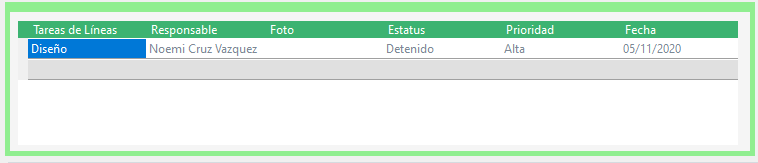


Ilustración 40. Parte dinámica 2.

Finalmente se pueden observar controles de tiempo y algunos cuadros de texto, esta sección se realizó para conocer cuál es tiempo total de trabajo en la línea de producción (ver ilustración 41), a continuación se explica que función tiene cada botón y componente:

1. Inicio del tiempo de trabajo.
2. Pausa del tiempo de trabajo, entra en acción el tiempo de descanso.
3. Pausa del tiempo de descanso, crea un paro en producción.
4. Termina la producción y muestra el total de tiempo invertido en el componente E) y F)
5. Muestra únicamente el tiempo que se ha registrado para el trabajo.
6. Muestra únicamente el tiempo que se ha registrado para el descanso.

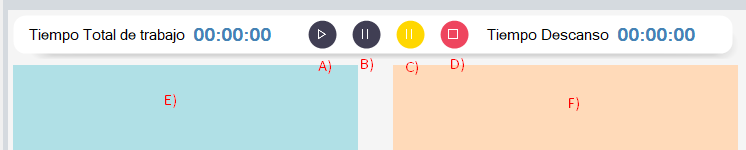


Ilustración 41. Controles de tiempo.