Sistema informático para gestionar basado en SCRUM, el desarrollo de proyectos de software

Colaboración
José Raúl López Morales; José Francisco Gazga Portillo; Juan Miguel Hernández Bravo; Alma Delia de Jesús Islao,
Tecnológico Nacional de México / Instituto Tecnológico de Acapulco

RESUMEN: En este artículo, se presenta el trabajo interdisciplinario de la Maestría en Sistemas Computacionales con apoyo del CONACyT, impartida en el Instituto Tecnológico de Acapulco. El artículo tiene por meta, presentar el desarrollo de un sistema informático que permite gestionar y monitorizar proyectos de desarrollo de software cuya construcción se basa en la metodología Scrum, la implementación de este sistema permitirá fortalecer el desarrollo de software en el departamento de desarrollo de la empresa: Proyectos, Instalaciones y Construcciones Civiles y Eléctrica (PICCE), al proveer una forma de monitorizar y gestionar los proyectos con base a la metodología Scrum, proporcionando así, una flexibilidad en los cambios de requerimientos, además de entregas funcionales en cortos periodos.

Se hace mención a que este artículo forma parte de una serie de trabajos siendo el cuarto y último de estos, el cual da seguimiento a un artículo previo titulado: Patrones de acceso de una aplicación basada en Scrum para la gestión de proyectos de desarrollo de software, escrito por los mismos autores y que fue aceptado en el séptimo congreso internacional de robótica y computación del Instituto Tecnológico de La Paz.

PALABRAS CLAVE: arquitectura MVC, ASP.NET Core, Metodología Scrum, unidad de trabajo, repositorio.

ABSTRACT: In this article, the interdisciplinary work of the Master in Computer Systems with the support of CONACyT, which is taught at the Instituto Tecnológico de Acapulco, is reflected. The article aims to present the development of a computer system that allows managing and monitoring software development projects, WHICH make use of the Scrum methodology, the implementation of this system will strengthen software development in the company's development department: Proyectos, Instalaciones y Construcciones Civiles y Eléctrica (PICCE), by providing a way to monitor and manage projects based on the Scrum methodology, thus providing flexibility in changing requirements, as well as functional deliveries in short periods. It is mentioned that this article is part of a series of works being the fourth and last one of these works, this article follows up on a previous one entitled: Access Patterns of a Scrum-based Application for the Management of Software Development Projects, written by the same authors and which was presented and published at the seventh international conference on robotics and computing at the Instituto Tecnológico de La Paz.

KEYWORDS: architecture MVC, ASP.NET Core, Scrum methodology, unit of work, repository.

INTRODUCCIÓN

A mediados de los 90's comenzó a forjarse la definición de desarrollo ágil como una reacción en contra de las metodologías tradicionales (modelo en cascada, prototipo, incremental, desarrollo rápido de aplicaciones), que son consideradas pesadas y rígidas por su carácter normativo y una fuerte dependencia de planeaciones detalladas previas al desarrollo. El uso de las metodologías de desarrollo ágil o metodologías ágiles, son más recientes en la ingeniería de software, las cuales están acaparando, a la vez, gran interés y controversia (Ruiz Guadalupe, 2014).

Los procesos de desarrollo del software rápido se diseñan para producir rápidamente un software útil. El software no se desarrolla como una sola unidad, sino como una serie de incrementos, y cada uno

de ellos incluye una nueva funcionalidad del sistema (Sommerville, 2016).

Existen varias formas de metodologías ágiles como el método de desarrollo de sistemas dinámicos (en inglés Dynamic Systems Development Method o DSDM), Scrum, la programación extrema (en inglés eXtreme Programming o XP) pero todas ellas comparten características similares. La piedra angular de cada rama, es la idea de la satisfacción del cliente (Blankenship, Bussa, & Millett, 2011).

Como se mencionó anteriormente, existen diversas metodologías y cada una se enfoca en ciertos aspectos del desarrollo de software, por ejemplo, la programación extrema se centra en la propia programación del producto, en cambio, la metodología Scrum se enfoca en la administración del proyecto. En el presente artículo se aborda el desarrollo de un sistema informático para la gestión y monitorización de proyectos de desarrollo de software basados en Scrum, que una plataforma web para gestionar proyectos de desarrollo de software de una forma colaborativa con base a los roles establecidos en la metodología Scrum, además permitirá la visualización del progreso de cada una de las tareas involucradas en cada sprint de un proyecto. Actualmente en el mercado existe diferentes plataformas para la gestión de proyectos las cuales son de propósito general. Los requerimientos solicitados por la empresa PICCE no son cumplidos en su totalidad por estas plataformas existentes, además de que tampoco desean adquirir un licenciamiento, por tal motivo surge la necesidad de desarrollar una plataforma para la gestión y monitorización de proyectos de desarrollo de software basado en Scrum para la empresa mencionada con anterioridad.

Objetivo general

El objetivo principal consiste en desarrollar una plataforma web para la gestión y monitorización de proyectos de desarrollo de software basados en la metodología Scrum para la empresa PICCE, utilizando el lenguaje de programación C# con el framewrok ASP. NET Core.

Planteamiento del problema

El departamento de desarrollo de software de la empresa PICCE en los últimos años ha estado usando los modelos desarrollo de software: prototipo e incremental, pero los han dejado de usar debido a ser robustos, es decir, implica que tienen que realizar una serie de fases e ir documentando cada una de ellas, pero no es permitido hacer cambios en los requerimientos que estaban establecidos al inicio de cada proyecto. Los nuevos requerimientos que se definían se tienen que hacer al final del producto entregable y esto causa que se prolongue la fecha de entrega del producto de manera considerable causando en ocasiones gastos para la empresa.

Actualmente, el departamento de desarrollo de la empresa PICCE ha estado implementado la metodología ágil Scrum en la gestión del desarrollo de proyectos de software debido a que, se trata de una metodología flexible en actualizaciones de requerimientos y las entregas del producto son más rápidas. Sin embargo, aun implementando dicha metodología, se llegan a presentar problemas de entregas de los sprints, originando la necesidad de implementar un sistema bajo los requerimientos propios de la empresa PICCE que le permita realizar la monitorización día a día del progreso de cada tarea que está involucrada en el desarrollo de sus productos de software, así como las horas diarias que invierte un desarrollador a dichas tareas. Así mismo, percibir, identificar los inconvenientes que se están presentando en el desarrollo de cada producto y de esta forma prevenir retrasos en las entregas de los sprints.

MATERIAL Y MÉTODOS

En esta sección se describe la arquitectura del sistema propuesto, las herramientas para su desarrollo, así como la metodología de desarrollo del proyecto.

Metodología

El desarrollo de esta investigación se realizó bajo el esquema de investigación aplicada, la cual consta de tres etapas:

- Ideación y conceptualización: Donde se eligió el tema de investigación, se planteó el problema a resolver, se investigó los antecedentes de las metodologías tradicionales y ágiles, así una metodología para la compresión de los procesos en la metodología Scrum, a través del modelo de negocios que se describe en esta sección.
- Ejecución: En la sección de desarrollo del sistema informático se describe el desarrollo de la plataforma, la cual es soportada por la identificación de los procesos en que estará involucrada la plataforma dentro de la metodología Scrum y en conjunto con las reglas de negocio proporcionadas por la empresa PICCE, utilizando las herramientas de desarrollo que se describen más adelante.
- Resultados de investigación (RI) y Transferencia: Los resultados de esta investigación se muestra en la sección resultados de este artículo, derivados de la etapa de ejecución.

Para el desarrollo de la plataforma, se toma en cuenta una visión global del sistema a desarrollar, en el Anexo A se muestra el modelo de negocios del sistema. Con este modelo se presentan los procesos de la propia metodología Scrum, esto permite tener una mejor compresión de los procesos que se llevan a cabo durante la implementación de la metodología Scrum, y a su vez, saber en qué procesos estará involucrada la plataforma a desarrollar.

En el diagrama que se ilustra en el Anexo A, se observa cuatro carriles que representan los roles de la metodología Scrum donde cada uno realiza ciertos procesos en la metodología, así como los artefactos o cambios en la base de datos que se generan en los procesos. A continuación, se describe con un alto nivel de abstracción cada uno de los procesos del modelado de procesos de negocio:

- Proceso: Colección de las historias de usuario. El cliente (StakeHolder) menciona las necesidades o requisitos (historias de usuario) al product owner y éste realiza la anotación de las mismas.
- Proceso: Identificación de las historias de usuario. El product owner identifica e interpreta las historias de usuario a través de un lenguaje de fácil entendimiento para el equipo de desarrollo, además éstas definirán la liberación del producto final.
- Proceso: Asignación de prioridad a las historias de usuario. El product owner prioriza las historias de usuario según al criterio del cliente y se obtiene un product backlog priorizado.
- Proceso: Estimación de tiempo para cada historia de usuario. El equipo de desarrollo junto con el product owner, estiman los tiempos para el desarrollo de cada historia de usuario.
- Proceso: Presentación de historias de usuarios con mayor prioridad. Se realiza una reunión antes de iniciar el Sprint, en la cual, el product owner presenta las historias de usuario con mayor prioridad hasta el momento al equipo de desarrollo, dando origen a un Sprint backlog.
- Proceso: Desglosamiento de tareas. El equipo de desarrollo divide en pequeñas tareas las historias de usuarios.
- Proceso: Estimación de tiempo para cada tarea. El equipo de desarrollo estima un tiempo para cada tarea que surgió del desglosamiento de tareas.
- Proceso: Asignación de tareas a realizar. Entre los integrantes del equipo de desarrollo se lleva a cabo la asignación de las tareas a realizar para completar cada una de las historias de usuario.
- Proceso: Realización de las tareas asignadas de cada historia de usuario. El equipo de desarrollo dedica tiempo diariamente a cada tarea para completar las historias de usuario según lo estimado.
- Proceso: Seguimiento de cada sprint. Cada integrante del equipo de desarrollo registra las horas invertidas en cada tarea asignada, esto sirve para saber el avance real de un proyecto.
- Proceso: Revisión del sprint. El product owner comprueba el progreso del proyecto, donde identifica las funcionalidades que se pueden considerar hechas y las que no.
- Proceso: Verificación de la desviación de tiempos de planeación. El scrum master verifica la desviación de tiempos de planeación en cada Sprint apoyándose con el burndown chart y poder mitigar riesgos en caso de existir anomalías en los tiempos estimados, permitiendo que no existan atrasos en las entregas de los Sprints.

Arquitectura del sistema informático

En el presente artículo se propone el desarrollo de un sistema informático para la gestión de proyectos de desarrollo de software basados en la metodología Scrum. El desarrollo del sistema propuesto, está diseñado con base en el patrón arquitectónico: Modelo-Vista-Controlador (Model-View-Controller o MVC) (ver figura 1) propuesto como arquitectura en el primer artículo (López Morales, Gazga Portillo, Hernández Bravo, & de Jesús Islao, 2019).

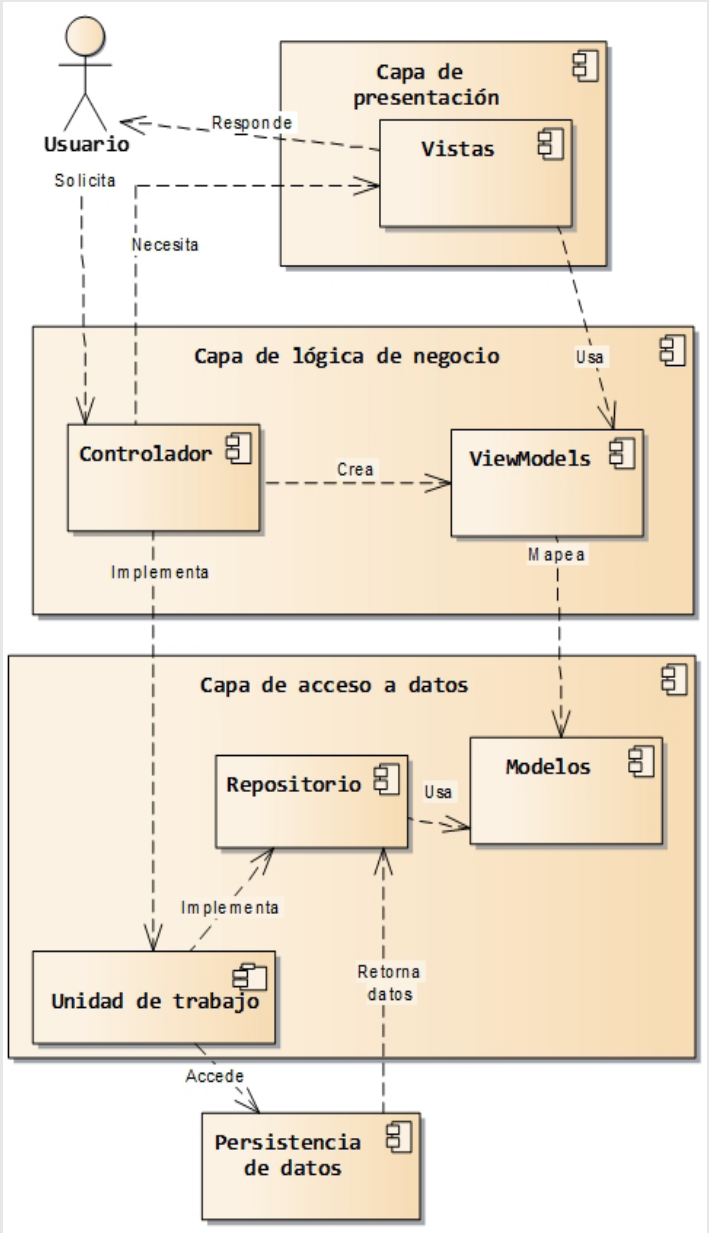

Figura 1. Arquitectura del sistema implementando el patrón MVC.

El diagrama de la figura 1 muestra los componentes que conforman cada una de las capas de la arquitectura del sistema y la manera en que se relacionan. En

la capa de acceso a datos se puede observar los patrones unidad de trabajo y repositorio donde la implementación de estos fue expuesta en el tercer artículo de la serie (López Morales, Gazga Portillo, Hernández Bravo, de Jesús Islao, & Ramírez Silva, 2020).

Se describe brevemente el flujo que sigue una petición realizada por un usuario a través de los componentes de cada capa de la arquitectura del sistema. El usuario realiza una petición al controlador el cual contiene la lógica de negocio, éste crea un objeto de un viewmodel el cual es una copia exacta de un modelo o la combinación de estos para representar un conjunto de datos de diferentes modelos, permitiendo crear una capa de abstracción entre los modelos que almacenan datos temporalmente y las vistas. La construcción de esta capa de abstracción, permite proporcionar una mayor seguridad debido a que los controladores no manipulan directamente los modelos, evitando que se realicen cambios en la persistencia de datos no deseados ya que los modelos son la representación de las tablas de la persistencia de datos. Debido a que el viewmodel es creado por el controlador, éste puede ser mapeado con el modelo o modelos correspondientes para que al momento de realizar una transacción utilizando el patrón unidad de trabajo, el patrón invoque uno de los métodos (agregar, consultar, modificar y/o eliminar) proporcionados por el patrón de repositorio, el cual va a recibir un modelo para realizar la transacción con la persistencia de datos. En caso de realizar una consulta de datos se lleva a cabo el mismo proceso que cuando un usuario efectúa una petición, pero al momento de consultar los datos a la persistencia de datos, éste devuelve un conjunto de datos al patrón de repositorio que es implementado por el patrón de unidad de trabajo que retorna un modelo o un conjunto de propiedades de diferentes modelos correspondiente a los datos devueltos, en seguida, previo a que llegue el modelo o el conjunto de propiedades con datos almacenados temporalmente, son mapeados con un viewmodel para que el controlador muestre los datos almacenados en el viewmodel temporalmente a la vista. En la figura 1 se observa entre la capa de presentación y lógica de negocio la relación entre el controlador, el viewmodel y la vista generando la misma estructura y relaciones de MVC.

Herramientas de desarrollo

Para el desarrollo del sistema propuesto se implementó la metodología Scrum, la cual fue descrita en el primer artículo (López Morales, Gazga Portillo, Hernández Bravo, & de Jesús Islao, 2019). En la tabla 1 se describe el propósito de haber usado cada tecnología en la implementación de los componentes para el desarrollo del sistema propuesto, tanto para la parte de lado del servidor (Back-End) como para la parte de lado del cliente (Front-End).

Tabla 1. Descripción de uso de cada tecnología para el desarrollo del sistema.

Back-End	
Tecnología	**Uso**
ASP.NET Core	Proporciona un marco de trabajo para desarrollar una aplicación web basado en el patrón **MVC**.
C#	El lenguaje de programación **C#** permite la creación de la **lógica de negocio** en los **controladores** y de los **modelos**.
Entity Framework Core	Esta tecnología facilitó la generación y conexión de la base de datos a partir de los modelos creados con **C#**. Esta tecnología es un **Direccionador de Objetos Relacionales** (en inglés **Object-Relational Mapper**, **O/RM**).
LINQ	Con **LINQ** se realizan **consultas SQL** nativamente con **C#**, facilitando la creación y mantenimiento de éstas.
AutoMapper	Esta librería facilita el mapeo entre clases, por ejemplo, entre **viewmodels** y **modelos**.
PostgreSQL	Para almacenar la **base de datos** creada a partir de los **modelos** con **Entity Framework Core,** se hizo uso del sistema gestor de base de datos **PostgreSQ**L.
Front-End	
Tecnología	**Uso**
HTML	HTML proporciona una estructura **web** general para cada **vista** del sistema propuesto.
CSS	Se usa con el fin de implementar, con base a los mecanismos de estilos proporcionados por **CSS**, la estructura web general de las **vistas** construidas en HTML, configurándoles los atributos tales como color, tipografía, márgenes, posicionamiento, etc.
JavaScript	Para la manipulación de información y validaciones en el **Front-End**, se utilizó el lenguaje de programación **JavaScript**.
JQuery	El uso de la librería de **JavaScript: JQuery**, tiene por objeto permitir agregar **interactividad** y **efectos visuales** en las **vistas**.
Bootstrap	Para crear **vistas** con **diseño adaptativo** haciendo uso de **CSS**, **JavaScript** y **JQuery**, se implementó el framework **Bootstrap**.
Chart js	Para la graficación del **burndown chart** se hizo uso de la librería **Chart js**.

Las herramientas para el desarrollo del sistema propuesto son las siguientes:
Visual Studio Community 2019.
PgAdmin 3.
Las características del equipo donde se desarrolla el sistema son las siguientes:
S.O Windows 10 a 64 bits.
16 Gb de memoria RAM.
Procesador Intel Core i7.

Desarrollo del sistema informático

Con el diseño del sistema presentado en el segundo artículo (Gazga Portillo, López Morales, Hernández Bravo, & de Jesús Islao, 2019), donde se mostraron los diagramas de casos de uso, clases, despliegue haciendo uso del UML, estos diagramas dan soporte al desarrollo del sistema. En esta sección se presentan los resultados del uso las herramientas y tecnologías elegidas. En la figura 2 se presenta la instancia (solución) creada en el IDE Visual Studio, que contiene el sistema propuesto (PiScrum_2_0).

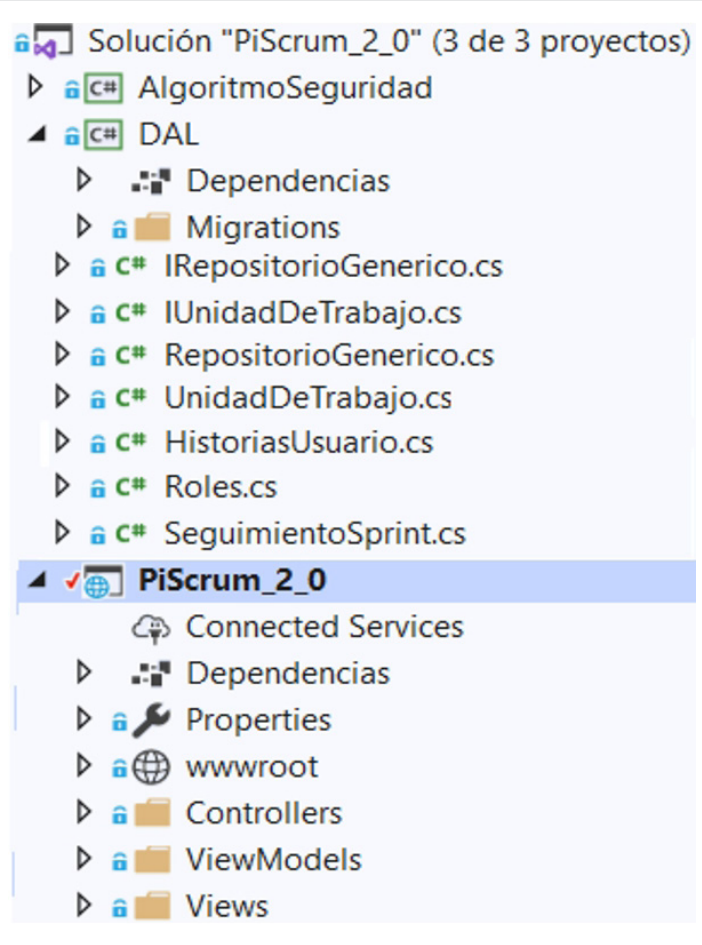

Figura 2. Instancia del sistema propuesto en Visual Studio.

En la figura se puede observar dos proyectos más, AlgoritmoSeguridad y DAL, los cuales proporcionan servicios al proyecto principal (PiScrum_2_0). El primer proyecto proporciona los mecanismos necesarios para el encriptado y descifrado de las contraseñas introducidas por los usuarios a través de las vistas del sistema. El segundo proyecto es la capa de acceso a datos, proporcionando los modelos (clases) y el uso de los patrones unidad de trabajo (UnidadDeTrabajo.cs) y repositorio (RepositorioGenerico.cs). Esta estructura fue proporcionada gracias a la tecnología ASP.NET Core y los modelos y patrones con extensión .CS, se tratan de clases creadas con C#.

En la figura 3 se ilustra pgAdmin 3 para la administración de base de datos en PostgreSQL de manera gráfica.

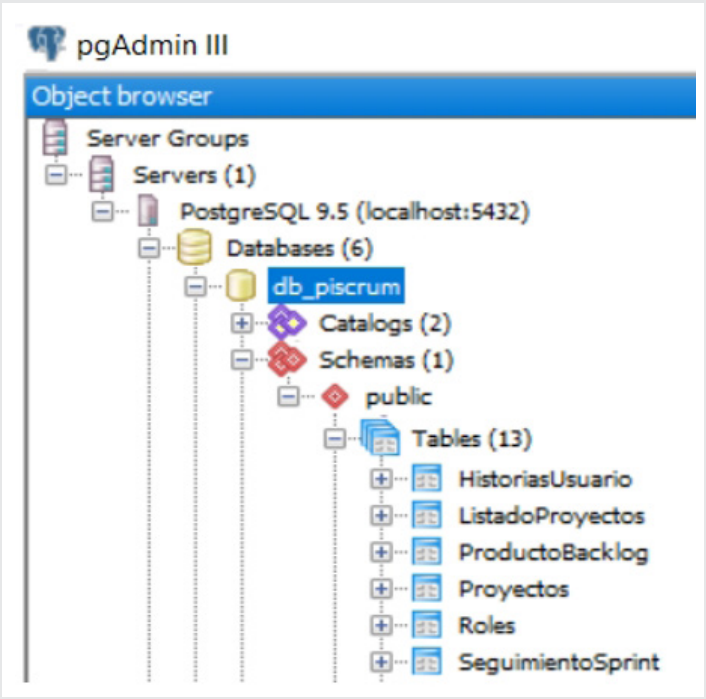

Figura 3. Base de datos del sistema propuesto en pgAdmin 3.

También se puede observar que contiene una base de datos llamada db_piscrum, la cual contiene las tablas generadas a través de los modelos creados en el proyecto DAL con el enfoque CodeFirst proporcionado por Entity Framework Core. Este enfoque además de permitir la migración (creación de una base de datos a partir de los modelos creados). También proporciona herramientas para modificar o actualizar automáticamente la base de datos cuando el modelo cambia (Munro, 2015).

En el proyecto principal (PiScrum_2_0), están contenidos los controladores (carpeta Controller), viewmodels (carpeta ViewModels) y las vistas (carpeta Views) (ver figura 4).

En la figura 4 se puede observar que la carpeta Controllers contiene una clase nombrada HistoriasUsuarioController, la cual contiene la lógica de negocio para la gestión de las historias de usuario en el sistema propuesto,

en la carpeta ViewModels se encuentra una clase llamada HistoriasUsuarioViewModel, que contiene las propiedades para almacenar temporalmente la información proporcionada por el usuario a través de las vistas, y pueda ser mapeada a la clase HistoriasUsuario contenida en el proyecto DAL (ver figura 2). Este mapeo se hace con AutoMapper, este proceso tiene dos funciones, primero crear una capa de abstracción entre los modelos y controladores y la otra es poder realizar transacciones en la base de datos con la tabla HistoriasUsuario (ver figura 3) con ayuda de Entity Framework y de los patrones unidad de trabajo y repositorio. La carpeta Views contiene subcarpetas con los mismos nombres de los controladores, esto para indicar que los archivos contenidos en estas subcarpetas son las vistas de sus controladores correspondientes. En la figura 4, se puede observar que existe una subcarpeta llamada HistoriasUsuario dentro de la carpeta Views, que contiene archivos con extensión cshtml, estos archivos son las vistas creadas con HTML de cada acción que contiene el controlador HistoriasUsuarioController. Con lo expuesto anteriormente, se cumple la estructura que se estableció en la arquitectura que se ilustra en la figura 1.

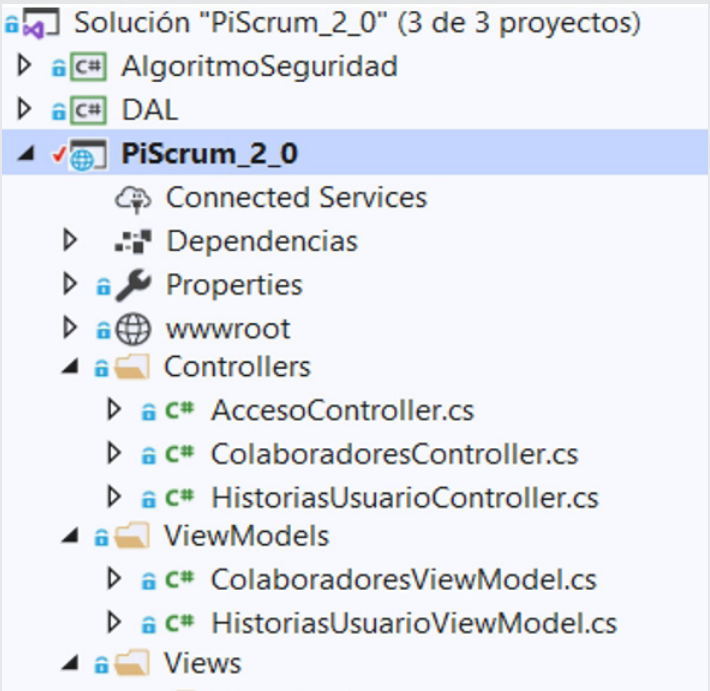

Figura 4. Controladores, ViewModels y Vistas del sistema propuesto.

RESULTADOS

El sistema informático propuesto en este artículo, actualmente está se encuentra en la fase pruebas en la empresa PICCE. A continuación se presentan algunos de los resultados derivados de estas pruebas. Como resultado del desarrollo de la aplicación web se tiene las siguientes vistas del sistema propuesto.

En la figura 5 se ilustra la vista donde los usuarios del sistema pueden consultar los proyectos donde estar involucrados, así como su rol.

Figura 5. Vista para consultar los proyectos gestionados.

Cuando se gestiona un proyecto, la vista principal muestra el product backlog y sus historias de usuario como se muestra en la figura 6, donde el proyecto con nombre Sistema de solitudes en el product backlog cuenta con historias de usuario y un sprint (INVENTARIOS) con sus historias de usuario y tareas contenidas. Todas las páginas cuentan con el menú lateral izquierdo (β) (ver figura 6).

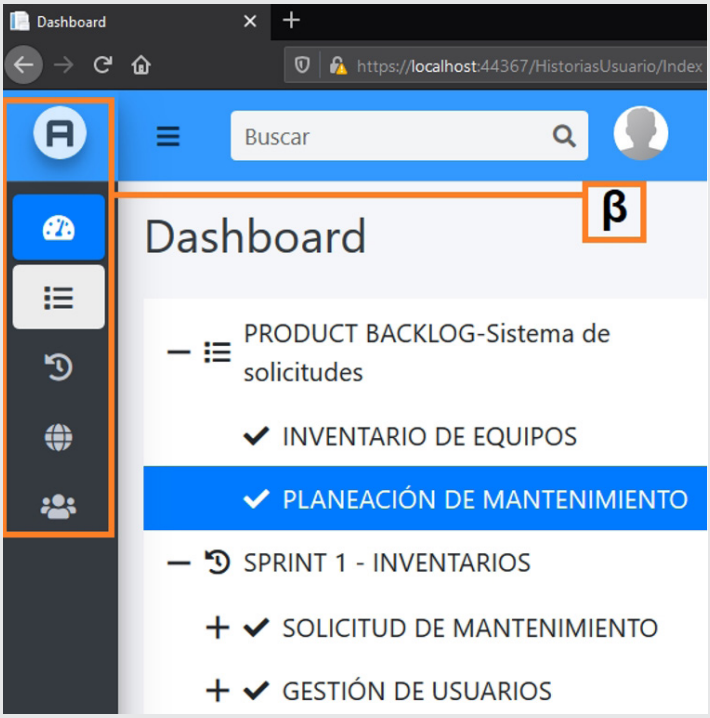

Figura 6. Página para la gestión del product backlog de los proyectos en el sistema propuesto.

Otra funcionalidad que ofrece el sistema es la consulta detallada del seguimiento de un sprint (ver figura 7). En la figura 7 se puede observar el seguimiento detallado de tiempo invertido en cada tarea involucrada en el sprint (INVENTARIOS) que fueron asignadas a cada integrante del equipo de desarrollo.

Consultar seguimiento: Buscar...

				Día	1	2	3	4	5	6	7
SPRINT 1 - INVENTARIOS				Horas pendientes	80	72	65	65	61	61	61
historia	Tarea	Responsable	Estado	Tiempo requerido							
SOLICITUD DE MANTENIMIENTO	Eliminar solicitud de mantenimiento	Raul Morales	FINALIZADO	10	5	0	3	0	0	2	0
GESTIÓN DE USUARIOS	Dar de baja usuario	Angel Rodriguez	FINALIZADO	8	3	4	0	0	1	0	0

Figura 7. Vista del detalle del seguimiento de un sprint.

En la figura 8 se ilustra el burndown chart del seguimiento de un Sprint, donde se grafica lo ideal (β) contra lo real (α).

El burndown chart del sprint (INVENTARIOS) muestra que no hubo retrasos en las tareas debido a que lo real está por debajo de lo ideal (ver figura 8).

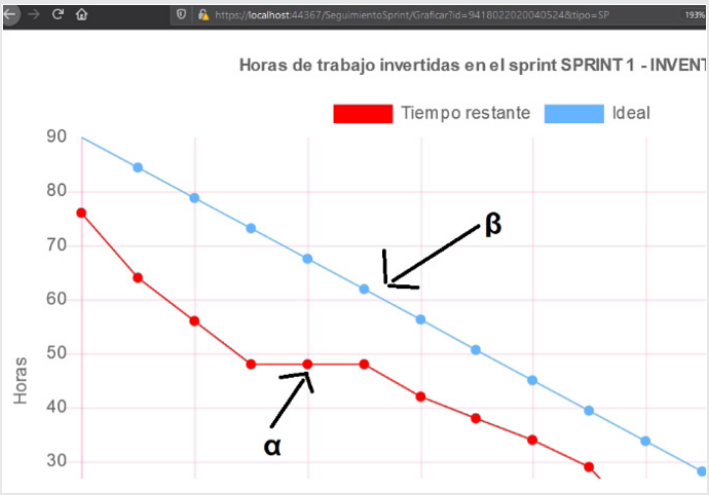

Figura 8. Gráfica del tiempo invertido en el Sprint (burndown chart).

CONCLUSIONES

Derivado de los requerimientos de la empresa PICCE, se provee de una plataforma que permite el control solicitado de las tareas asignadas a los integrantes del equipo de desarrollo y del progreso de cada una de ellas a través de gráficos a través del burndown chart, facilitando la toma de decisiones en caso de existir inconvenientes. Con esta plataforma se podrá administrar correctamente el recurso humano, es decir, que cada una de las personas involucradas en un proyecto tendrá un rol especifico y tareas específicas con la finalidad de tener mayor productividad y mejor calidad en los productos de software de la empresa PICCE. Así como también la administración de los tiempos en cada una de las tareas para alcanzar el objetivo en cada entrega de un sprint, y esto es gracias al detalle del seguimiento del sprint que proporciona la plataforma.

Los usuarios no tendrán la necesidad de realizar una instalación y configuración experta para su uso, además no será requerido de un equipo con características de alto rendimiento debido a que la plataforma es una aplicación web, simplemente debe contar con un navegador web, el cual hoy en día, la mayoría de las plataformas operativas ya cuentan con uno de ellos incluido en el proceso de instalación y configuración de las mismas. También mencionar que PICCE decide contar con su propia herramienta de gestión de desarrollo para no depender de software de terceros.

BIBLIOGRAFÍA

[1] Blankenship J., Bussa M. y Millett S.(2011). Pro Agile .Net Development with Scrum, New York: Apress.

[2] Sommerville I.(2016). Ingenieria de software, México: Pearson.

[3] Munro J.(2015). ASP.NET MVC 5 with Bootstrap and Knockout.js, United State of America: O'Relly.

[4] Ruiz Guadalupe C.M. (2014). Implementación de una red social usando metodologías ágiles para mejorar el proceso de participación estudiantil en la universidad autónoma del Perú, Abril, 10, 2020, http://repositorio.autonoma.edu.pe/bitstream/AUTONOMA/123/1/RUIZ%20GUADALUPE-TELAYA%20ESCOBEDO.pdf.

[5] López Morales J. R., Gazga Portillo J. F., Hernández Bravo J. M. y de Jesús Islao A. D. y Ramírez Silva, R. (2020). Patrones de acceso de una aplicación basada en Scrum para la gestión de proyectos de desarrollo de software, Instituto Tecnológico de La Paz, 273-277.

[6] López Morales J. R., Gazga Portillo J. F., Hernández Bravo J. M. y de Jesús Islao A. D. (2019). Diseño de una herramienta informática basada en la metolodología Scrum para la gestión del desarrollo de software, Academia Journals, 11(6), 789-794.

[7] White S. A. y Miers P. a. D. (2008). BPMN Modeling and Reference Guide, USA: Future Strategies Inc.

[8] Braz A., Rubira C. M.F. y Vieira M. (2015). Development of complex software with agile method, IEEE Xplore, 97-101.

[9] Cervantes Maceda H., Castro Careaga L. y Velasco-Elizondo P.(2016). Arquitectura de software: Conceptos y ciclo de desarrollo, Cuidad de México: Cengage Learning.

[10] Sutherland J. (2014). Scrum: The art of doing twice the work in half the time, United States: Crown Business.

[11] López Morales J. R., Gazga Portillo J. F., Hernández Bravo J. M. y de Jesús Islao A. D. (2019). Propuesta de una herramienta basada en la métodología Scrum para la gestión del desarrollo de software, Academia Journals, 11(2), 1597-1602.

Anexo A Diagrama