Ciencias de la Computación

Ingeniantes

Implementación de un backend orientado a objetos para clusters de BD heterogéneas y clientes web adaptativos asíncronos en un ERP para la gestión integral de los Institutos Tecnológicos del TecNM

RESUMEN: La planeación, organización y dirección escolar son actividades necesarias en una institución educativa, pero la complejidad para la gestión de ellas se incrementa dependiendo del nivel educativo, aunque en el mercado hay disponible software de diferentes desarrolladores, pero ninguno cubre todas las necesidades actuales, por ello se desarrolla "El Sistema Integral del Tecnológico de Misantla", ERP (Enterprise Resource Planning) para integrar, sistematizar y automatizar los procesos docentes y administrativos primordiales del Tecnológico de Misantla sobre una plataforma web. En su desarrollo se usan los principios de la Programación Orientada a Objetos (POO) con la finalidad de implementar la reutilización de componentes de software en los códigos generadores de la presentación y la transferencia de la información entre los clientes y el servidor. Resultando en un modelo de comunicación con el servidor muy simple, transferencia de códigos y datos reducida al mínimo, una eficiente renderización de los componentes de interfaz, y de forma inherente se ha elevado la seguridad del sistema. Logrando con ello la creación de una herramienta de alto nivel para el área académica y administrativa al alcance de todos los involucrados anexando funciones al sistema de coordinación utilizado hasta este momento en el ITSM e instituciones similares.

PALABRAS CLAVE: BackEnd, Componentes, ECMAScript6, FrontEnd, HTML5, PDO, PHP, POO.

Colaboración

Antonio Aquino Ramos Salvador; Francisco Adolfo Aguilar Gómez; Carmen Juliana Aguilar Fernández; José Aurelio Carrera Melchor; Jorge Mario Figueroa García, Instituto Tecnológico Superior de Misantla

ABSTRACT: Planning, organization and school management are necessary activities in an educational institution, but the complexity to manage them increases depending on the educational level, although in the market is available software from different developers, none covers all current needs, so develops "The Integral System of the Technological of Misantla", ERP (Enterprise Resource Planning) to integrate, systematize and automate the educational and administrative processes primordial of the Technological of Misantla on a web platform. In its development, the principles of Object Oriented Programming (OOP) are used in order to implement the reuse of software components in the codes that generate the presentation and the transfer of information between the clients and the server. Resulting in a very simple server communication model, minimized data and code transfer, efficient rendering of interface components, and inherently system security has been raised. Achieving with this the creation of a high-level tool for the academic and administrative area within reach of all those involved, attaching functions to the coordination system used up to this moment in the ITSM and similar institutions.

KEYWORDS: BackEnd, Component, ECMAScript6, FrontEnd, HTML5, PDO, PHP, POO.

INTRODUCCIÓN

Desde 2007 existen diferentes implementaciones de sistemas integrales a nivel institucional en México, podemos mencionar instituciones como el Tecnológico de Monterrey, Universidad

Veracruzana y la Universidad Pedagógica Nacional desarrollados en formato web, el Tecnológico de Misantla tiene un sistema implementado con más de 20 años de antigüedad, por ello se está desarrollando el Sistema Integral del Tecnológico Superior de Misantla (SITM) que permite adecuar a las condiciones actuales los sistemas informáticos que hoy intervienen en los procesos administrativos y académicos del Tecnológico de Misantla y por la necesidad de sistematizar aquellos que aún no lo están. Mediante el SITM, se interconectan los diversos departamentos y/o funciones que intervienen en los procesos de gestión y docencia de la institución y se integra software de terceros y a terceras entidades; se dota así de la interconectividad, posibilidades de expansión y eficiencia que los sistemas actuales de la institución no poseen.

El sistema está dividido en dos subsistemas: Un Backend y un FrontEnd. El BackEnd es un subsistema totalmente autónomo, compuesto por un agente listener extensible y configurable junto con un conjunto de clases que permiten llevar a cabo una representación orientada a objetos de las bases de datos y sus elementos, con la capacidad de interactuar con uno o más clusters de bases de datos heterogéneas [1] que pueden o no estar localizadas en un mismo lugar. El FrontEnd es un subsistema compuesto por un grupo de clientes web asíncronos y adaptativos que extienden una o más clases ECMAScript6 [2] que sirven de plantilla y dotan a sus extensiones de las capacidades necesarias para interactuar con el listener del BackEnd y generar dinámicamente los elementos que componen la interfaz de usuario. Como resultado se ha obtenido un modelo de comunicación con el servidor muy simple, transferencia de códigos y datos reducida al mínimo, una eficiente renderización de los componentes de interfaz, y de forma inherente se ha elevado la seguridad del sistema.

MATERIAL Y MÉTODOS

Objetivos

- Diseñar un Backend Orientado a Objetos para Cluster de Base de Datos heterogéneas reutilizable para tener acceso a diferentes Bases de Datos autónomas preexistentes y futuras.
- Diseñar un FrontEnd adaptativo y asíncrono basado en componentes, encaminado a la reutilización.

Metodología

Por la propia naturaleza del paradigma de programación empleado en este proyecto se ha estado trabajando bajo el análisis y diseño en espiral [3], esencialmente por la capacidad de adaptación gradual de este enfoque de desarrollo, contrastado con el modelo en cascada se gana en flexibilidad, interacción y un mayor ajuste a las necesidades del cliente que pudieran surgir en el desarrollo del proyecto.

Como patrón de diseño se tomó como referencia el Modelo Vista Controlador [4] para generar uno propio, que se adecue a las necesidades del proyecto. Se establecieron las siguientes premisas:

- Mediante la reutilización y el uso de la herencia reducir al mínimo los códigos generadores de la interfaz de usuario y también la transferencia de la información entre los clientes y el servidor.
- Un cliente web debe estar basado o extender un componente plantilla adaptativo.
- Los principios de la POO deben prevalecer a lo largo de todo el desarrollo con la finalidad de promover la creación de componentes de software y la reutilización.
- Todas las herramientas de desarrollo han de ser OpenSource en categoría de estables.
- La comunicación entre los clientes y el servidor siempre será asíncrona y solo será posible mediante el listener del módulo BackEnd, el cual árbitra el acceso al servidor.
- Fases que constituyen el Desarrollo del Sistema.
- Análisis y Diseño del Sistema.
- Desarrollo del Conjunto de Clases que forman una plantilla.
- Desarrollo de los Módulos: Fichas, Cajas, Inscripción, Alumnos, Jefes de Carrera, Docentes y Servicios Escolares.

Diseño del BackEnd

Dada su magnitud, el BackEnd, fue enfocado de lo más simple a lo más complejo con la finalidad de poder factorizar el problema. Secuencialmente fueron programadas las siguientes clases:

Campo.php: Constituye el molde para crear la unidad mínima que puede ser representada en un sistema de bases de datos, el campo. Su constructor permite definir el nombre, el tipo de dato e indicar si se trata de un campo clave. Posee también un método con el cual se puede generar una representación JSON [5] de sí mismo.

Registro.php: Encapsula el conjunto de campos que definen un registro de base de datos. Emulando polimorfismo, su constructor permite crear objetos de diferentes maneras para satisfacer las necesidades de construcción que fueron detectadas durante el diseño. Posee también un método para crear un objeto JSON con la descripción del registro y los valores de sus campos. Estos objetos también sirven de contenedores para las consultas o procedimientos almacenados donde el resultado es un valor o un solo registro. Adicionalmente es capaz de generar el código SQL de las consultas SELECT, INSERT, UPDATE y DELETE tomando como base su propia estructura y los valores de sus campos.

Tabla.php: Los productos de esta clase son capaces de almacenar tantos objetos de tipo Registro como sean necesarios para describir la estructura de una Tabla de base de datos, pueden ser empleados para ejecutar

procedimientos almacenados de la base de datos a la cual hayan sido asociados en donde el resultado sea un valor, un registro o un conjunto de registros y son capaces de contener el dataset resultante de una consulta SQL en un array dinámico de objetos de tipo Registro.

BD.php: Esta clase es una extensión de la librería PDO [6] de PHP 7.0 que consigue conectar de forma transparente a diversos gestores de bases de datos. Es capaz de contener un array de objetos Tablas o resultados de consultas que pueden ser cargadas de inicio para poderlas enviar al cliente como un paquete de datos de tipo JSON. Puede ejecutar las operaciones SELECT, INSERT, UPDATE Y DELETE de un registro ó una o más tablas.

Gestión de la concurrencia.

En un gran sistema, como es el caso del SITM, la gestión de la concurrencia a las bases de datos es un factor determinante para su correcto funcionamiento. En el SITM la gestión de la concurrencia se lleva a cabo mediante la técnica o patrón conocida como singleton, la cual nos permite gestionar las conexiones para hacer posible que exista una y solo una conexión por cliente, independientemente del número de solicitudes que éste realice. Las clases descendientes de la clase BD del proyecto quedan habilitadas para implementar un Singleton y así evitar la sobrecarga de conexiones que podría ocasionar una solicitud masiva de peticiones de un cliente. Por defecto, el constructor de la clase conecta con un gestor MySQL y a petición se puede realizar la conexión a otros gestores como son MSSQL, Oracle, PostgreSQL, SQLite, etc., de forma transparente. Las siguientes líneas muestran el singleton creado para conectar con un gestor de base de datos MSSQL ubicado en un segundo servidor. Todos los singleton pertenecen a clases derivadas de la clase BD.php

```
        // Singleton para el manejo de la conexión a
MSSQL
        private static $BD;
        public static function BD(){
                if (!isset(self::$BD))
self::$BD = new BD("E00758\SMISANTLA", "gas",
"user", "clave", "MSSQL");
                return self::$BD;
        }
```

Gestión del cluster de base de datos heterogéneas y comunicación con los clientes.

Se tomó la decisión de crear un cluster de base de datos y no una mega base de datos para simplificar su gestión, mantenimiento y primordialmente mantener la capacidad de interactuar con otros sistemas sin la necesidad de llevar a cabo una migración de datos o adecuar las consultas a cada gestor, simplemente solicitando la ejecución de procedimientos almacenados alojados en cada instancia, favoreciendo colateralmente un acceso seguro a los datos.

Se programó la clase "Proceso.php" con la finalidad de que sea extendida para crear clases que definan el comportamiento de objetos cuyas funciones serán trabajar como un listener de las peticiones POST que llegan al servidor y árbitro de acceso a las bases de datos. Las extensiones de esta clase, verifican que la solicitud sea una operación reconocida, que la naturaleza de los datos de solicitud sea la esperada y si todo es correcto, es él, y no el cliente, quien accede al cluster para entregar los resultados de la consulta solicitada.

Si la solicitud no es reconocida, el listener asumirá una amenaza de ataque y devolverá un mensaje de error que quizás terminaría ocasionando un mal funcionamiento en el cliente. Cada cliente que se comunique con el server debe hacerlo mediante una petición asíncrona de tipo AJAX [7] y enviar/recibir datos en formato JSON. La interacción del listener con el cluster de bases de datos y los clientes queda representada en la Figura 1.

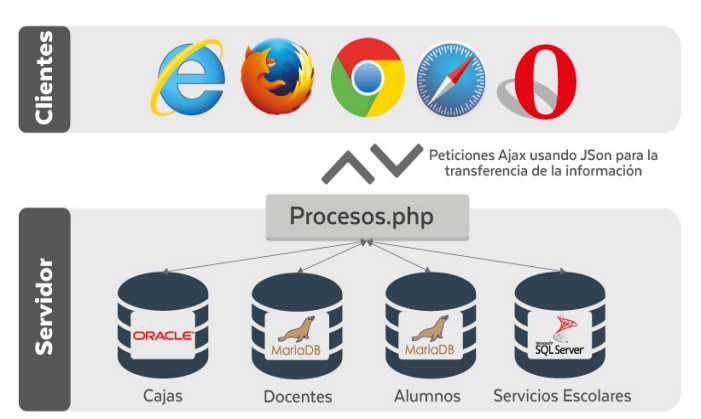


Figura. 1. Interacción del BackEnd y el FronEnd.

Diseño del FrontEnd

Se tomó como punto de partida el patrón de arquitectura: Modelo Vista Controlador y se adaptó un motor de comunicaciones asíncronas propio basado en Ajax. (Véase Anexo 1) muestra el esquema aplicado.

Vista: Los clientes fueron creados bajo los principios de la web adaptativa establecidos por Ethan Marcotte en 2009 [8]. Marcotte indica que existen tres elementos fundamentales para lograr un diseño web adaptativo, que son: cuadrícula fluida, imágenes flexibles y media queries. Gran parte de la solución la proporciona un buen manejo de CSS3 Flexbox y una parte la programación en Javascript para adaptar la interfaz de usuario a las resoluciones de pantalla que tienen los diferentes dispositivos.

Representación de la información orientada a objetos.

Dado el número de veces que se ocupa la terna campo, registro, tabla, se programaron las tres clases si-

guientes con el fin de conseguir un modelo orientado a objetos de la información a representar:

Campo.js: Esta clase se utiliza para crear un campo que es la unidad mínima que contiene el dato JSON que estamos recibiendo, en el constructor se recibe nombre y contenido de dicho campo, estos datos se muestran en la interfaz por medio de un contenedor de tipo span.

Registro.js: Permite la agrupación múltiples objetos de tipo Campo, tantos como sea necesario incluir dentro de un mismo registro, para mostrarlos en la interfaz se utiliza un contenedor de tipo div.

Tabla.js: Con esta clase se define una estructura que puede agrupar uno o más objetos de tipo registro según se requiera, esta agrupación que se hace con la ayuda del contenedor div permite representar la información al usuario final como una tabla.

Un ejemplo de aplicación de este conjunto de clases lo representa la forma en que se gestionó la descripción de las materias y su conjunto, como se muestra en la Figura 2.

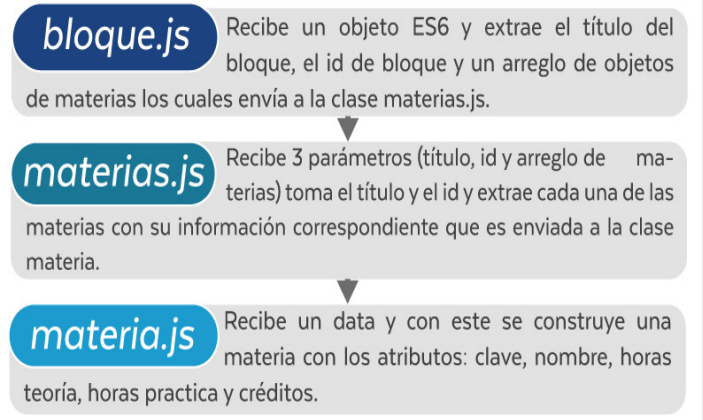


Figura. 2. Representación OO de la información.

Controlador: El control de la vista se lleva a cabo mediante la instanciación de tres clases cuya finalidad es la siguiente:

modulo_ev.js: Es la clase que se encarga de la gestión de los eventos generados por el usuario.

modulo.js: Es una extensión de la clase yTube.js, que lleva a cabo la actualización de la vista y mantiene las callbacks resultantes de las solicitudes Ajax dirigidas al servidor.

yTube.js: Es el componente por el cual, mediante su extensión, sirve de plantilla a la vista manejada por la instancia de la clase modulo.js. Es también la responsable de: designar la estructura y áreas correspondientes para insertar cada componente, definir la estructura de los componentes principales (Menú lateral, Menú superior y encabezado), definir el comportamiento y las propiedades de cada componente principal, realizar el insertado de cada componente principal al finalizar su creación.

Interfaz de Comunicación:

El segmento de interfaz de comunicación está constituido por la clase denominada ajax.js que crea y envía peticiones asíncronas de métodos AJAX derivados de la librería JQuery consumiendo objetos tipo EcmaScript6 que son convertidos a JSON para ser enviados al servidor, este a su vez devuelve una respuesta en el mismo formato que convierte en objetos EcmaScript6 para ser dirigidos al cliente.

RESULTADOS

El desarrollo de este proyecto es un punto de partida para lograr la completa automatización de los diferentes servicios que al sistema integral le competen, sumado a lograr la unificación de los diferentes departamentos solidificando un formato global para la consolidación de datos del alumno/docente/jefe de carrera.

En el diseño de sus interfaces, el proyecto aplica nuevas tecnologías ya estandarizadas considerando las distintas características de los dispositivos más utilizados, el uso de flexbox en gran medida fue la solución para que la distribución de la página se adapte al tamaño de la pantalla del dispositivo. (Fig. 3).

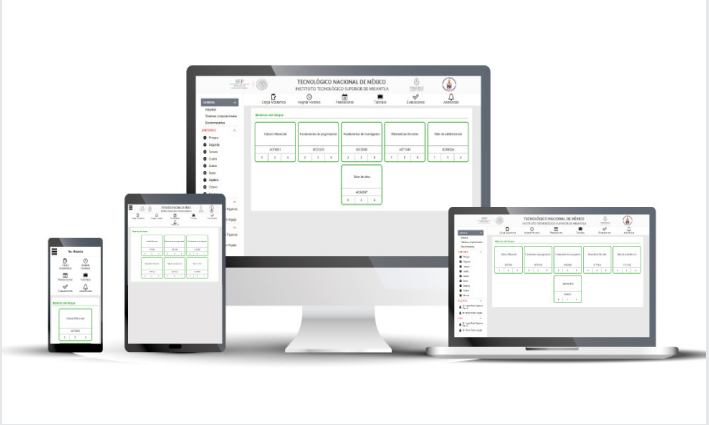


Figura 3. Diseño adaptativo mostrado en diferentes dispositivos de izquierda a derecha Celular con una pantalla de 5'', Tablet con Pantalla de 10'', Equipo de Escritorio con una pantalla de 19'', Laptop con una pantalla de 15''.

Mediante la creación dinámica de componentes, se logra eliminar más del 90% de las líneas de código de la interfaz de ellos, obteniéndose una significativa reducción en la taza de transferencia de este, agilizándose la carga de los diferentes módulos.

La estructura del documento HTML sigue la recomendación de la W3C, que consta de header, aside, nav y section. Como se aprecia en la Figura 4.

Figura 4. Estructura HTML principal.

La plantilla/componente del FrontEnd opera ya satisfactoriamente, se mantiene sujeto a las adecuaciones que requiera el desarrollo en espiral.

Se anexa Imagen UML del caso de estudio del Sistema FrontEnd en el cual se aprecia como el Usuario/cliente interactúa con la plantilla para comunicarse con el servidor (Véase figura Anexo 2).

Los módulos: "Solicitud de Ficha", "Inscripciones" y "Caja" están totalmente terminados, probados y espera de implantación. Se encuentran en desarrollo a un 50% los módulos de; "docentes", "jefes de carrera", "alumnos" y "servicios escolares".

Con respecto al BackEnd, este se encuentra terminado en su totalidad, superando diversas pruebas de funcionamiento y, además, está operando de manera adecuada en otros proyectos.

CONCLUSIONES

El proyecto SITM se encuentra en fase de desarrollo y espera estar concluido, implantado y en producción para el mes de junio de 2018. Su enfoque orientado a componentes permite establecer un sistema de mantenimiento en cascada y hacerlo crecer mediante la integración de nuevos módulos que cumplan las premisas de desarrollo que fueron establecidas. La capacidad de reutilización del módulo BackEnd y los componentes de la plantilla del FrontEnd podrán ser utilizados en futuros desarrollos de software. Al concluir la primera implantación del sistema se procederá a agregar un módulo que integre al SITM la gestión de la biblioteca, tutorías y los exámenes en línea.

BIBLIOGRAFÍA

[1] Pippal K.S., Kushwaha, D.S. (2013) A simple, adaptable and efficient heterogeneous multi-tenant database architecture for ad hoc cloud, Journal of Cloud Computing: Advances, Systems and Applications, pp. 1-14.

[2] Ecma International (2017). Standard Ecma-262: ECMAScript 2017 Language Specification [online] Disponible en: http://www.ecma-international.org/ecma-262/6.0/ECMA-262.pdf [26 de agosto de 2017].

[3] Pressman S. R. (2010). Ingeniería del Software un enfoque práctico (pp. 33-43), Madrid: McGrawHill.

[4] Fernández, Y., Díaz Y. (2012). Patrón Modelo-Vista-Controlador. Revista Telem@tica, 11(1), 47-57.

[5] Ecma International (2013), Standard Ecma-404: The JSON Data Interchange Format. [online] Disponible en: https://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf [31 de agosto de 2017]

[6] Popel D., Chittar L. (2007) Learning PHP Data Objects: A Beginner's Guide to PHP Data Objects, Database Connection Abstraction Library for PHP 5, PACKT, pp. 6-44.

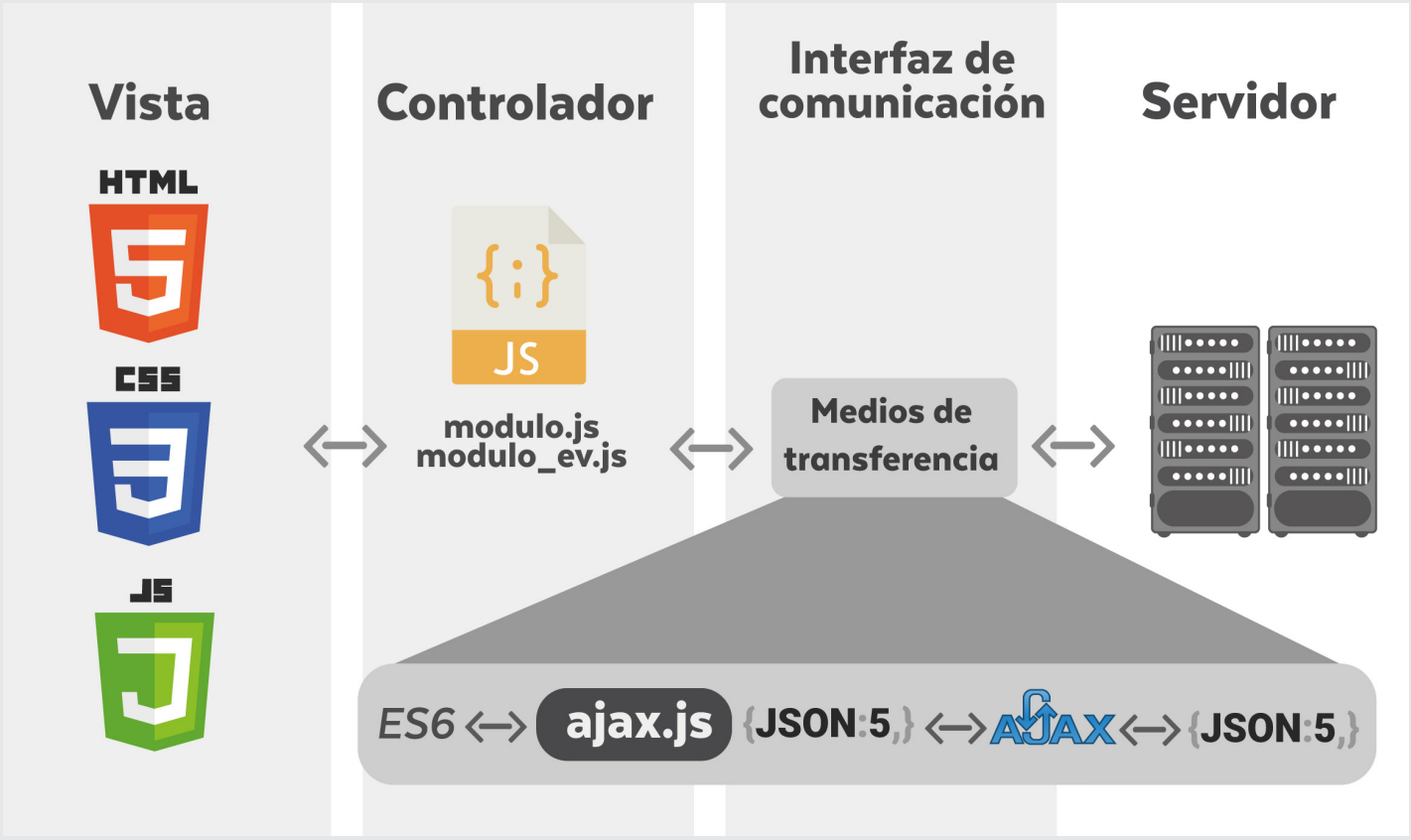
[7] Woychowsky, E. (2008). Ajax Creating Web Pages with Asynchronous JavaScript and XML (pp.20-134), Indiana: Pearson Education.

[8] Marcotte, E. (2011), Responsive web design (pp. 52-124). New York: A Book Apart.

Agradecimientos

Agradecemos el apoyo brindado por el Consejo Nacional de Ciencia y Tecnología (CONACYT) y al Instituto Tecnológico Superior de Misantla por las facilidades otorgadas para la realización del presente proyecto.

Anexo 1. Adaptación del MVC al proyecto.



Anexo 2. Diagrama UML Caso de estudio del sistema.