Artículo **Revista de Sistemas Computacionales y TIC's**
Marzo 2017 Vol.3 No.7 7-12

Desarrollo de aplicación web con herramientas MEAN para el procesamiento y comunicación de información de estudiantes en la UT de Puerto Peñasco

AMPARAN-DURAN, Rafael Gibrán†*

Universidad Tecnológica de Puerto Peñasco

Recibido Octubre 11, 2016; Aceptado Enero 20, 2017

Resumen

* El procesamiento y comunicación de información relacionada con los datos de estudiantes dentro de una institución educativa, como son calificaciones, bajas, documentos de ingreso, titulación y registro de horas de servicio, es una tarea de gran demanda dentro de los departamentos administrativos, los cuales deben de formar procesos para manejar dicha información de manera organizada, ágil y precisa. En el siguiente artículo se plantea el diseño y descripción del desarrollo de un sistema web altamente asíncrono haciendo uso de las herramientas MEAN (MongoDB, ExpressJS, AngularJS y NodeJS) para la comunicación y procesamiento de información en un sistema escolar web desarrollado enteramente en Javascript. El propósito es generar un prototipo funcional de código abierto que cualquier grupo de desarrollo de software conformado por académicos y estudiantes dentro de instituciones educativas pueda tener acceso, para que a mediano plazo, cada institución educativa pueda probar y personalizar la aplicación a sus necesidades, incrementar sus funcionalidades y compartir sus herramientas con otros desarrolladores.

Motor gráfico, interfaces cerebro computadora (BCI), aprendizaje de las TI

Abstract

The processing and communication of information related to students data within an educational institution, such as grades, drop-offs, personal documents, requirements for obtaining their degree and hours of service registration, is a task of great demand within administrative departments, which must form processes to handle such information in an organized, agile and precise manner. The following article discusses the design and description of the development of a highly asynchronous web system using MEAN tools (MongoDB, ExpressJS, AngularJS and NodeJS) for the communication and processing of information in a web system developed entirely in Javascript. The purpose is to generate an open source functional prototype that any software development group made up of academics and students within educational institutions could have access, so that in the medium term, each educational institution can test and customize the application to their needs, increase their functionality and share their tools with other developers.

Graphic engine, Brain - Computer interfaces, Learning of IT

Citación: AMPARAN-DURAN, Rafael Gibrán. Desarrollo de aplicación web con herramientas MEAN para el procesamiento y comunicación de información de estudiantes en la UT de Puerto Peñasco. Revista de Sistemas Computacionales y TIC'S 2017, 3-7: 7-12

* Correspondencia al Autor (Correo Electrónico: gibranamparan@utpp.edu.mx)
† Investigador contribuyendo como primer autor.

Introducción

Debido a la profunda integración de los sistemas de información en cada una de las actividades que realizamos tanto en nuestra vida personal como laboral, la demanda de sistemas informáticos robustos y de calidad lanza un fuerte reto a los desarrolladores para crear aplicaciones altamente incrementables, de alto rendimiento, velocidad, calidad y bajo costo, lo cual solamente es posible aplicando arquitecturas que dividan claramente la separación de responsabilidades entre cada uno de los componentes que integran el sistema para el desarrollo de aplicaciones altamente modulares que permitan la reutilización componentes de manera flexible. En este artículo se describe el diseño fundamental de un sistema para la administración de información de estudiantes hecho con tecnologías de código abierto y libre para su adaptación dentro de otras instituciones que necesiten resolver problemas similares, dando énfasis en transacciones asíncronas y notificaciones en tiempo real, todo desarrollado en Javascript.

Planteamiento del problema

El trabajo publicado previamente (Amparan Duran, 2016) aborda el planteamiento del problema desde una perspectiva de análisis y recolección de requerimientos de usuario, donde se identifican las diferentes entidades que participan dentro de la aplicación, así como las principales funciones del sistema. Sin embargo, una vez identificados los requerimientos, se procede a determinar qué características técnicas debe de cumplir el desarrollo de la aplicación para poder representar una ventaja competitiva con respecto a otros sistemas existentes.

La problemática que se aborda en este trabajo, es realizar una propuesta de diseño y desarrollo de una arquitectura que permita la presentación de datos y comunicación automática de los usuarios y a su vez que permita conservar en el software las propiedades de fácil mantenimiento, escalabilidad y flexibilidad aún costa del crecimiento que pudieses tener, lo que a su vez permita a cualquier institución, poder desarrollar módulos personalizados con facilidad.

Tecnologías utilizadas

Las posibilidades que ofrecen las tecnologías basadas en Javascript dentro del conjunto de herramientas MEAN (Haviv, 2014) para el desarrollo de front-end como es Angular 2, permitirán lograr un alto nivel de cohesión y bajo grado de acoplamiento a través de la aplicación de conceptos como módulos, componentes y servicios, los cuales son implementados por la librería de Angular 2, programada enteramente en Typescript (lenguaje super-conjunto de Javascript).

Implementar Angular 2 de patrón MV* (Google Angular, 2017) de manera desacoplada y con un API-Rest desarrollado en el server, permitiría en futuros proyectos reutilizar el acceso a los mismos datos desde otras aplicaciones hechas sobre otras plataformas como por ejemplo aplicaciones móviles o aplicaciones de terceros.

Con un servidor en ExpressJS sobre NodeJS, se pretende atender los problemas de rendimiento ante una posible alta demanda, a través de ejecución orientada a eventos en vez de multihilo, como lo haría un servidor convencional (Tilkov & Vinoski, 2010). Estas herramientas soportan websockets para reportar eventos en tiempo real utilizando SocketIO (Azaustre, 2015).

ISSN-2444-5002

Finalmente, la integración de una base de datos NoSQL como MongoDB termina de homogeneizar el lenguaje de desarrollo de desarrollo de todos las partes del sistema a Javascript, completando así el conjunto de herramientas MEAN. Además, permite explotar los beneficios que tiene este tipo de servidores de bases de datos en cuanto a rendimiento se refiere (mongoDB, 2016), lo se cubre con más detalle en la publicación previa (Amparan Duran, 2016).

Diseño del lado del cliente

La arquitectura manejada para el diseño de la aplicación permite concentrar prácticamente todo el trabajo en el desarrollo del front-end sobre una aplicación Angular 2, donde a cada entidad le corresponden un juego de componentes y servicios. Los componentes representan secciones del front-end y porciones de estas secciones, mientras que todo el código relacionado con la adquisición de datos es conferido a los servicios, los cuales ejecutan llamadas HTTP al API desarrollado del lado del servidor.

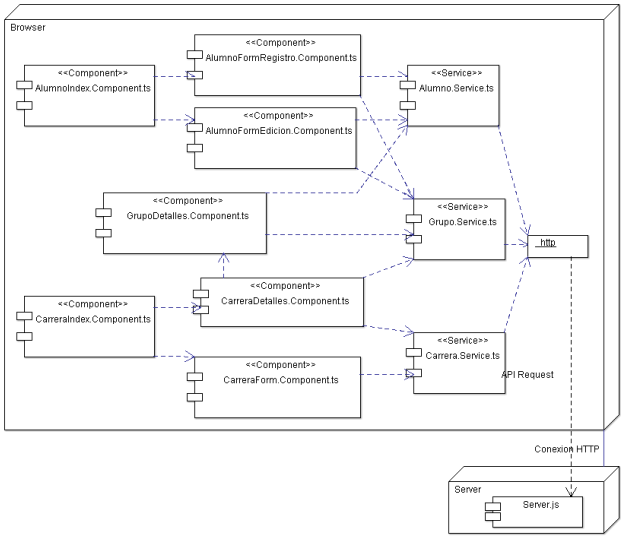

Figura 1 Diagrama de componentes del front-end. Elaboración propia

Dentro del diseño ilustrado en el diagrama de componentes mostrado en la figura 1, se ha categorizado como módulos todos aquellos conjuntos de recursos en el front-end que estén relacionados con las funciones principales ejecutadas sobre su entidad correspondiente.

Cada módulo está compuesto principalmente de 2 tipos de clases definidas como componentes y servicios. Estos dos tipos de recursos tienen propósitos muy específicos dentro del desarrollo de las vistas:

- Componente: Sobre este se declara toda la lógica correspondiente a una vista o un segmento de ella, invocando servicios para obtener datos y guardarlos en variables internas para ser ubicados en pantalla.
- Servicios: Aunque no existe una definición concreta dentro de Angular 2 de este tipo de componente, si es referido en la documentación como un tipo de archivo con la tarea de proveerle recursos específicos al componente para poder funcionar. Para esta aplicación, los servicios son utilizados para codificar las solicitudes de datos al servidor haciendo llamadas asíncronas al API hospedado en el servidor.

Diseño del lado del servidor

El servidor está desarrollado sobre una plataforma NodeJS con ExpressJS. Estas herramientas permiten configurar accesos a bases de datos, recursos y administración del hilo principal de ejecución y memoria para atender a las solicitudes HTTP de una gran cantidad de clientes conectados a la vez, a través de declaraciones de eventos que se activan sólo en el momento que un cliente intenta hacer conexión.

El objeto que permite atender dichas solicitudes se llama Router (Node.js Foundation), y sobre de este se declaran funciones según el verbo de la solicitud HTTP (GET, POST, PUT, etc.) para responder al cliente con algún recurso especifico proveniente del servidor. Como se puede observar en el diagrama de componentes de la figura 2, se muestra una distribución de componentes para la atención de llamadas HTTP en el servidor.

ISSN-2444-5002

Para solicitar datos por medio de Mongoose, el componente Router permitirá desarrollar nuestro API-Rest al asociar solicitudes HTTP con bloques de código de servidor programados sobre controladores. Las responsabilidades de acceso de información y seguridad son separadas por entidad en diferentes Routers. Este diseño es incrementable repitiendo la arquitectura de módulos, servicios y controladores en servidor para cada entidad que vaya requiriendo el crecimiento del sistema.

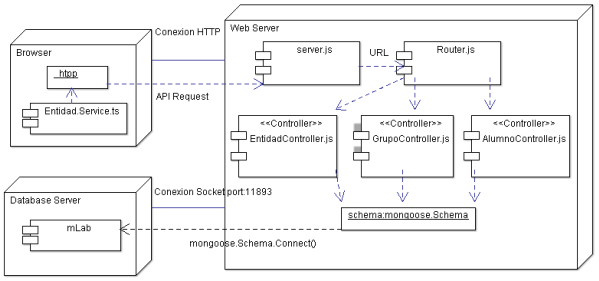

Figura 2 Diagrama de componentes del backend. Elaboración propia

La base de datos Mongo se configuró sobre el servicio mLAB, y se utiliza Mongoose (Learnboost, 2017), una herramienta de mapeo objeto-relacional (ORM), que contiene toda una serie de objetos y métodos para diseñar un modelo representativo de la base de datos sobre nuestro servidor y ejecutar solicitudes obteniendo los resultados en formato JSON.

Notificaciones y comunicación en tiempo real

Una de las herramientas que contribuyen a incrementar el valor agregado del sistema con respecto a otros del mismo tipo, son las notificaciones y comunicación en tiempo real. El diagrama de secuencias de la figura 3 muestra un ejemplo de ejecución de un alta de un nuevo registro de alumno, lo cual es notificado a ambos usuarios conectados a la aplicación al mismo tiempo y de forma asíncrona, lo mismo sucede si el otro usuario decide eliminar dicho registro. Esto es logrado a través del uso de websockets con SocketIO (Azaustre, 2015), sobre el cual se declaran eventos personalizados para notificar a uno o varios cliente en tiempo real información según un evento detonado del lado del servidor.

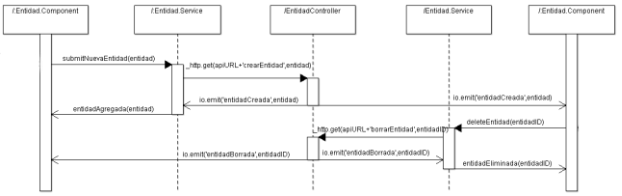

Figura 3 Diagrama de secuencia de ejemplo de alta y baja de registro de alumno. Elaboración propia

Debido a que la gestión administrativa de datos de estudiantes es manipulada por más de un departamento, es importante que los usuarios conectados al sistema sean comunicados al instante cada vez que haya modificaciones, lo cual contribuye en gran medida a la coordinación entre departamentos.

Resultados

Como ejemplo, consideremos la figura 4, donde se muestra un panel de monitoreo de preinscripciones y una forma de registro de preinscripción, el registro de estudiantes y el monitor de preinscripciones reflejan los cambios de la base de datos en tiempo real.

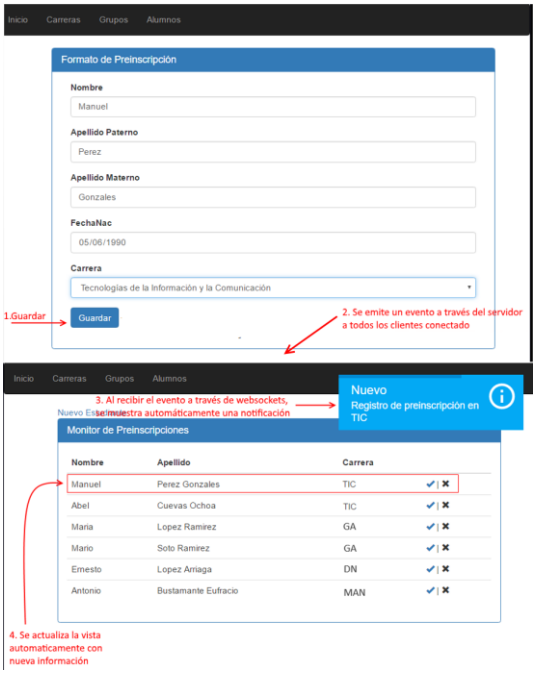

Figura 4 Ejemplo web sockets sobre interfaz de usuario. Elaboración propia

Se puede observar en la sección de anexos los bloques de código que permiten esta operación, donde el futuro estudiante ingresa sus datos (bloque A), comunicados al servidor a través de un servicio (bloque B), para ser almacenados emitiendo un evento (bloque C) y a su vez, servicios escolares monitorea en tiempo real el crecimiento de la tabla de preinscripciones (bloque D). Para tener acceso al código completo y dar seguimiento al desarrollo, puede acceder al código en el repositorio de Github (Amparan Duran, Github.com, 2017).

Conclusiones

Actualmente el sistema se encuentra dentro de un proceso incremental de pruebas y desarrollo iterativo. El módulo de preinscripción se aplicará sobre los procesos ejecutados dentro del departamento de servicios escolares en integración con el módulo de pagos en el departamento de finanzas en la Universidad Tecnología de Puerto Peñasco para el ciclo escolar que inicia en agosto del 2017.
Al aplicar y conocer las capacidades de este tipo de tecnologías, es de resaltar la importancia de resolver problemáticas de gestión escolar aplicando herramientas de última generación y de código abierto para abrir la puerta al trabajo colaborativo libre por diferentes academias de desarrollo de software de otras universidades

El código y características técnicas mas detalladas se encuentran disponibles en el repositorio Github (Amparan Duran, Github.com, 2017), actualmente sobre la aplicación se encuentran haciendo publicaciones diarias de nuevas funcionalidades y modificaciones.

Anexos

Bloques de código correspondientes al ejemplo ilustrado en la figura 4.

A) formaAlumno.component.ts

```
//Evento ejecutado por el submit de la forma
agregarAlumno(){
        this._alumnoService
        /*Envia datos de estudiante a servidor para
ser guarados*/
        .addAlumno(this.nuevoAlumno)
        .subscribe(
                //Si el servidor responde
                (data:Alumno)=>{
               /*Toma registro del servidor    con ID
            y asociaciones ya generadas*/
                        this.nuevoAlumno = data;
                },
                error=>alert(error),
                ()=>console.log('done!')
        );
}
```

B) alumno.service.ts

```
/*Servicio con la responsabilidad especifica de
solicitar al servidor almacenar un registro de
alumno*/
addAlumno(Alumno){
        return this._http.post(this.domain,Alumno)
        .map(res=>res.json())
}
```

C) AlumnoRouter.js (Servidor)

```
//Ejecuta Metodo para salvar nuevo registro
nuevoAlumno.save(function(err){
    if(err){console.log(chalk.red('error al guardar
alumno'));
        res.send(err);
    }else{
        console.log(chalk.green('AlumnoCreado'));
        nuevoAlumno._carrera = carrera;
        //Con Socket.io Se emite evento a todos los
clientes
        io.sockets.emit('AlumnoCreado',nuevoAlumno);
        res.json(nuevoAlumno);
    }
});
```

D) alumnoIndex.component.ts

```
/*Evento que se ejecuta al detectar mensaje emitido
por servidor para indicando que un nuevo registro fue
creado*/
this.socket.on('AlumnoCreado', function(data){
  //Se actualiza los datos de la vista
  this.Alumnos.push(data);
  this.sortAlumnosByDate();

  //Se genera y muestra notificación en pantalla
  let carrera = data._carrera.abreviacion;
  this._notificationsService.info("Nuevo",
        "Registro de preinscripción en "+carrera);
}.bind(this));
```

Agradecimiento

Agradezco el apoyo de PRODEP por la aprobación de este proyecto dentro de la convocatoria de nuevo PTC, a los estudiantes de TIC Osuna Talamantes Adrian y Roque Morales Alexis Giovanny de la Universidad Tecnológica de Puerto Peñasco de la generación por servir de apoyo técnico en el desarrollo de este proyecto aún en construcción y a la misma administración de la Universidad por servir como plataforma para la implementación piloto y el desarrollo del mismo.

Referencias

Amparan Duran, R. G. (2016). ScholarNode: Implementación de MongoDB, Express-Sails, Angular y NodeJS en desarrollo de sistema de gestión escolar. En 6º. Congreso Nacional De Tecnologías De La Información Y Comunicación; Conatic 2016.

Haviv, Amos Q. "Introducing MEAN." En "MEAN Web Development", de Amos Q. Haviv, 10. Birmingham, Mumbai: Packt Publishing, 2014

Google Angular. (2017). ARCHITECTURE OVERVIEW. Obtenido de angular.io: https://angular.io/docs/ts/latest/guide/architecture.html (último acceso: 8 de Feb de 2017)

Tilkov, Stefan, y Steve Vinoski. "Node.js: Using JavaScript to Build High-Performance Network Programs." IEEE Internet Computing (Volume: 14 , Issue: 6), 2010: 80 – 83

Azaustre, C. (2015 de Septiembre de 24). WEBSOCKETS: CÓMO UTILIZAR SOCKET.IO EN TUS APLICACIONES WEB. Obtenido de https://carlosazaustre.es: https://carlosazaustre.es/blog/websockets-como-utilizar-socket-io-en-tu-aplicacion-web/ (último acceso: 15 de Feb de 2017)

mongoDB. "Top 5 Considerations When Evaluating NoSQL Databases.", mongodb.com, Junio de 2016. https://www.mongodb.com/thank-you/white-paper/nosql-considerations (último acceso: 3 de agosto de 2016)

Node.js Foundation. (2017). Routing. Obtenido de Express: https://expressjs.com/en/guide/routing.html (último acceso: 8 de Feb de 2017)

Learnboost. (2017). Schemas. Obtenido de MongooseJS: http://mongoosejs.com/docs/guide.html (último acceso: 8 de Feb de 2017)

Amparan Duran, R. G. (2017). ScholarNode Repository. 30 de Ene de 2017. https://github.com/gibranamparan/ScholarMean1 (último acceso: 15 de Feb de 2017

ISSN-2444-5002