INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE OCCIDENTE

Departamento de Electrónica, Sistemas e Informática

Desarrollo tecnológico y generación de riqueza sustentable

PROYECTO DE APLICACIÓN PROFESIONAL (PAP)
PAP PROGRAMA DE CIUDADES INTELIGENTES

4L05 Vida Digital
Actualización backend del sistema de monitoreo WSN

PRESENTAN

Programas educativos y Estudiantes

Ing. Sistemas Computacionales. Carlos Soto Pérez

Ing. Sistemas Computacionales. Héctor Antonio Chávez Morales

Profesor PAP: Mtro. Luis Eduardo Pérez Bernal

Tlaquepaque, Jalisco, diciembre de 2020

ÍNDICE

Contenido

Tabla de ilustraciones

REPORTE PAP

Presentación Institucional de los Proyectos de Aplicación Profesional

Los Proyectos de Aplicación Profesional (PAP) son una modalidad educativa del ITESO en la que el estudiante aplica sus saberes y competencias socio-profesionales para el desarrollo de un proyecto que plantea soluciones a problemas de entornos reales. Su espíritu está dirigido para que el estudiante ejerza su profesión mediante una perspectiva ética y socialmente responsable.

A través de las actividades realizadas en el PAP, se acreditan el servicio social y la opción terminal. Así, en este reporte se documentan las actividades que tuvieron lugar durante el desarrollo del proyecto, sus incidencias en el entorno, y las reflexiones y aprendizajes profesionales que el estudiante desarrolló en el transcurso de su labor.

Resumen

Este resumen contiene la descripción de las actividades, herramientas y planeación realizadas por el equipo de backend en el proyecto “Sistema de monitoreo ambiental” para la modernización del servicio web existente. El objetivo de este proyecto es modernizar el backend existente a una tecnología más nueva, como lo es Node JS, para facilitar un futuro mantenimiento y actualización de este servicio web y empatarlo con las tecnologías que se enseñan en el ITESO.

Este documento describe las actividades que se realizaron para llegar al resultado pactado en la planeación, así como los ajustes a esta planeación e interacciones, ya sea entre otros equipos en el proyecto o internamente. Además, este documento incluye las reflexiones realizadas a nivel personal y de equipo de los aprendizajes profesionales, sociales, éticos y personales obtenidos por la realización de este proyecto

Para proporcionar un mejor detalle de las actividades, el documento cuenta con 3 anexos, el primero, Anexo A, muestra la primea versión del plan de trabajo, el Anexo B muestra el reporte técnico y el último, Anexo C el reporte de pruebas.

1. Introducción

1.1. Objetivos

El ITESO cuenta con una red de nodos-sensores en el bosque de la primavera, estos nodos envían datos a un servicio web para el almacenaje de las lecturas en una base de datos para su posterior manejo. Para consultar la información existe un sitio web en donde se representa de una forma visual todos estos datos o también mediante la descarga de un archivo. Lo que se busca trabajar en este proyecto es terminar de modernizar la restAPI hacia una tecnología más moderna, además generar una documentación de la nueva versión, para así lograr un mejor mantenimiento y actualización a las tecnologías que se enseñan en el ITESO. Se busca migrar el 100% de los servicios antiguos a la nueva versión, así como dotar al nuevo servicio web de la habilidad de manejar usuarios y permisos.

1.2. Justificación

Este proyecto busca aportar información a “Anillo primavera A.C” para que tengan una herramienta de medición confiable y siempre disponible con la cual puedan recopilar información de manera constante y, en caso de ser necesario, advertir a las autoridades gubernamentales correspondientes sobre alguna situación específica relacionada con el bosque de la primavera.

Es importante contar con fuentes de datos para poder realizar una mejor toma de decisiones, este proyecto busca ser una fuente de datos de confianza tanto para decisiones como para investigaciones.

Estos datos pueden ayudar a mostrar mejor la realidad del bosque y así mismo ayudar a la carrera de Ingeniería Ambiental en la recolección de mediciones para las prácticas que realicen.

1.3 Antecedentes

Dada la invasión urbana en el bosque de la primavera, los constantes incendios, los requerimientos de la carrera Ingeniería Ambiental y la posesión del ITESO de terrenos en el bosque de la primavera convergen en este proyecto de red de sensores de monitoreo.

Anteriormente sin la existencia de este método de medición ambiental, no era posible mantener un registro detallado y constante de los problemas que aquejaban al bosque ya mencionados anteriormente.

1.4. Contexto

Actualmente se cuentan con 15 nodos sensores, repartiendo 10 de ellos dentro del bosque en los terrenos que el ITESO tiene a su disposición y el resto en el campus de la universidad a modo de prueba. Hoy en día se cuenta con un servicio web completamente funcional el cual fue realizado en PHP, y es el que se busca modernizar para que empate con las tecnologías que se enseñan en ITESO.

2. Desarrollo

2.1. Sustento teórico y metodológico

Backend: El back-end es el código que se ejecuta en el servidor, que recibe solicitudes de los clientes y contiene la lógica para enviar los datos apropiados al cliente. El *back-end* también incluye la base de datos, que almacenará de manera persistente todos los datos de la aplicación (codecademy, 2020).

Servidor: Es una computadora que escucha las solicitudes entrantes. Aunque existen máquinas diseñadas y optimizadas para este propósito en particular, cualquier computadora que esté conectada a una red puede actuar como un servidor (codecademy, 2020).

API: Interfaz de Programación de Aplicaciones que define un conjunto de directivas que pueden ser usadas para tener una pieza de software funcionando con algunas otras(Mozilla, 2019).

Rest API: Una interfaz entre sistemas que use HTTP para obtener datos o generar operaciones sobre esos datos en todos los formatos posibles (BBVA API_Market, 2016).

Node.js: Es un entorno que trabaja en tiempo de ejecución, de código abierto, multi-plataforma, que permite a los desarrolladores crear toda clase de herramientas de lado servidor y aplicaciones en JavaScript (Mozilla, 2019).

Express: Es una infraestructura de aplicaciones web Node.js mínima y flexible que proporciona un conjunto sólido de características para las aplicaciones web y móviles(Express, s.f.).

JavaScript: Es un lenguaje de programación ligero, interpretado, o compilado justo-a-tiempo. Es conocido como un lenguaje de scripting para páginas web, y es usado en entornos fuera del navegador, tal como Node.js, Apache CouchDB y Adobe Acrobat. JS es un lenguaje de programación basada en prototipos, multiparadigma, de un solo hilo, dinámico, con soporte para programación orientada a objetos, imperativa y funcional(Mozilla, 2020).

Programación Funcional: Puede considerarse lo opuesto a la programación orientada a objetos. Los objetos son cápsulas que contienen algún estado interno junto con una colección de llamadas a métodos que le permiten modificar este estado, y los programas

consisten en realizar el conjunto correcto de cambios de estado. La programación funcional quiere evitar los cambios de estado tanto como sea posible y trabaja con datos que fluyen entre funciones (Kuchling, 2020).

Servicio Web: Es un sistema de software designado para dar soporte a la interacción de máquina a máquina interoperativa a través de una red. Un servicio web realiza una tarea o un conjunto de tareas, y se describe mediante una descripción de servicio en una notación XML estándar llamada WSDL. La descripción de servicio proporciona todos los detalles necesarios para interactuar con el servicio, incluidos los formatos de mensaje, los protocolos de transporte y la ubicación (IBM, s.f.).

Base de datos: Es una colección organizada de información estructurada, o datos, típicamente almacenados electrónicamente en un sistema de computadora. Una base de datos es usualmente controlada por un sistema de gestión de base de datos (DBMS). En conjunto, los datos y el DBMS, en conjunto con otras aplicaciones se conocen como un sistema de base de datos, que a menudo se reducen a solo base de datos (Oracle, s.f.).

MySQL: Es un sistema de gestión de bases de datos SQL de código abierto más popular, está desarrollado, distribuido y respaldado por Oracle Corp (MySQL, s.f.).

Host: Es un servicio de TI, que proporciona acceso remoto a servidores físicos o virtuales fuera de las instalaciones y recursos asociados. El alojamiento de servidores permite a los equipos de TI aprovisionar y comenzar a utilizar servidores de datos y aplicaciones sin el costo inicial, las demoras y la mano de obra de comprar, configurar, administrar y mantener el hardware del servidor físico en el sitio. (IBM, 2020)

JWT: Es un estándar abierto que define un método compacto y autocontenido para encapsular y compartir información de manera segura entre distintas partes mediante el uso de objetos JSON. Se puede confiar y verificar el contenido del token cuando este está firmado digitalmente (JWS, RFC 7515). La firma se puede generar usando claves simétricas (HMAC) o claves asimétricas (RSA o ECDSA). Adicionalmente los JWT pueden contener también datos cifrados (JWE, RFC 7516) para proteger datos sensibles. (BBVA API, 2020)

Metodología ágil: Es un enfoque iterativo de la gestión de proyectos y el desarrollo de software que ayuda a los equipos a proporcionar valor a sus clientes más rápido y con

menos problemas. En lugar de centrarse en un lanzamiento de gran envergadura, un equipo ágil entrega el trabajo en incrementos pequeños, pero que se pueden consumir(Atlassian, s.f.).

***Swagger*:** Es una especificación abierta para definir documentación sobre las API REST. El documento *Swagger* especifica la lista de recursos que están disponibles en la API y las operaciones a las que se puede llamar en esos recursos. El documento también especifica la lista de parámetros para una operación, nombre y el tipo de parámetros y si son obligatorios u opcionales, e información sobre los valores aceptables. Además, puede incluir un esquema JSON que describe la estructura del cuerpo de la solicitud que se envía a una operación en una API REST, y el esquema JSON describe la estructura de los cuerpos de respuesta que se devuelven de una operación (IBM, s.f.).

HTTP: Es un protocolo de la capa de aplicación para la transmisión de documentos hipermedia, como HTML. Fue diseñado para la comunicación entre los navegadores y servidores web, aunque puede ser utilizado para otros propósitos. Sigue el modelo cliente-servidor, en el que un cliente establece una conexión, realizando una petición a un servidor y espera una respuesta de este (Mozilla, 2019).

JSON: Es una sintaxis para serializar objetos, arreglos, números, cadenas, booleanos y nulos. Está basado sobre sintaxis JavaScript (Mozilla, 2020).

Refactorización: Refactorizar es el proceso de modificar el código de un desarrollo para mejorar su estructura interna sin alterar la funcionalidad que ofrece el desarrollo externamente (Universidad Pais Vasco, s.f.).

MVC: Es un patrón de diseño que se utiliza para desacoplar la interfaz de usuario (vista), los datos (modelo) y la lógica de la aplicación (controlador). Este patrón ayuda a lograr la separación de preocupaciones (Microsoft, s.f.).

CRUD: Es un acrónimo de los cuatro tipos básicos de comandos SQL*: Create, Read, Update, Delete*. Una aplicación CRUD es aquella que usa formularios para obtener datos dentro y fuera de una base de datos (Jboss.org, s.f.).

PHP: Es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web. PHP está enfocado principalmente a la programación de scripts del lado del

servidor, por lo que se puede hacer cualquier cosa que pueda hacer otro programa CGI, como recopilar datos de formularios, generar páginas con contenidos dinámicos, o enviar y recibir cookies (PHP, s.f.).

Pruebas unitarias: Una prueba unitaria se utiliza para comprobar que un método concreto del código de producción funciona correctamente, probar las regresiones o realizar pruebas relacionadas. Es importante conocer claramente cuál es el objetivo del *test*. Cualquier desarrollador debería poder conocer claramente cuál es el objetivo de la prueba y su funcionamiento. Esto sólo se consigue si se trata el código de pruebas como el código de la aplicación (UA, s.f.).

JEST: Es un poderoso marco de pruebas de JavaScript basado en la simplicidad, funciona con proyectos de Babel, TypeScript, Node, React, Angular, Vue y más (JEST, s.f.).

Para llevar a cabo el proyecto se realizarán reuniones todos los lunes para revisar los avances de la semana, resolver dudas y bloqueos en el desarrollo, de la misma manera se tendrán presentaciones de los avances de la semana cada martes. Para realizar un mejor control y seguimiento del proyecto este se separó en 5 actividades principales: documentación, refactorización, controladores CRUD, seguridad y pruebas.

2.2. Planeación y seguimiento del proyecto

Descripción del proyecto

Actualizar y documentar el servicio backend de la red de sensores con la que cuenta el ITESO dentro del bosque de la primavera. Para realizar un mejor control del progreso el proyecto este se separó en 5 actividades principales:

1. **Documentación**: En la etapa de documentación se proponen las rutas basándose en las tablas de la base de datos, por cada tabla se generará un controlador y sus acciones. Después, generamos un documento JSON con toda la información de las rutas: ruta completa, método HTTP, y parámetros, para que la herramienta *Swagger* lo pueda procesar y generar un sitio de documentación.
2. **Refactorización**: En esta etapa se analiza código del servidor realizado en el periodo Primavera 2020 para convertir estas rutas al modelo propuesto mencionado en el punto

anterior y refactorizar el código para que este siga el patrón de diseño MVC junto con el diseño de los módulos de JavaScript.

3. **Controladores CRUD**: Para esta etapa terminamos de actualizar el viejo servicio realizado en PHP a la nueva versión en NodeJS al terminar de implementar todas las operaciones CRUD mediante los controladores.
4. **Seguridad**: En esta etapa se asignan las rutas utilizando JWTs como proveedores de identidad, autentificación, y permisos, autorización, así como el diseño de las tablas y funciones necesarias para realizar el manejo de los permisos.
5. **Pruebas**: En la etapa de pruebas se generan, como su nombre lo indica, las pruebas para asegurar que la implementación de los métodos sea correcta mediante la herramienta Jest

Plan de trabajo

El proyecto tiene previsto que se tengan las siguientes actividades:

Tabla 1. Plan de trabajo

Actividades	RH	Tipo de actividad	Fecha [Inicio]	Fecha [Entrega]	Semana
Realizar plan de trabajo	Carlos Soto Pérez, Héctor Chávez Morales	Operativo	01/09/2020	01/09/2020	3
Reporte	Carlos Soto Pérez, Héctor Chávez Morales	Operativo	03/11/2020	04/12/2020	3
Documentar el trabajo final	Carlos Soto Pérez, Héctor Chávez Morales	Operativo	03/11/2020	07/11/2020	3 a 15
Documentar controlador de Nodes	Héctor Chávez Morales	Operativo	02/09/2020	09/09/2020	3 a 4
Documentar controlador de LecturaNodes	Carlos Soto Pérez	Operativo	02/09/2020	09/09/2020	3 a 4
Documentar controlador de NodeValues	Héctor Chávez Morales	Operativo	02/09/2020	09/09/2020	3 a 4
Documentar controlador de Values	Carlos Soto Pérez	Operativo	02/09/2020	09/09/2020	3 a 4
Documentar controlador de Usuarios	Héctor Chávez Morales	Operativo	02/09/2020	09/09/2020	3 a 4

Reunión de revisión de progreso interno	Carlos Soto Pérez, Héctor Chávez Morales	Operativo	07/09/2020	07/09/2020	4
Reunión de revisión de progreso interno	Carlos Soto Pérez, Héctor Chávez Morales y el profesor	Operativo	08/09/2020	08/09/2020	4
Refactorización de LecturaNodos	Carlos Soto Pérez	Técnica	09/09/2020	16/09/2020	4 a 5
Refactorización de Nodes	Héctor Chávez Morales	Técnica	09/09/2020	16/09/2020	4 a 5
Reunión de revisión de progreso interno	Carlos Soto Pérez, Héctor Chávez Morales	Operativo	14/09/2020	14/09/2020	5
Reunión de revisión de progreso interno	Carlos Soto Pérez, Héctor Chávez Morales y el profesor	Operativo	15/09/2020	15/09/2020	5
Crear script modelo de Nodes	Héctor Chávez Morales	Técnico	16/09/2020	30/09/2020	5 a 7
Crear script modelo de LecturaNodes	Carlos Soto Pérez	Técnico	16/09/2020	30/09/2020	5 a 7
Crear Script modelo de NodeValues	Héctor Chávez Morales	Técnico	16/09/2020	30/09/2020	5 a 7
Crear script modelo de Values	Carlos Soto Pérez	Técnico	16/09/2020	30/09/2020	5 a 7
Crear Script modelo de Usuarios	Héctor Chávez Morales	Técnico	16/09/2020	30/09/2020	5 a 7
Reunión de revisión de progreso interno	Carlos Soto Pérez, Héctor Chávez Morales	Operativo	21/09/2020	21/09/2020	6
Reunión de revisión de progreso interno	Carlos Soto Pérez, Héctor Chávez Morales y el profesor	Operativo	22/09/2020	22/09/2020	6
Reunión de revisión de progreso interno	Carlos Soto Pérez, Héctor Chávez Morales	Operativo	28/09/2020	28/09/2020	7
Reunión de revisión de progreso interno	Carlos Soto Pérez, Héctor Chávez Morales y el profesor	Operativo	29/09/2020	29/09/2020	7
Crear controlador de Nodes	Héctor Chávez Morales	Técnico	30/09/2020	07/10/2020	7 a 8
Crear controlador de LecturaNodes	Carlos Soto Pérez	Técnico	30/09/2020	07/10/2020	7 a 8

Crear controlador de NodeValues	Héctor Antonio Morales	Técnico	30/09/2020	07/10/2020	7 a 8
Crear controlador de Values	Carlos Soto Pérez	Técnico	30/09/2020	07/10/2020	7 a 8
Crear controlador de autentificación	Carlos Soto Pérez	Técnico	30/09/2020	07/10/2020	7 a 8
Crear controlador de Usuarios	Héctor Chávez Morales	Técnico	30/09/2020	07/10/2020	7 a 8
Reunión de revisión de progreso interno	Carlos Soto Pérez, Héctor Chávez Morales	Operativo	05/10/2020	05/10/2020	8
Reunión de revisión de progreso interno	Carlos Soto Pérez, Héctor Chávez Morales y el profesor	Operativo	06/10/2020	06/10/2020	8
Crear las rutas del servidor	Carlos Soto Pérez, Héctor Chávez Morales	Técnico	30/09/2020	07/10/2020	8 a 9
Reunión de revisión de progreso	Carlos Soto Pérez, Héctor Chávez Morales	Operativo	05/10/2020	05/10/2020	9
Reunión de revisión de progreso interno	Carlos Soto Pérez, Héctor Chávez Morales y el profesor	Operativo	06/10/2020	06/10/2020	9
Reunión de revisión de progreso interno	Carlos Soto Pérez, Héctor Chávez Morales	Operativo	12/10/2020	12/10/2020	10
Reunión de revisión de progreso interno	Carlos Soto Pérez, Héctor Chávez Morales y el profesor	Operativo	13/10/2020	13/10/2020	10
Reunión de revisión de progreso interno	Carlos Soto Pérez, Héctor Chávez Morales	Operativo	19/10/2020	19/10/2020	11
Reunión de revisión de progreso interno	Carlos Soto Pérez, Héctor Chávez Morales y el profesor	Operativo	20/10/2020	20/10/2020	11
Reunión de revisión de progreso interno	Carlos Soto Pérez, Héctor Chávez Morales	Operativo	26/10/2020	26/10/2020	12
Reunión de revisión de progreso interno	Carlos Soto Pérez, Héctor Chávez Morales y el profesor	Operativo	27/10/2020	27/10/2020	12

Reunión de revisión de progreso interno	Carlos Soto Pérez, Héctor Chávez Morales	Operativo	01/11/2020	02/11/2020	13
Reunión de revisión de progreso interno	Carlos Soto Pérez, Héctor Chávez Morales y el profesor	Operativo	03/10/2020	03/10/2020	13
Controlador Actualizaciones	Isaac Cabrera	Desarrollo	06/10/2020	13/10/2020	9 a 10
Controlador Comunicaciones	Carlos Soto Pérez	Desarrollo	06/10/2020	13/10/2020	9 a 10
Controlador Datos Gobierno	Alessandro Pallaro	Desarrollo	06/10/2020	13/10/2020	9 a 10
Controlador Recomendaciones	Isaac Cabrera	Desarrollo	13/10/2020	20/10/2020	10 a 11
Controlador Authetificación	Carlos Soto Pérez	Desarrollo	13/10/2020	20/10/2020	10 a 11
Controlador Mapa	Alessandro Pallaro	Desarrollo	13/10/2020	20/10/2020	10 a 11
Crear instanciador de JWTs	Carlos Soto Pérez	Técnico	07/10/2020	03/11/2020	8 a 13
Crear middlewares para verificar autentificación	Héctor Chávez Morales	Técnico	07/10/2020	03/11/2020	8 a 13
Pruebas unitarias	Carlos Soto Pérez, Héctor Chávez Morales	Técnico	09/09/2020	30/11/2020	7 a 13
Pruebas de integración	Carlos Soto Pérez, Héctor Chávez Morales	Técnico	07/10/2020	30/11/2020	7 a 13

Tabla 2. Registro de reuniones

Fecha	Participantes	Acuerdos o acciones
01/09/2020	Carlos Soto Pérez, Héctor António Chávez Morales y Profesor	Presentación de avances semana 3.
01/09/2020	Carlos Soto Pérez y Héctor António Chávez Morales	Reunión para el diseño de los módulos. En esta reunión decidimos utilizar el diseño más utilizado para una aplicación con *Express*, pero con la variante de usar solo programación funcional.

08/09/2020	Carlos Soto Pérez, Héctor Antônio Chávez Morales y Profesor	Presentación de avances semana 4.
08/09/2020	Carlos Soto Pérez, Isaac Cabrera Cortez y Héctor Antônio Chávez Morales	Reunión para la documentación del API de COVID. Dada la mala calidad de la documentación del *backend* para la aplicación del covid, el equipo que está trabajando en la aplicación móvil nos pidió transportar la documentación actual a *Swagger*, así como la mala implementación en código del *backend*. Se llegó al acuerdo de documentar el *backend* de COVID y una vez terminado el *backend* de la red de sensores se iba a refactorizar el *backend* de COVID.
08/09/2020	Carlos Soto Pérez, Gerardo de Jesús Cruz Plazola y Héctor Antônio Chávez Morales	Reunión para actualizar el rumbo del trabajo. El equipo de frontend nos pidió si podíamos tener algunos *endpoints* listos para que ellos puedan probar sus códigos.
15/09/2020	Carlos Soto Pérez, Héctor Antônio Chávez Morales y Profesor	Presentación de avances semana 5.
17/09/2020	Carlos Soto Pérez y Héctor Antônio Chávez Morales	Reunión para la creación del entorno de pruebas/desarrollo. Tras darnos cuenta de que no podíamos conectarnos a la base de datos del servidor y que no nos podíamos quitar de ese bloqueo, creamos una base de datos con un

		proveedor de plataforma como servicio y así poder probar antes de subir al servidor.
22/09/2020	Carlos Soto Pérez, Héctor Antônio Chávez Morales y Profesor	Presentación de avances semana 6.
24/09/2020	Carlos Soto Pérez y Profesor	Revisión, comentarios y acciones para la correcta implementación de lo que se tiene hasta este momento en el backend de la red de sensores, se detectó un error de formato en la inserción de l información de las lecturas.
28/09/2020	Carlos Soto Pérez y Héctor Antônio Chávez Morales	Reunión para revisar el progreso actual y definir el curso para las pruebas unitarias y e2e. Se definió hacer una introducción rápida a Jest para los miembros del equipo
29/09/2020	Carlos Soto Pérez, Héctor Antônio Chávez Morales y Profesor	Presentación de avances semana 7.
29/09/2020	Carlos Soto Pérez, Isaac Cabrera, Gerardo de Jesús Cruz Plazola y Héctor Antônio Chávez Morales	Revisión de plan de acción para la refactorización del backend de la aplicación de COVID. Se llegó al acuerdo de dividir el equipo de *backend*, Héctor se quedará trabajando en el backend del sistema de monitoreo, Carlos, Isaac y Alessandro se pondrán a refactorizar el *backend* de COVID
02/10/2020	Carlos Soto Pérez, Isaac Cabrera y Alessandro Pallaro	Comunicación de los estándares que Héctor y Carlos seguimos para el backend de los nodos y así poder integrarlos en el *backend* del COVID

05/10/2020	Carlos Soto Pérez, Isaac Cabrera y Alessandro Pallaro	Definición de elementos de trabajo y repartición de tareas, se hizo una tabla de Excel donde se menciona cada controlador, sus acciones y el responsable de realizarlo
05/10/2020	Carlos Soto Pérez y Héctor António Chávez Morales	Definición de estándares, cobertura y pruebas para el backend de los nodos. Se llegó al acuerdo de utilizar Jest y la Liberia *supertest* para probar el API desde los archivos de prueba.
06/10/2020	Carlos Soto Pérez, Héctor António Chávez Morales y Profesor	Presentación de avances semana 8.
13/10/2020	Carlos Soto Pérez, Héctor António Chávez Morales y Profesor	Presentación de avances semana 9.
19/10/2020	Carlos Soto Pérez, Héctor António Chávez Morales	Revisión de pruebas unitarias para Nodos y covid y ´problema con librería Bcrypt.
20/10/2020	Carlos Soto Pérez, Héctor António Chávez Morales y Profesor	Presentación de avances semana 10. En esta reunión se expusieron los problemas de la semana para darles seguimiento
20/10/2020	Carlos Soto Pérez, Isaac Cabrera, Gerardo de Jesús Cruz Plazola y Héctor António Chávez Morales	Esclarecimiento de problema en el endpoint. El equipo de backend no notificó a frontend del cambio en un endpoint para la autentificación.

20/10/2020	Carlos Soto Pérez, Héctor Antônio Chávez Morales, Isaac Cabrera y Alessandro Pallaro	Discusión sobre el método de autentificación para el backend de covid.
23/10/2020	Carlos Soto Pérez, Héctor Antônio Chávez Morales y Profesor	Trabajo en reporte técnico. Se distribuyeron las tareas y temas del reporte técnico. 1. Documentación del código ¿JSDoc? Exportar en MD 2. Diseño de código a. Que va en cada paquete 3. Validaciones de esquema *4. Swagger* 5. executionContext 6. Pruebas 7. CI 8. Phusion Passenger 9. guía de contribución y código de conducta
27/10/2020	Carlos Soto Pérez, Héctor Antônio Chávez Morales y Profesor	Presentación de avances semana 11. En esta reunión se expusieron las soluciones de la semana pasada, semana 10
03/11/2020	Carlos Soto Pérez, Héctor Antônio Chávez Morales y Profesor	Presentación de avances semana 12.
03/11/2020	Carlos Soto Pérez, Héctor Antônio Chávez Morales	Definición de pruebas para el *backend* de covid
10/11/2020	Carlos Soto Pérez, Héctor Antônio	Presentación de avances semana 13.

	Chávez Morales y Profesor	
17/11/2020	Carlos Soto Pérez, Héctor António Chávez Morales y Profesor	Presentación de avances semana 14.
18/11/2020	Carlos Soto Pérez, Isaac Cabrera, Gerardo de Jesús Cruz Plazola y Héctor António Chávez Morales	Esclarecimiento de dudas y resolución de problemas de autentificación en las acciones de los controladores que requieren que la solicitud contenga identidad.
22/11/2020	Carlos Soto Pérez, Isaac Cabrera, Gerardo de Jesús Cruz Plazola y Héctor António Chávez Morales	Soporte en las rutas que regresaban un error 500.
24/11/2020	Carlos Soto Pérez, Héctor António Chávez Morales y Profesor	Presentación de avances semana 15

Desarrollo de propuesta de mejora

Semana 3

En esta semana se trabajó sobre la documentación y diseño de los *endpoints* de cada controlador, para hacer eso se utilizó la herramienta *swagger* para generar una página de

documentación. Para tener una línea de diseño utilizamos los estándares de una REST API, el cual se tienen controladores y acciones.

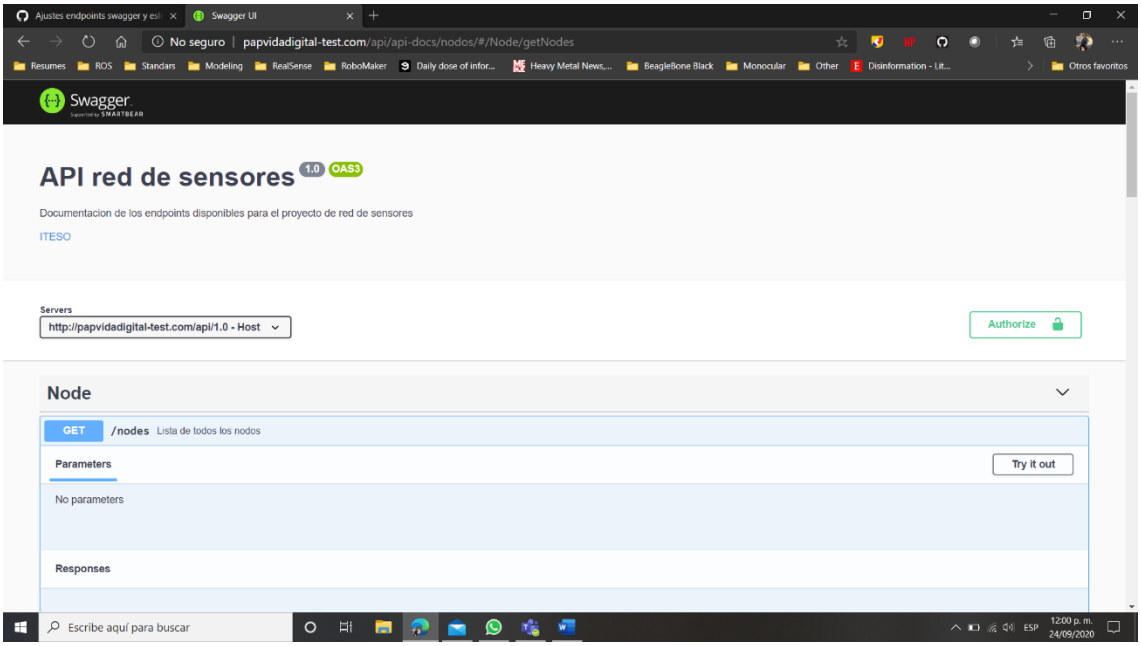


Ilustración 1. Página de documentación del API de nodos

Semana 4

En esta semana, después de tener la documentación, se realizó el diseño de los módulos de la aplicación, así como la refactorización de la versión anterior e incompleta de la aplicación en NodeJS. Seguimos el diseño base de una aplicación NodeJS con el *framework Express*.

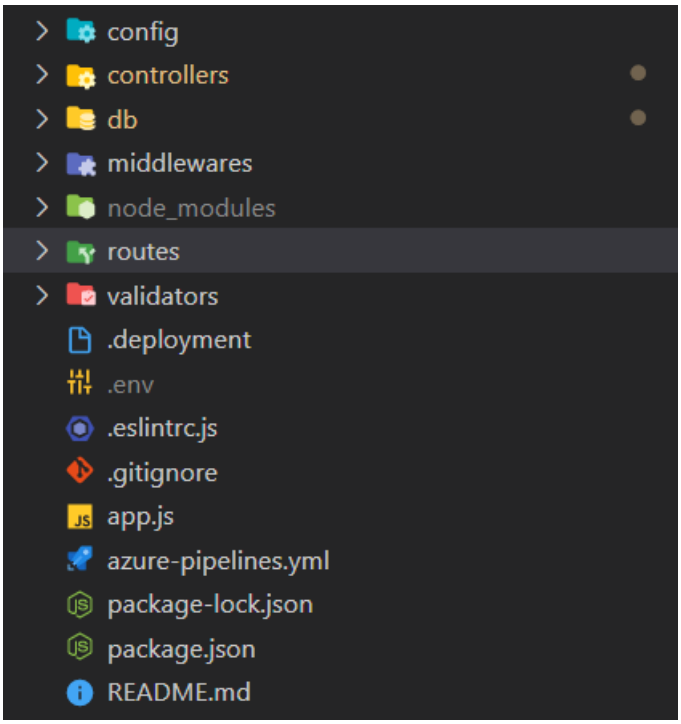


Ilustración 2. Diseño de módulos de la aplicación de nodos

Semana 5

Para la semana 5 realizamos bajo petición del equipo de COVID la documentación del API con *Swagger*.

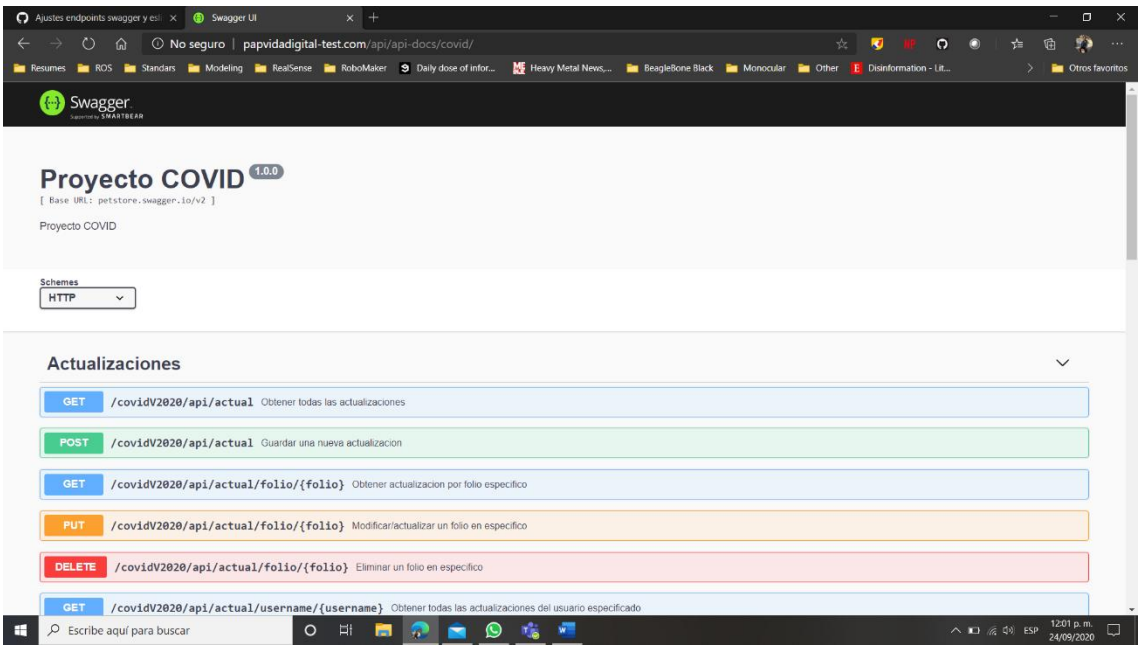


Ilustración 3. Documentación v1 de la API de COVID

Semana 6

Se subió la aplicación hasta este entonces al servidor, dejamos de lado las documentaciones para empezar a programar para que nuestros compañeros encargados del *frontend* puedan empezar lo más pronto posible su desarrollo.

Endpoints listos hasta este entonces

- Documentación /api/api-docs/nodes
- Lecturas
    - Insertar
    - Leer por día, semana, mes o año
- Nodos
    - Crear nodo
    - Obtener nodo por ID
    - Obtener todos los nodos
    - Actualizar nodo
    - Eliminar nodo
- Usuarios
    - Crear usuario
    - Obtener usuario por ID
    - Obtener todos los usuarios
    - Actualizar usuario
    - Eliminar usuario
- Valores
    - Obtener sensores
    - Obtener sensores en nodo
    - Obtener descripción de un sensor
    - Eliminar sensor de un nodo
    - Actualizar sensor de un nodo
- Variables
    - Obtener todas las variables

Semana 7

Para el desarrollo de esta semana se añadieron validaciones por esquemas a los datos ingresados a la aplicación, se tomaron las definiciones de los esquemas desde el documento del *Swagger*

- Documentación /api/api-docs/nodes
- Lecturas
    - Insertar
    - Leer por día, semana, mes o año
    - Obtener por nodo
    - Obtener n-ultimas
- Nodos
    - Crear nodo
    - Obtener nodo por ID
    - Obtener todos los nodos
    - Actualizar nodo
    - Eliminar nodo
- Usuarios
    - Crear usuario
    - Obtener usuario por ID
    - Obtener todos los usuarios
    - Actualizar usuario
    - Eliminar usuario
- Valores
    - Obtener sensores
    - Obtener sensores en nodo
    - Obtener descripción de un sensor
    - Eliminar sensor de un nodo
    - Actualizar sensor de un nodo
- Variables
    - Obtener todas las variables
    - Actualizar variables
    - Eliminar variable
    - Crear variable

Semana 8

- Nuevo plan de trabajo
- Trabajo de definición de estándares para las pruebas

- Mejoras en endpoints ya entregados.
- Inicio de desarrollo para el backend de covid, creación de repositorio y código base para empezar todos

Semana 9

- Ajuste en el *pipeline* que corre las pruebas en el repositorio para asegurar que no pase código con errores.

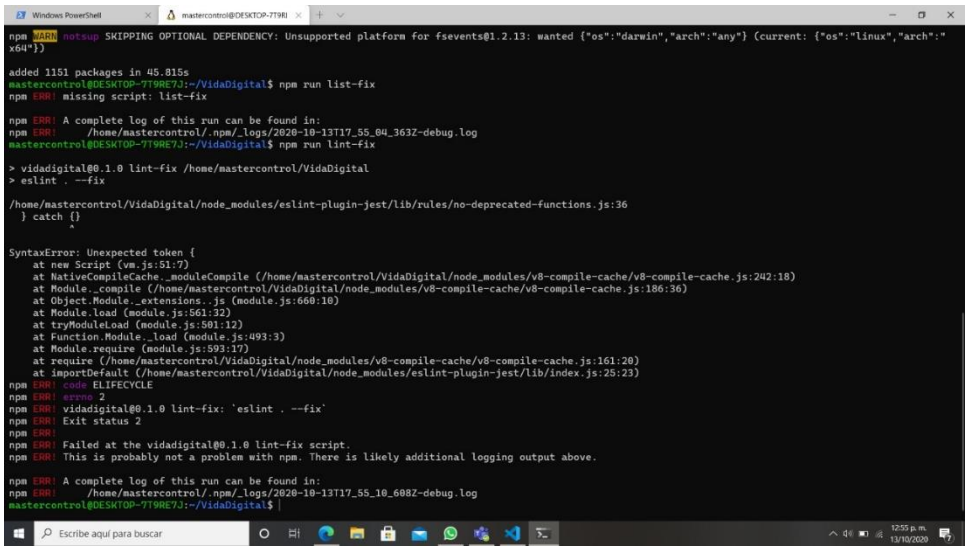


Ilustración 4. Salida de conola del error de la versión de la libería Jest

Este error se dio por la librería que se encarga de analizar el estilo del código, la solución fue bajar la versión una anterior

- Controlador de COVID: comunicaciones, usuarios y pruebas

Semana 10

Esta semana estuvo marcada por las fallas misteriosas en el servidor, se llegó a una aproximación del problema utilizando el patrón de diseño *adapter* y *context* para ejecutar las consultar a la base de datos y evitar que se acaben las conexiones disponibles.

Esto siguió presentando problemas, después de una revisión de los logs se encontró con el siguiente mensaje: `Checking whether to disconnect long-running connections for process 21039, application /home/papvida1/O2020 (production)` Realizando una investigación se llegó a la posible raíz del problema, el servidor de NodeJS se está ejecutando sobre el servidor de aplicaciones, al detectar que no se está en uso pero se tiene un consumo de memoria, este apaga e servidor sin dar aviso alguno. La posible solución será buscar que cosa está consumiendo la memoria para arreglarlo o modificar ese comportamiento en Passenger.

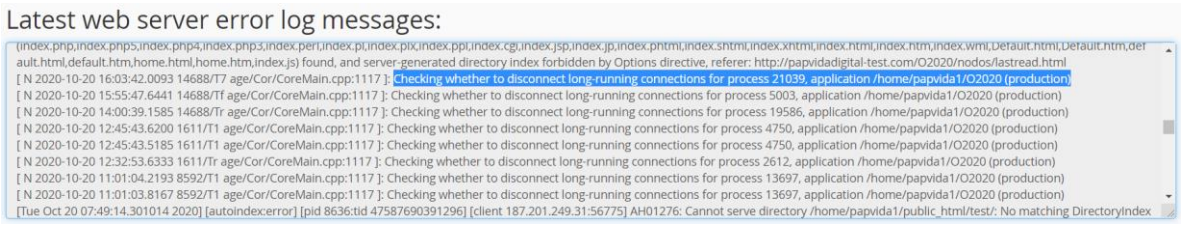


Ilustración 5. Registros de Phusion Passenger

Esta semana también se incrementó la cobertura de código a un 74% de líneas ejecutadas, se generó el endpoint de autentificación. Para esto último se presentó un error en la versión de la librería bcrypt que se utiliza para hashear las contraseñas.

Por el lado de la aplicación de COVID se implementó el contexto de ejecución y se terminaron los controladores de usuario, datosGov, Comunicaciones y pruebas.

Semana 11

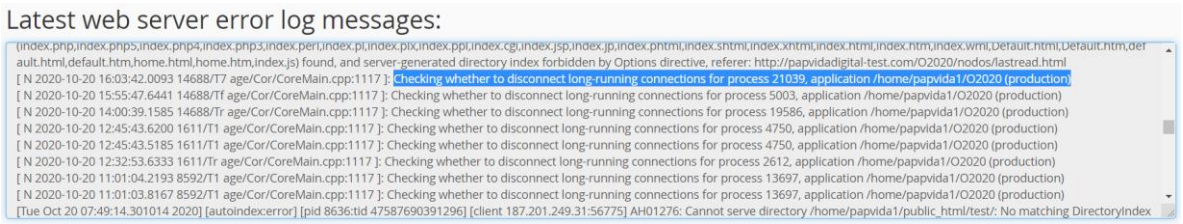


Ilustración 6. Registros de Phusion Passenger

Esa salida no indica ningún problema, como indica la documentación de *Phusion Passenger*, este es un comportamiento esperado después que la aplicación pasa una cantidad predeterminada de segundos sin actividad, este termina la aplicación para que no esté utilizando recursos, cuando se vuelve a necesitar Passenger lo vuelve a levantar.

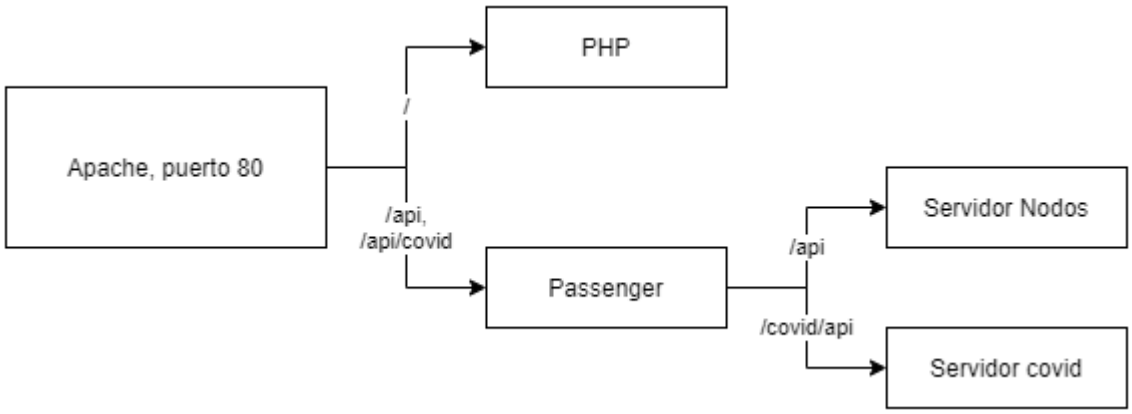


Ilustración 7. Diagrama de relación de los diferentes servicios del CPanel

Para lograr que la aplicación se levante de manera automática no se encontró una metodología de despliegue adecuada, la única observación que se tiene es, al momento de subir código nuevo, se requiere detener e iniciar la aplicación en lugar de solo reiniciar. Esta diferencia puede ser un problema de CPanel y no de *Passenger*.
Para verificar el correcto funcionamiento se creó un CORN Job que manda una solicitud cada hora, una vez pasando los 2 días se empezó con el nodo HM1.

Semana 12

En esta semana se completaron las pruebas al código. Logrando una cobertura en todos los endpoint con un total de 50 pruebas ejecutadas, un 82% de funciones probadas y un 79% de líneas probadas. Las pruebas fueron ejecutadas en la plataforma Azure DevOps como parte del proceso de integración continua.

Se trabajó y terminó la documentación técnica, Anexo B.

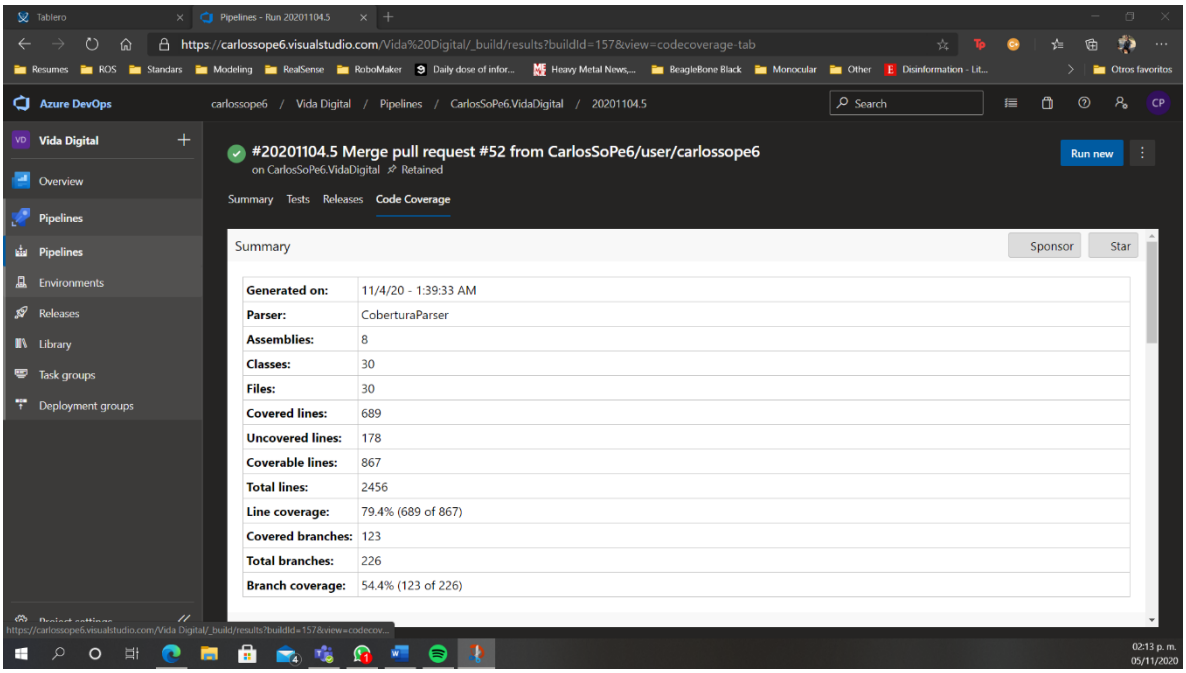


Ilustración 8. Reporte de cobertura en Azure Pipelines

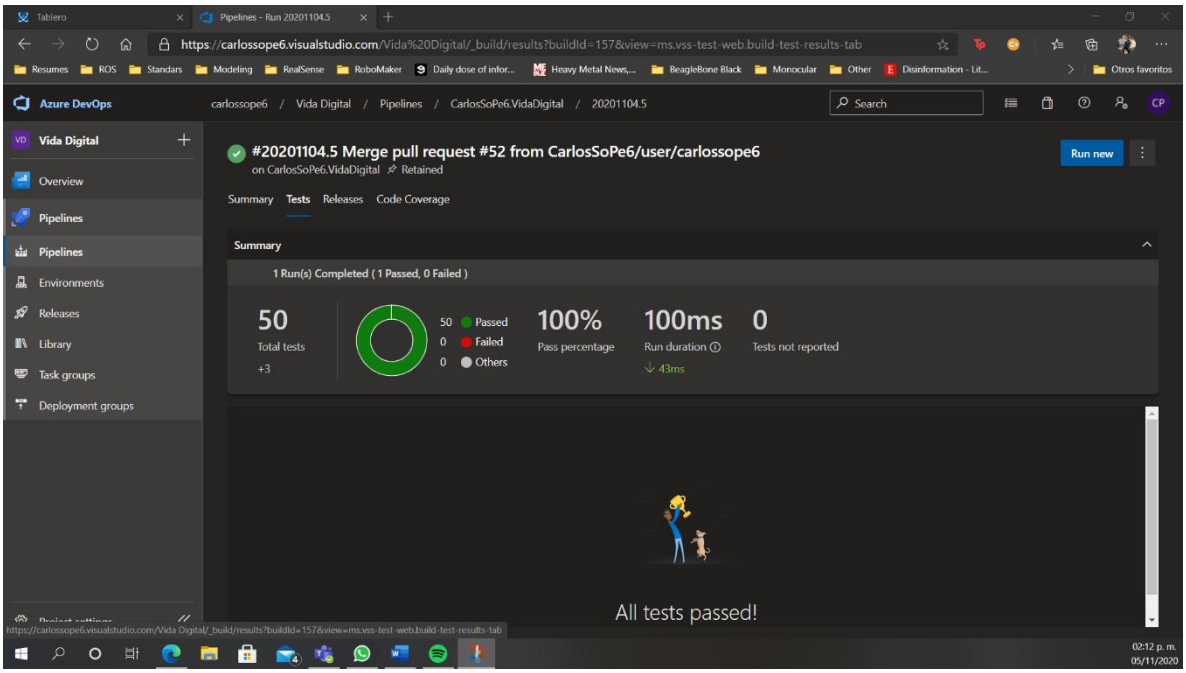


Ilustración 9. Reporte de pruebas Azure Pipeline

Semana 13

Se tienen completados los endpoints de

- Usuarios
- Recomendaciones
- Autentificación

Falta

- Pruebas
- DatosGov
- Mapa

Semana 14

Se tienen completados los endpoints de

- Usuarios
- Recomendaciones
- Autentificación
- Pruebas
- DatosGov

Falta

- Mapa

Se trabajó en el reporte de las pruebas, se analizaron diferentes estándares para documentar, no se llegó a una buena solución y se va a discutir en la presentación del reporte PAP. Además, se empezó a trabajar junto con el equipo de diseño y *front* el documento de los escenarios.

Semana 15

Se trabajó en actualizar el *backend* de covid para hacer una diferencia entre los posibles casos y los casos que ya se pudieron haber recuperado en la DB. Para esto se actualizó la tabla de Actualizaciones añadiéndole una columna booleana con el nombre “activo”. Esta se usará para filtrar de forma eficiente los registros válidos o inválidos, ya sea por fecha o porque el usuario realizó otra prueba. Además, se generó un procedimiento almacenado para obtener los casos acumulados y una rutina que cada día a las 0h actualiza los caos que probablemente estén recuperados.

3. Resultados del trabajo profesional

1. *Backend* de la aplicación de red de sensores. El código de JavaScript funcional y probado. El desarrollo de la aplicación en NodeJS, dividida en diferentes módulos, el cual cuenta con diferentes maneras de documentación, en donde principalmente se dividen las siguientes:
    a. Autentificación
    b. Lecturas
    c. Usuario
    d. Valores
    e. Variables
2. Documentación técnica. Documento de Word y Mark Down en el repositorio.
3. Documentación API en *Swagger* y el JSON de la descripción que utiliza *Swagger* para generarla. Además, se generó un endpoint especial el cual puede ser utilizado como referencia para consultar los demás endpoints disponibles en la aplicación, con esto se puede ver los tipos de respuesta y valores esperados para el correcto funcionamiento.
4. Documentación de los endpoints en Postman.
5. Documentación de las pruebas utilizando Jest probamos el funcionamiento del código, en donde verificamos que las respuestas sean las esperadas, este además genera un reporte visual en donde podemos observar que ha sido probado y que no Anexo C.

4. Reflexiones del alumno o alumnos sobre sus aprendizajes, las implicaciones éticas y los aportes sociales del proyecto

Aprendizajes profesionales

Algunas de las competencias desarrolladas y fomentadas a lo largo del semestre fueron, el trabajo en equipo, la responsabilidad y la apertura a la aceptación y evaluación de nuevas ideas y conceptos propuestos por compañeros.

Se pudo poner a prueba los conocimientos en una problemática real, la cual se aportaron soluciones a un problema tan grave como lo es el cuidado del medio ambiente en una zona de gran importancia como lo es el bosque de la primavera.

Poder trabajar con diferentes equipos y poder arreglar los inconvenientes conforme iban apareciendo fue algo que aporto a mi experiencia personal.

Carlos Soto Pérez

Como se mencionó anteriormente, el aprendizaje más importante es, y siempre será, el trabajar en equipo. Con este proyecto pude ver bien en claro la diferencia entre un código mal diseñado a diferencia de uno donde si se le dedicó su tiempo a la parte de diseño, esto se pudo ver con las fases de refactorización y el diseño del ExecutionContext. Estas “buenas prácticas” expuestas en este párrafo considero que fue un factor que nos ayudó a completar el desarrollo 1 semana antes de lo planeado.

Héctor Antonio Chávez Morales

Un aprendizaje que valoro fue el poder haber empezado el ciclo de desarrollo de esta aplicación y tener la libertad de gestionar el desarrollo del proyecto, además de las herramientas que serían utilizadas. Fue gratificante haber tenido la oportunidad de definir las prácticas de uso del código según nuestros conocimientos.

Otro aprendizaje valioso fue el haber podido trabajar en equipos, tanto con mi compañero en el desarrollo de la aplicación como con los otros equipos. La comunicación fue esencial para que todos pudiéramos completar nuestros requerimientos

Aprendizajes sociales

El impacto que se puede lograr con el uso de la aplicación de COVID es el proveer una fuente de información confiable y actualizada a la población en México. En el caso del Covid se provee

información relevante sobre la pandemia, ya sea números telefónicos de emergencia (línea Covid), número de contagios, etc. los beneficiados más importantes son población mexicana al informada sobre la situación de la pandemia, así como comercios informales y pacientes con enfermedades crónicas.

Con el sistema de la red de sensores se proporciona un sistema que facilita el monitoreo del bosque de la primavera, con el fin de prevenir posibles desastres en la zona y obtener información relevante del bosque para la generación de históricos que puedan ser utilidad para Anillo Primavera A.C. Con este sistema se benefician principalmente las personas que viven en cerca de las áreas que están ciento monitoreadas, así como personas interesadas en obtener datos de mediciones ambientales del bosque de la primavera.

La identificación de la problemática del COVID y de la necesidad de monitorear el bosque de la primavera surgieron de la observación, tanto al ver que negocios están cerrados o que la mayoría de las personas portan un cubrebocas para evitar contagios, o al experimentar como nos han tenido que evacuar de la universidad por contingencia ambiental causada por un incendio en el bosque.

Aprendizajes éticos

Dentro del equipo, se tomó una decisión que nos ayudó mucho al momento de desarrollar el backend de la red de sensores la cual fue la refactorización, esa decisión nos dejó muy en claro la necesidad de seguir los patrones de diseño para ayudar a los demás desarrolladores a tener un lenguaje común y así hacer más ágil el proceso de desarrollo.

Carlos Soto Pérez

Es por bien de todos seguir las buenas prácticas, y facilitarle la vida a todos los involucrados en el desarrollo, así como llevar una comunicación respetuosa entre todos los integrantes del equipo para dar una apertura a el intercambio de ideas y criticas constrictivas. Estos pensamientos están reflejados en el código de conducta: “Los desarrolladores son responsables de la calidad y estabilidad del código”.

Héctor Antonio Chávez Morales

Las principales decisiones en el desarrollo fueron tomadas en equipo, se hablaba y discutíamos para poder encontrar la solución que nos pareciera la más completa. En donde la comunicación jugo un papel esencial.

Siempre tener en cuenta las opiniones de los compañeros ya que pueden ser muy valiosas para el desarrollo de cualquier tipo de proyecto que se esté trabajando.

Aprendizajes en lo personal

Carlos Soto Pérez

Esta experiencia me dio para conocer que aún falta mucho camino para aprender el manejo de equipos o la distribución de tareas y que eso es algo que se aprende con la experiencia. Por otro lado, dadas las características del proyecto solo fue posible convivir con personas del DESI, la interacción con nuestros compañeros de la carrera de diseño se vio reducida.

Dado el trabajo de refactorización que realizamos me lleva a cuestionar sobre el compromiso de los compañeros que diseñaron el primer código y a los que le seguirán dando mantenimiento a este código, considero que un buen ingeniero en software debería de tomar un tiempo para pensar si su solución podrá ser mantenible en un futuro.

Héctor Antonio Chávez Morales

Una experiencia que me dejo este proyecto es que el trabajo en equipo es esencial para el desarrollo de cualquier proyecto. A lo largo de todo el proyecto trabajamos en equipo para definir desde el alcance hasta las implementaciones, entonces este proyecto me ayudo para poder saber más detalladamente como es mi forma de trabajar en equipo y mantener en cuenta siempre a las demás personas con las que estoy involucrado.

Otra experiencia importante que me dejo este proyecto fue el haber diseñado y desarrollado prácticamente el proyecto desde un inicio, fue una experiencia que no había vivido en otros proyectos “laborales”, ya que se debió tener en cuenta que cualquier persona que necesite o desee realizar cambios a la aplicación se le haga lo más fácil posible.

5. Conclusiones

Carlos Soto Pérez. Creo que de las cosas que se pueden mejorar en el PAP es hacer un *code review* entre los participantes del PAP, con esto se debería de mejorar la calidad del código y así se pueden evitar las refactorizaciones. Además, algo que se debe de hacer es documentar el código mientras se está escribiendo, eso es algo que el profesor nos pide hacer con el reporte PAP y es bueno para no perder detalles al momento de documentar, el programador que está realizando ese código es la primera audiencia, con esa documentación se puede tener una mejor idea del código que se está haciendo, y como parte de la documentación, mantener, al menos para backend, la documentación del *Swagger* actualizada mientras se hacen los cambios al código.

En cuanto lo que nos faltó terminar

1. la parte del buzón de mensajes a los nodos, para ello es necesario levantar los requerimientos necesarios y diseñar la interacción entre todos los componentes del sistema.
2. Mejorar la respuesta en los errores 400 para informar mejor cual fue el error en la petición.
3. Verificar todas las entradas donde no se esté usando la validación por esquema para prevenir un error 500 inesperado.
4. Mejorar el manejo de sesiones por usuario, la solución actual no se sigue todo el ciclo de vida de una autentificación con OAuth y JWT, falta mandar el token de actualización de sesión y hacer más corta la vida de los tokens de sesión actuales, de 1h a 15m.
5. Realizar la separación de los *logs* en diferentes archivos, el *Phusion Passenger* nos puede echar a perder un poco este esfuerzo puesto que se tiene un log general de todo.
6. Explorar opciones para alojar el sitio, investigar si es posible en los recursos de la OSI o una exploración costo/beneficio a usar un proveedor diferente al actual.

7. Mejorar las consultas a la base de datos para el Covid, analizar donde se necesitan índices para mejorar las consultas y pasar las consultas a procedimientos almacenados en la db para mejor rendimiento.

Adicional a las conclusiones y a manera de retrospectiva, algo que hicimos mal es hacer las pruebas al código sin una guía, las pruebas se deben de hacer contra los requerimientos del proyecto.

Héctor Chávez. Habiendo trabajado, y teniendo en cuenta el trabajo heredado en este proyecto encuentro algunos elementos a mejorar y algunos otros que pueden ser añadidos para añadir funcionabilidad extra.

1. Implementación del tema de sesiones dentro del sitio web, ya que actualmente no existe dicho concepto al ser solamente un inicio de sesión sin vencimiento, añadir esta funcionabilidad brindaría de un poco más de seguridad a la aplicación al no permitir sesiones antiguas.
2. *Code review* entre compañeros, ya que se debe de mantener un nivel del código aceptable y con esta técnica se podrá asegurar que si se cumpla.
3. Elementos de Buzon y *Token message* ya que no existe ninguna implementación por falta de los requerimientos, pero completando dichos elementos podría brindar de nueva funcionabilidad al proyecto y hacerlo más completo.
4. Mejorar los estados de respuesta en *request* fallidas, ya que actualmente no se tiene el mejor manejo de mensajes de error sobre estas al ser muy genéricas.
5. Mantener una documentación completa, útil y actualizada para que todos los compañeros puedan consultarla en caso de dudas.

6. Bibliografía

Atlassian. (s.f.). *Atlassian agile coach*. Obtenido de https://www.atlassian.com/es/agile

BBVA API. (17 de junio de 2020). *BBVA*. Obtenido de https://www.bbva.com/es/json-web-tokens-jwt-claves-para-usarlos-de-manera-segura/

BBVA API_Market. (15 de marzo de 2016). *BBVA API_Market*. Obtenido de https://bbvaopen4u.com/es/actualidad/api-rest-que-es-y-cuales-son-sus-ventajas-en-el-desarrollo-de-proyectos

codecademy. (3 de agosto de 2020). *codecademy*. Obtenido de https://www.codecademy.com/articles/back-end-architecture

Express. (s.f.). *Expressjs*. Obtenido de https://expressjs.com/

IBM. (s.f.). Obtenido de https://www.ibm.com/cloud/learn/server-hosting

IBM. (25 de abril de 2014). *IBM Support*. Obtenido de https://www.ibm.com/support/knowledgecenter/es/SSMKHH_9.0.0/com.ibm.etools.mft.doc/ac55710_.htm

IBM. (14 de julio de 2020). Obtenido de https://www.ibm.com/support/knowledgecenter/SSMKHH_10.0.0/com.ibm.etools.mft.doc/bi12018_.htm

IBM. (s.f.). *IBM Developers*. Obtenido de https://www.ibm.com/developerworks/opensource/library/os-ecref/

Jboss.org. (s.f.). Obtenido de https://docs.jboss.org/tools/4.0.1.Final/en/seam/html/crud_database_application.html

JEST. (s.f.). *JESTjs*. Obtenido de https://jestjs.io/

Kuchling, A. M. (24 de septiembre de 2020). *Pyrhon docs*. Obtenido de https://docs.python.org/3/howto/functional.html

Microsoft. (s.f.). *Microsoft dotnet*. Obtenido de https://dotnet.microsoft.com/apps/aspnet/mvc

Mozilla. (23 de marzo de 2019). *MDN web docs*. Obtenido de https://developer.mozilla.org/es/docs/Glossary/API

Mozilla. (18 de marzo de 2019). *Mozilla web docs*. Obtenido de https://developer.mozilla.org/es/docs/Web/HTTP

Mozilla. (10 de junio de 2020). *MDN web docs*. Obtenido de https://developer.mozilla.org/es/docs/Learn/Server-side/Express_Nodejs/Introduction

Mozilla. (11 de agosto de 2020). *MDN web docs*. Obtenido de https://developer.mozilla.org/es/docs/Web/JavaScript

Mozilla. (19 de septiembre de 2020). *Mozilla web docs*. Obtenido de https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Objetos_globales/JSON

MySQL. (s.f.). *MySQL dev*. Obtenido de https://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html

Oracle. (s.f.). *Oracle*. Obtenido de https://www.oracle.com/mx/database/what-is-database.html

PHP. (s.f.). Obtenido de https://www.php.net/manual/es/intro-whatis.php

UA. (s.f.). Obtenido de https://si.ua.es/es/documentacion/c-sharp/documentos/pruebas/07pruebasunitarias.pdf
Universidad Pais Vasco. (s.f.). Obtenido de https://ocw.ehu.eus/pluginfile.php/16698/mod_resource/content/2/lab2-refactor.pdf

7. Anexos

Anexo A. Primera versión de plan de trabajo

Actividades	RH	Tipo de actividad	Fecha [Inicio]	Fecha [Entrega]	Semana
Realizar plan de trabajo	Carlos Soto Pérez, Héctor Chávez Morales	Operativo	01/09/2020	01/09/2020	3
Reporte	Carlos Soto Pérez, Héctor Chávez Morales	Operativo	03/11/2020	04/12/2020	3
Documentar el trabajo final	Carlos Soto Pérez, Héctor Chávez Morales	Operativo	03/11/2020	07/11/2020	3 a 15
Documentar controlador de Nodes	Héctor Chávez Morales	Operativo	02/09/2020	09/09/2020	3 a 4
Documentar controlador de LecturaNodes	Carlos Soto Pérez	Operativo	02/09/2020	09/09/2020	3 a 4
Documentar controlador de NodeValues	Héctor Chávez Morales	Operativo	02/09/2020	09/09/2020	3 a 4
Documentar controlador de Values	Carlos Soto Pérez	Operativo	02/09/2020	09/09/2020	3 a 4
Documentar controlador de Usuarios	Héctor Chávez Morales	Operativo	02/09/2020	09/09/2020	3 a 4
Reunión de revisión de progreso interno	Carlos Soto Pérez, Héctor Chávez Morales	Operativo	07/09/2020	07/09/2020	4
Reunión de revisión de progreso interno	Carlos Soto Pérez, Héctor Chávez Morales y el profesor	Operativo	08/09/2020	08/09/2020	4
Refactorización de LecturaNodos	Carlos Soto Pérez	Técnica	09/09/2020	16/09/2020	4 a 5
Refactorización de Nodes	Héctor Chávez Morales	Técnica	09/09/2020	16/09/2020	4 a 5
Reunión de revisión de progreso interno	Carlos Soto Pérez, Héctor Chávez Morales	Operativo	14/09/2020	14/09/2020	5
Reunión de revisión de progreso interno	Carlos Soto Pérez, Héctor Chávez Morales y el profesor	Operativo	15/09/2020	15/09/2020	5

Crear script modelo de Nodes	Héctor Chávez Morales	Técnico	16/09/2020	30/09/2020	5 a 7
Crear script modelo de LecturaNodes	Carlos Soto Pérez	Técnico	16/09/2020	30/09/2020	5 a 7
Crear Script modelo de NodeValues	Héctor Chávez Morales	Técnico	16/09/2020	30/09/2020	5 a 7
Crear script modelo de Values	Carlos Soto Pérez	Técnico	16/09/2020	30/09/2020	5 a 7
Crear Script modelo de Usuarios	Héctor Chávez Morales	Técnico	16/09/2020	30/09/2020	5 a 7
Reunión de revisión de progreso interno	Carlos Soto Pérez, Héctor Chávez Morales	Operativo	21/09/2020	21/09/2020	6
Reunión de revisión de progreso interno	Carlos Soto Pérez, Héctor Chávez Morales y el profesor	Operativo	22/09/2020	22/09/2020	6
Reunión de revisión de progreso interno	Carlos Soto Pérez, Héctor Chávez Morales	Operativo	28/09/2020	28/09/2020	7
Reunión de revisión de progreso interno	Carlos Soto Pérez, Héctor Chávez Morales y el profesor	Operativo	29/09/2020	29/09/2020	7
Crear controlador de Nodes	Héctor Chávez Morales	Técnico	30/09/2020	07/10/2020	7 a 8
Crear controlador de LecturaNodes	Carlos Soto Pérez	Técnico	30/09/2020	07/10/2020	7 a 8
Crear controlador de NodeValues	Héctor Antonio Morales	Técnico	30/09/2020	07/10/2020	7 a 8
Crear controlador de Values	Carlos Soto Pérez	Técnico	30/09/2020	07/10/2020	7 a 8
Crear controlador de autentificación	Carlos Soto Pérez	Técnico	30/09/2020	07/10/2020	7 a 8
Crear controlador de Usuarios	Héctor Chávez Morales	Técnico	30/09/2020	07/10/2020	7 a 8
Reunión de revisión de progreso interno	Carlos Soto Pérez, Héctor Chávez Morales	Operativo	05/10/2020	05/10/2020	8
Reunión de revisión de progreso interno	Carlos Soto Pérez, Héctor Chávez Morales y el profesor	Operativo	06/10/2020	06/10/2020	8
Crear las rutas del servidor	Carlos Soto Pérez, Héctor Chávez Morales	Técnico	30/09/2020	07/10/2020	8 a 9

Reunión de revisión de progreso	Carlos Soto Pérez, Héctor Chávez Morales	Operativo	05/10/2020	05/10/2020	9
Reunión de revisión de progreso interno	Carlos Soto Pérez, Héctor Chávez Morales y el profesor	Operativo	06/10/2020	06/10/2020	9
Reunión de revisión de progreso interno	Carlos Soto Pérez, Héctor Chávez Morales	Operativo	12/10/2020	12/10/2020	10
Reunión de revisión de progreso interno	Carlos Soto Pérez, Héctor Chávez Morales y el profesor	Operativo	13/10/2020	13/10/2020	10
Reunión de revisión de progreso interno	Carlos Soto Pérez, Héctor Chávez Morales	Operativo	19/10/2020	19/10/2020	11
Reunión de revisión de progreso interno	Carlos Soto Pérez, Héctor Chávez Morales y el profesor	Operativo	20/10/2020	20/10/2020	11
Reunión de revisión de progreso interno	Carlos Soto Pérez, Héctor Chávez Morales	Operativo	26/10/2020	26/10/2020	12
Reunión de revisión de progreso interno	Carlos Soto Pérez, Héctor Chávez Morales y el profesor	Operativo	27/10/2020	27/10/2020	12
Reunión de revisión de progreso interno	Carlos Soto Pérez, Héctor Chávez Morales	Operativo	01/11/2020	02/11/2020	13
Reunión de revisión de progreso interno	Carlos Soto Pérez, Héctor Chávez Morales y el profesor	Operativo	03/10/2020	03/10/2020	13
Crear instanciador de JWTs	Carlos Soto Pérez	Técnico	07/10/2020	03/11/2020	8 a 13
Crear middlewares para verificar autentificación	Héctor Chávez Morales	Técnico	07/10/2020	03/11/2020	8 a 13
Pruebas unitarias	Carlos Soto Pérez, Héctor Chávez Morales	Técnico	09/09/2020	30/11/2020	7 a 13
Pruebas de integración	Carlos Soto Pérez, Héctor Chávez Morales	Técnico	07/10/2020	30/11/2020	7 a 13

Anexo B. Reporte técnico

Documentación del código

La documentación del código está hecha con JSDocs, y esta toma los comentarios de las funciones. Es similar a la sintaxis Javadoc en Java, pero en este caso es para JS.
JSDocs genera un HTML en donde se puede visualizar de forma amigable.

Para volver a generar la documentación es necesario tener el proyecto descargado y ejecutar el script "gen-docs" el cual generará una carpeta llamada "out" en donde se encontrará el archivo HTML.

Diseño del código

Nos apegamos al patrón de diseño MVC (Modelo-Vista-Controlador), pero haciendo una omisión al componente de Vista, dado que no existe interacción con el usuario.

El código además esta divido en más carpetas para poder diferenciar de manera clara la funcionalidad. El código está dividido principalmente en los siguientes paquetes

- ***Config*:** Aquí existen todos los archivos de configuración como la obtención de credenciales y la conexión a la base de datos, el archivo de definición JSON para *Swagger* y la encriptación de datos.

- ***Controllers*:** Aquí, como su nombre lo indica van todos los controladores y por lo tanto las validaciones son llevadas a cabo aquí. Además, las pruebas también se encuentran en este paquete y estas van sobre su respectivo controlador

- **Db:** Este paquete es el equivalente al "modelo" del patrón de diseño, aquí es en donde se hace la consulta a la base de datos necesaria.

- ***Loggers*:** Para poder mantener un registro de fácil visualización fue necesario crear este paquete y aquí se definen los archivos que escriben en los archivos que se dividen en los logs de errores y en los logs de lecturas

- ***Middleware*:** En este paquete van los middlewares, al momento solamente existen 2 que son, 'Auth' y 'Verify'. El primero se encarga de verificar que se tenga un JWT en los endpoints que necesitan una autentificación. El segundo se encarga es utilizado en los endpoints más sensibles en donde solamente lo usan usuarios autentificados y que sean administradores

- ***Routes*:** Para hacer más fácil el mantenimiento del código se decidió a y utilizamos este middleware de Express para manejar de manera separada cada ruta.

- ***Validators*:** Para poder hacer las validaciones y poder reutilizarlas, se pusieron en esta carpeta.

Validaciones por esquema

Es necesario hacer validaciones en métodos de inserción o de actualización de datos (POST, PUT, PATCH) o incluso con otros métodos, la forma en que nosotros decidimos realizar la validación de tipos fue ayudarnos con los modelos ya hechos en *Swagger*.

Estas validaciones se encuentran en la carpeta *validators* y se podrán diferenciar porque llevan el nombre de "validarEsquema" y la forma para poder validar los recibidos se hace la comparación atributo a atributo contra el modelo de *swagger*.

Swagger

Para poder documentar la API decidimos utilizar *Swagger*, que es una herramienta que permite entre otras cosas documentar y utilizar servicios web REST.

Toda la información proviene de un archivo JSON en donde se define las propiedades de cada elemento y a partir de ese documento se genera una interfaz que puede ser accesada desde un endpoint definido previamente en donde se podrá visualizar de manera.

Aquí esta las rutas están agrupadas de acuerdo con el controlador que pertenece cada una, y dentro se encuentran listadas cada uno de los *endpoints* que dispone la aplicación, además dentro de cada petición se encuentra los elementos que espera recibir y los códigos de respuestas y la información que regresa la petición.

Para poder editar y visualizar al mismo tiempo sin la necesidad de ejecutar el proyecto, se puede hacer uso de la herramienta ofrecida por la misma compañía, en donde se coloca el json o yaml y se puede visualizar en tiempo real los cambios hechos: https://editor.swagger.io/

Para acceder al archivo generado por *swagger* que se encuentra en producción puede ser accesado desde: http://papvidadigital-test.com/api/api-docs/nodos/
Es necesario actualizar manualmente cualquier cambio en el archivo json para que se pueda ver reflejado en

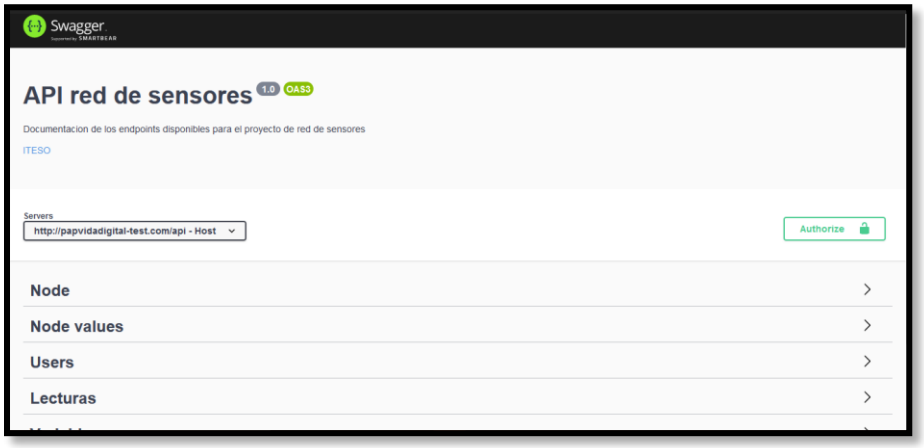


Ilustración 10. Captura de pantalla de Swagger

Contexto de ejecución

Uno de los problemas que se presentaron al momento de la implementación del código fue el manejo del pool de conexiones. En ingeniería de software, un pool de conexiones es una cache de conexiones a una base de datos para que estas sean mantenidas en futuras peticiones a la base de datos, de esta forma se mejora el rendimiento al ejecutar consultas

en una base de datos ya que los recursos son adquiridos una sola vez en el ciclo de vida de la aplicación.
Esta solución tiene como desventaja que se tiene un número limitado de conexiones simultaneas a la base de datos que se deben de manejar adecuadamente para mantener los recursos disponibles para todos. Para la librería utilizada para las conexiones a la base de datos, estas se tienen que cerrar explícitamente, para facilitar la implementación y evitar tener código repetido y aumentar la posibilidad de error, se llegó a una solución que fue llamada “ExecutionContext” la cual usa el patón de diseño *Adapter* y *Context*.
Esta implementación recibe una promesa, preferiblemente dentro de un *closure*, y retorna otra promesa.

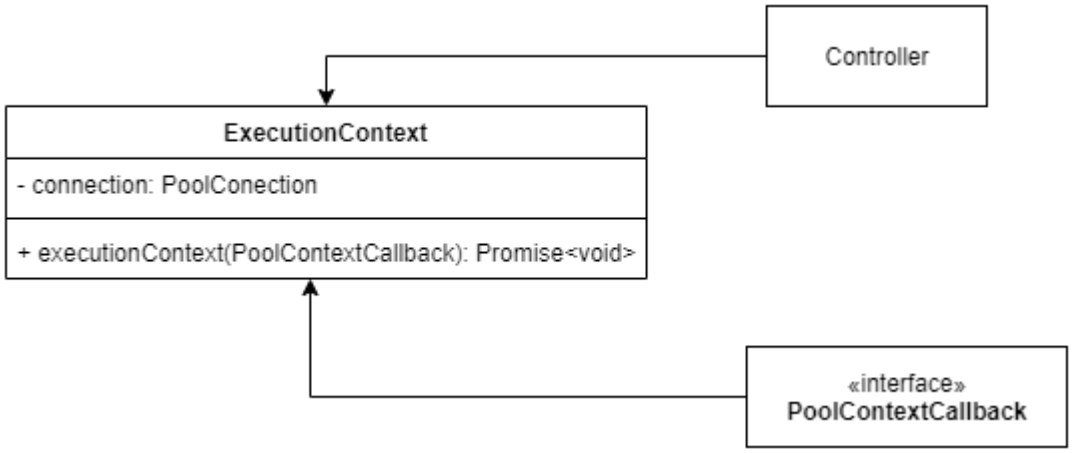


Ilustración 11. Diagrama para contexto de ejecución.

Con este contexto de ejecución se logra una comunicación entre las diferentes capas de la aplicación y con el *Adapter* se modifica el comportamiento de” ExecutionContext", logrando así, antes de cada ejecución de `PoolContextCallback` se adquiera una conexión y esa se pase dentro del contexto, al terminar la ejecución de `PoolContextCallback` se libera la conexión para ser utilizada en otra petición.
Para mantener un buen diseño y buenas prácticas se obliga a encapsular la lógica de una consulta en una función, para poder acceder a la conexión existente en el contexto de ejecución, esta se debe de pasar como el primer parámetro de la función seguido por los relevantes para la función. Esta llamada debe de retornar una promesa.

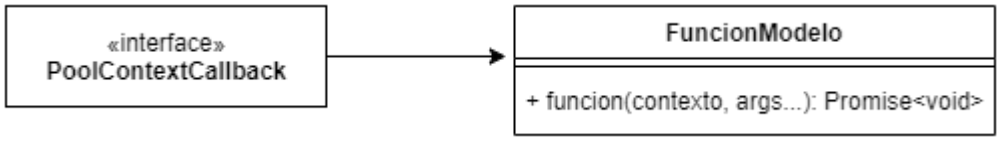


Ilustración 12. Diagrama para función modelo

Un ejemplo de uso toma la función `getVariables` del controlador variables. Podemos observar que tanto la función del modelo, la interfaz `PoolContextCallback` y el `ExectionContext` retornan una promesa. Se obliga a retornar promesas en todos los casos.

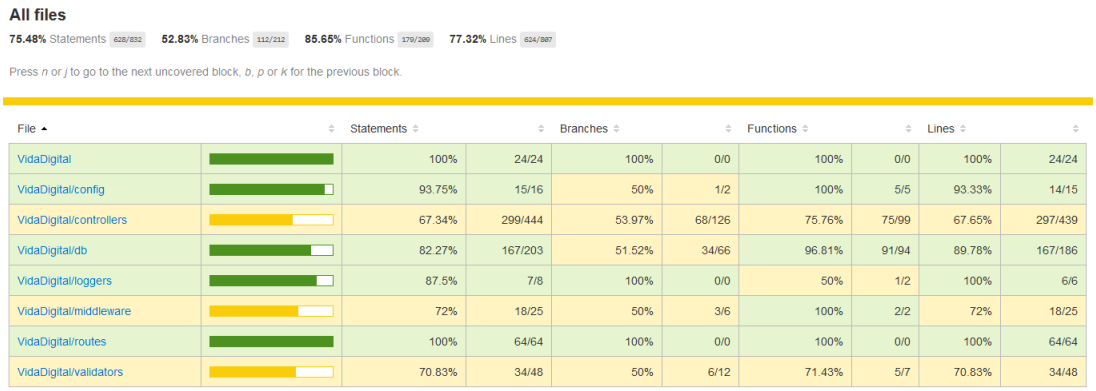
```
async function getVariables(req, res) {
  try {
    await executionContext(async (context) => {
      const { connection } = context;
      const result = await variablesModel.getVariables(connection);
      res.json(result);
    });
  } catch (e) {
    errorLog(e.message);
    res.status(500).send('Internal server error');
  }
}
```

Se obliga al desarrollador a seguir este patrón para evitar posibles errores y no cerrar una conexión antes de tiempo o nunca ser cerrada. Esta solución sigue los pasos

1. Adquisición de recursos
2. Utilización de recursos
3. Limpieza o liberación de recursos.

Pruebas

Para realizar las pruebas utilizamos el *framework* de Jest, el cual permite hacer pruebas unitarias a cualquier proyecto basado en JS, ese *framework* incluye a su vez la dependencia de *Istanbul*, la cual permite generar reportes de cobertura de manera visual.
Las pruebas están en el paquete de Controladores puesto que la prueba se hace hacia el controlador (se hace una prueba a cada endpoint).
Al ejecutar el script *test* se generarán las siguientes carpetas "coverage\lcov-report" en donde se econtrara el archivo *index* HTML. Ese archivo es un reporte visual de las pruebas hechas, en donde se podrá observar %Statements, #Branches, #Funciones y #Lineas cubiertas por cada archivo, de las pruebas escritas.



All files

75.48% Statements 628/832 **52.83%** Branches 112/212 **85.65%** Functions 179/209 **77.32%** Lines 624/807

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

File	Statements		Branches		Functions		Lines	
VidaDigital	100%	24/24	100%	0/0	100%	0/0	100%	24/24
VidaDigital/config	93.75%	15/16	50%	1/2	100%	5/5	93.33%	14/15
VidaDigital/controllers	67.34%	299/444	53.97%	68/126	75.76%	75/99	67.65%	297/439
VidaDigital/db	82.27%	167/203	51.52%	34/66	96.81%	91/94	89.78%	167/186
VidaDigital/loggers	87.5%	7/8	100%	0/0	50%	1/2	100%	6/6
VidaDigital/middleware	72%	18/25	50%	3/6	100%	2/2	72%	18/25
VidaDigital/routes	100%	64/64	100%	0/0	100%	0/0	100%	64/64
VidaDigital/validators	70.83%	34/48	50%	6/12	71.43%	5/7	70.83%	34/48

Ilustración 13. Reporte de cobertura de Jest

Integración continua

Hablando desde DevOps, La integración continua es una práctica de desarrollo de software mediante la cual los desarrolladores combinan los cambios en el código en un repositorio central de forma periódica, tras lo cual se ejecutan versiones y pruebas automáticas. La integración continua se refiere en su mayoría a la fase de creación o integración del proceso de publicación de software y conlleva un componente de automatización.

Para lograr la integración continua de los nuevos componentes de software se realizan dos acciones principales, evaluación de estilo de código y pruebas al código y estas se ejecutan cada vez que se realiza un *Pull Request* en la plataforma Azure DevOps. La configuración de las etapas de integración continua está descrita en el archivo azure-pipelines.yaml.

Para evaluar el estilo del código se utiliza una herramienta llamad Linter, para este caso en específico se utiliza el ESLint. Esta herramienta cuenta con una extensión en el editor de código Visual Studio Code. La descripción de las reglas está descrita en el repo JavaScript de Airbnb, estas se configuran en el archivo .eslint.

Phusion Passenger

Phusion Passenger es un servidor web gratuito que brinda soporte para Ruby, Python y NodeJS. Está diseñado para ser integrado con Apache o Nginx. Está pensado para tener alto rendimiento y eficiencia. Este servidor empezó a causar problemas al detener el servicio de NodeJS de forma inesperada con el siguiente error: `Checking whether to disconnect long-running connections for process 21039, application /home/papvida1/O2020 (production)`

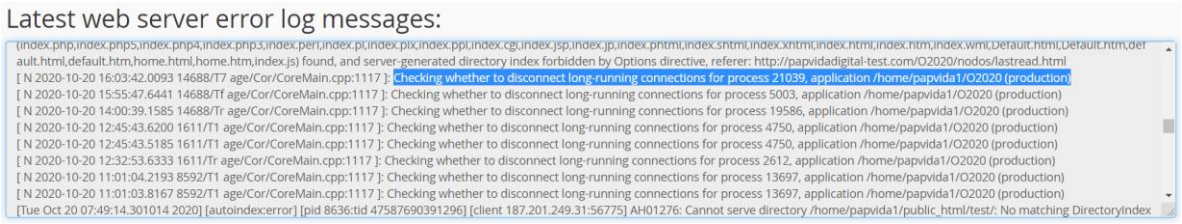


Ilustración 14. Captura del error de Passenger

Ese problema se encuentra documentado en el siguiente Issue en GitHub

Esa salida no indica ningún problema, como indica la documentación de Phusion Passenger, este es un comportamiento esperado después que la aplicación pasa una cantidad predeterminada de segundos sin actividad, este termina la aplicación para que no esté utilizando recursos, cuando se vuelve a necesitar Passenger lo vuelve a levantar.

Para lograr que la aplicación se levante de manera automática no se encontró una metodología de despliegue adecuada, la única observación que se tiene es, al momento de subir código nuevo, se requiere detener e iniciar la aplicación en lugar de solo reiniciar. Esta diferencia puede ser un problema de CPanel y no de Passenger

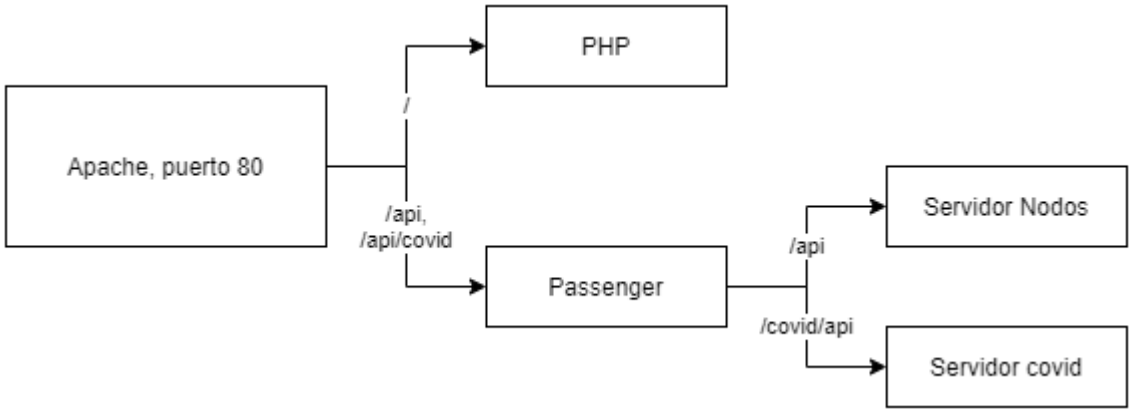


Ilustración 15. Diagrama sobre el manejo de peticiones por Apache

Contribuciones y código de conducta

Para contribuir en este repositorio, favor de pedir acceso vía correo electrónico a cualquiera de los dos contribuidores originales.

Proceso para integración de código

1. Verificar que las dependencias trabajen con la versión 9.11.x de Node Js.
2. Asegurarse que cualquier dependencia necesaria durante las pruebas locales es removida
3. Asegurarse que todas las pruebas de los componentes modificados o creados pasen en un entorno local y controlado
4. Crear un Pull Request (PR) y pedir a alguien más que lo revise con la siguiente información
    a. Título
    b. Que se cambió
    c. Porque son importantes estos cambios
    d. Cambios a futuro/código que faltó integrar.
5. Atender a los posibles errores en el pipeline

Código de conducta

Nuestras plegarias

Para el interés de los pasados, presentes y futuros desarrolladores de este código, así como del usuario final y las personas que mantienen funcionando este proyecto, pedimos que se genere un ambiente agradable durante la participación en este proyecto. Pedimos que en este proyecto sea construido junto a una comunidad de profesionales con los estándares profesionales y éticos que se asumen por ser miembro del ITESO, Universidad Jesuita de Guadalajara.

Comportamiento esperado

- Ser respetuoso con todos los participantes
- Aceptar las críticas constructivas
- Enfocarse en lo que es bueno para la comunidad
- Ser empático con los demás

Responsabilidades

Los desarrolladores de este proyecto son responsables de mantener la calidad tanto funcional y de codificación. Las buenas prácticas de programación son obligatorias
La seguridad de los datos no es un juego, los datos se deben de mantener, seguros y accesibles en todo momento. Las buenas prácticas de seguridad son obligatorias.
Todos somos responsables de El Bosque de la Primavera, este proyecto nos hace partícipes de una solución a los problemas del bosque.

Anexo C. Pruebas

Estrategia de pruebas

Para las pruebas realizadas en la aplicación de la red de sensores, se limita solo al backend. Es importante realizar estas pruebas porque así nos aseguramos de que el código nuevo se integre bien al ya existente y que estas integraciones sin errores se puedan comprobar.
El tipo de prueba deseado es a nivel de sistema, se decidió por este tipo para poder hacer peticiones por prueba, además, puesto que se tienen demasiadas llamadas asíncronas dentro de un mismo controlador se pretende de una manera fácil probar el conjunto de todas las llamadas asíncronas.
En cuanto al ambiente de pruebas, se requiere que estas sean ejecutadas sobre Node 9.11.x para garantizar la compatibilidad con el servidor donde el backend estará corriendo, para generar un ambiente de pruebas aislado de el servidor en producción se generó una instancia de una base de datos para mantener una base de datos de prueba y un proyecto en Azure Pipelines para poder ejecutar las pruebas para un proceso de CI. Puesto que se necesitan de históricos no se borrará información después de las pruebas salvo para probar las acciones DELETE de los controladores.
Para correr las pruebas se usará Jest en la versión 23.6.0 para poder ser compatible con Node 9.11.x. Para aprobar las pruebas se deberá de tener un porcentaje mayor al 75% en la cobertura de funciones.
Para cada prueba, el responsable de cada endpoint la realizará, con esto se asegura que si se presenta un error no se tendrá que esperar a que otro miembro del equipo modifique el código que ese está probando.
Al terminar la fase de pruebas, se obtendrá el reporte de cobertura de código.

Casos de prueba

Autentificación

Nombre	Endpoint	Esperado
Login	POST /api/auth/login	Estado igual a 201
Logout	DELETE /api/auth/logout	Estado igual a 200

Lecturas

Nombre	Endpoint	Esperado
Logs	GET /api/lecturas/logs	Estado igual a 200
Crear lectura	GET /api/lecturas?cmd	Estado igual a 201
Crear lectura error protocolo	GET /api/lecturas?cmd	Estad igual a 400
Crear lectura protocolo incompleto	GET /api/lecturas?cmd	Estado igual a 400
Crear lectura sin comando de inserción	GET /api/lecturas?cmd	Estado igual a 400
Crear lectura con error de validación	GET /api/lecturas?cmd	Estado igual a 400

Todos	GET /api/lecturas/t	Estado igual a 200
Todos con parámetro	GET /api/lecturas/t?count=diez	Estado igual a 400
Por Id de lectura	GET /api/lecturas/id/:id	Estado igual a 200
Por Id de lectura, inexistente	GET /api/lecturas/id/:id	Estado igual a 404
Por id de nodo	GET /api/lecturas/n/:id	Estado igual a 200
Por id de nodo, inexistente	GET /api/lecturas/n/:id	Estado igual a 404
Por día	GET /api/lecturas/dia/:año/:mes/:dia	Estado igual a 200
Por día, no existente	GET /api/lecturas/dia/:año/:mes/:dia	Estado igual a 404
Por semana	GET /api/lecturas/semana/:año/:mes/:dia	Estado igual a 200
Por semana no existente	GET /api/lecturas/semana/:año/:mes/:dia	Estado igual a 404
Por mes	GET /api/lecturas/mes/:año/:mes	Estado igual a 200
Por mes no existente	GET /api/lecturas/mes/:año/:mes/	Estado igual a 404
Por año, formato CSV	GET /api/lecturas/anio/:año?format=csv	Estado igual a 200
Por año	GET /api/lecturas/anio/:año	Estado igual a 200
Por año inexistente	GET /api/lecturas/anio/:año	Estado igual a 404

Nodos

Nombre	Endpoint	Esperado
Añadir incorrecto	POST /api/nodos	Estado iguala 400 Cuerpo de respuesta con los errores
Añadir correcto	POST /api/nodos	Estado igual a 201
Actualizar nodo incorrecto	PUT /api/nodos	Estado iguala 400 Cuerpo de respuesta con los errores
Actualizar nodo correcto	PUT /api/nodos	Estado igual a 200
Obtener nodo por ID	GET /api/nodos/:id	Estado igual a 200
Obtener todos los nodos	GET /api/nodos/nodos/todos	Estado igual a 200
Eliminar nodo	DELETE /api/nodos/:id	Estado igual a 200

Usuario

Nombre	Endpoint	Esperado
Añadir usuario, incorrecto	POST /api/usuario	Estado igual a 400
Añadir usuario	POST /api/usuario	Estado igual a 200
Obtener un usuario	GET /api/usuario/:username	Estado igual a 200
Actualizar usuario	PATCH /api/usuario/type/:username	Estado igual a 200
Actualizar contraseña	PATCH /api/usuario/password/:username	Estado igual a 200
Eliminar usuario	DELETE POST /api/usuario/:usename	Estado igual a 200
Obtener todos los usuarios	GET POST /api/usuario/todos/usuarios	Estado igual a 200

Valores

Nombre	Endpoint	Esperado
Añadir señor a nodo	POST /api/valores/nodo/sensor	Estado igual a 201
Obtener sensores por Id de nodo	GET /api/valores/sensores/:nodo	Estado igual a 200
Obtener nodos por ID de sensor	GET /api/valores/nodos/:sensor	Estado igual a 200
Verificar sensor	GET /api/valores/ nodo/:nodo/sensor/:sensor	Estado igual a 200
Eliminar sensor de nodo	DELETE /api/valores/nodo/:nodo/sensor/:sensor	Estado igual a 200

Variables

Nombre	Endpoint	Esperado
Todas las variables	GET /api/variables	Estado igual a 200
Crear variable incorrecta	POST /api/variables	Estado igual a 400 Respuesta con errores
Crear variable	POST /api/variables	Estado igual a 201
Obtener una variable	GET /api/variables/:id	Estado igual a 200
Actualizar variable	PUT /api/variables/:id	Respuesta igual a 200 Verificación de valor actualizado
Eliminar una varaible	DELETE /api/variables/:id	Estado igual a 200