**DEFINICIÓN DE TRABAJO 2**

**DOCUMENTO 3**

**Materia: Tópicos especiales en telemática**

**Profesor: Edwin Nelson Montoya**

**Estudiantes: Diego Alejandro Pérez**

**Daniel Hoyos Ospina**

**Edwin Montoya Jaramillo**

**Daniela Serna Escobar**

**Fecha: 10 de septiembre de 2017**

**Universidad EAFIT**

**Contenido del documento 2:**

**1. Miembros del equipo**

**2. Diseño de arquitectura de la Aplicación y Sistema**

1. **Vista de desarrollo**
2. **Definición de Tecnología de Desarrollo**
3. **URLs de repositorio (github)**
4. **Vista de despliegue**
5. **Definición de Tecnología – Infraestructura TI: Servidores, Software Base, Redes, etc.**
6. **URL de ejecución (http://10.131.137.x)**

**3. Implementación y Pruebas por Atributo de Calidad**

1. **Implementación**
2. **Herramientas utilizadas**
3. **Cambios en la implementación de la aplicación**
4. **Esquemas de pruebas para comprobar el Atributo de Calidad.**

**Desarrollo**

**1. Miembros del equipo**

* Daniel Hoyos Ospina
* Edwin Montoya Jaramillo
* Diego Alejandro Pérez
* Daniela Serna Escobar

**2. Diseño de arquitectura de la Aplicación y Sistema**

1. **Vista de desarrollo**
2. **Definición de Tecnología de Desarrollo**

* Sistema operativo: Microsoft Windows 10 home single language (v10.0.1293)
* lenguaje de programación: node js (v8.2.1)
* framework web – backend: express (v4.15.3)
* framework web - frontend (si aplica): angularjs (v4.3.1)
* web app server: embebido (nodejs)
* web server: nginx (v1.12.1)
* base de datos: mongodb (v3.4.6)
* editor: webstorm jetbrains (v2017.1.4)
* git (cli y gui): git(v2.11.0), sourcetree (v2.5.3)

1. **URLs de repositorio (github)  
   https://bitbucket.org/emonto15/topicostelematicaproyecto02/overview**
2. **Vista de despliegue**
3. **Definición de Tecnología – Infraestructura TI: Servidores, Software Base, Redes, etc.**
   1. Sistema operativo: Centos 7.2
   2. lenguaje de programación: node js (v8.2.1)
   3. framework web – backend: express (v4.15.3)
   4. framework web - frontend (si aplica): angularjs (v4.3.1)
   5. web app server: embebido (nodejs)
   6. web server: nginx (v1.12.1),haproxy (v1.5.18)
   7. base de datos: mongodb (v3.4.6)
   8. editor: vim (v7.4.160)
   9. git (cli y gui): git(v2.11.0)
4. **URL de ejecución (http://10.131.137.x)**

**3. Implementación y Pruebas por Atributo de Calidad**

1. **Implementación**
2. **Herramientas utilizadas**

**Disponibilidad:** HAProyxy, GlusterFS,Mongo Replication Set

**Rendimiento:** HAProyxy, GlusterFS,Mongo Replication Set

**Seguridad:** OAuth 2.0 (Google), firewall

1. **Cambios en la implementación de la aplicación**
2. **Esquemas de pruebas para comprobar el Atributo de Calidad.**

**Disponibilidad:** Para probar que la disponibilidad funciona correctamente se puede tumbar un servidor y máximo dos cajas de base de datos y la aplicación seguirá funcionando.

**Rendimiento:** Para probar que el rendimiento funciona correctamente

**Seguridad:** Para comprobar que la seguridad está funcionando correctamente podemos:

* Usar WireShark para interceptar el paquete, no se puede ver ya que utilizamos protocolo HTTPS
* Realizar una entrada no autorizada, podemos pegar la URI estática y no mostrará nada si no está logueado (seguridad en los REST)
* Tratar de pegarse al puerto 3000, no permitirá el ingreso