ANEXO I: ERS.

Véase archivo “TFA-ERS.pdf” disponible en la carpeta “/ANEXOS” del CD que acompaña el trabajo.

ANEXO II: Cómo hacer el despliegue.

A continuación se comenta sobre la arquitectura PaaS y el servicio Heroku, puede saltarse esta sección y dirigirse directamente a las instrucciones para el despliegue en el punto 2.2. si ya esta familiarizado con el termino PaaS y el servicio Heroku.

2.1 Introducción.

2.1.1 Sobre las Plataformas como Servicio o PaaS.

PaaS (en inglés *platform as a service*) es un modelo de servicio en la nube que provee herramientas para los procesos de construcción de aplicaciones en la nube; existe una estrecha analogía de entenderlo como una abstracción del sistema operativo y el middelware del entorno en la nube. PaaS provee a los desarrolladores la plataforma que estará por debajo para desarrollar su aplicación. Se encargan de dar soporte a un lenguaje o tecnología específica y la pila de servicios que necesitan usar los desarrolladores. Algunos proveedores también habilitan la posibilidad de escalar los recursos bajo demanda, tanto de computación como de almacenamiento, automáticamente liberando al administrador de la tarea de asignarlos manualmente.

En PaaS, el consumidor del servicio controla el despliegue y la configuración. El proveedor de la plataforma se encarga de proveer los servidores, la red y las necesidades computacionales que pueda tener la aplicación del consumidor.

El modelo PaaS permite tener arquitecturas multiusuario, para qué múltiples usuarios pueden usar la aplicación web de manera segura, estable, concurrente y aprueba de fallos. Servicios más sofisticados pueden también integrar entornos de desarrollo de aplicaciones, con editores colaborativos, control de versiones y despliegue [35].

Heroku, Google App Engine, Engine Yard, AWS Elastic Beanstalk, Cloud Foundry, OpenShift, Microsoft Azure, Bluemix son algunos ejemplos de plataformas PaaS.

2.1.2 Introducción a Heroku.

Heroku es un proveedor reconocido en el negocio de software en la nube, ha probado ser una solución tanto para pequeños como para grandes negocios por igual. Con la mejora continua y la filosofía de "conveniencia" sobre "configuración", se ha convertido en una plataforma reconocida, ha sido usada por más de 40.000 sitios web hasta la

fecha. La filosofía de Heroku es dejar que los desarrolladores se enfoquen únicamente en escribir sus aplicaciones web y se olviden acerca de los servidores. Heroku se hará cargo, automáticamente y bajo demanda, de construir, desplegar, ejecutar y escalar la aplicación por ellos.

Heroku es una plataforma políglota que provee soporte para Ruby, Ruby on Rails, Java, Node.js, Clojure, Scala, Python, y PHP. La arquitectura de complementos (add-on) permite a los desarrolladores customizar el servicio con el uso de más de 100 paquetes de servicios de terceros que puede agregar según sus necesidades. Tiene la flexibilidad para permitir elegir entre una cuenta básica gratuita y planes pagos que se ajustan a los requerimientos de cada sitio. Agregando complementos para potenciar las aplicaciones.

Heroku también provee flexibilidad para administrar la aplicación una vez que este desplegada en su plataforma. El desarrollador puede administrarla utilizando una herramienta de línea de comandos desde su máquina o desde un panel de control que corre en su infraestructura.

Heroku es altamente escalable, este escalamiento es transparente para el usuario, puede soportar picos de tráfico y servir hasta 1000 peticiones sostenidas por segundo.

Heroku tiene integrado Git, lo que permite un flujo de trabajo concentrado haciendo fácil compartir código, colaborar con otro desarrollador y hacer despliegues frecuentes, reduciendo el tiempo que tarda la aplicación en llegar a los usuarios [35].

Puede encontrar y seguir una guía oficial, paso a paso, para utilizar el servicio con varios lenguajes de programación en: https://devcenter.heroku.com/start

2.2 Despliegue de la aplicación en Heroku.

A continuación se describen los pasos para realizar el despliegue de la aplicación en Heroku.

Deberá ingresar los comandos mencionados a continuación en una consola o terminal del sistema operativo. La línea de comando se representa con un signo $ o con >, luego de tipear cada comando deberá presionar la tecla “Enter”.

1. Crear una cuenta en Heroku.

Crear una cuenta en Heroku. Desde: https://id.heroku.com/login

2. Instalar Heroku Toolbelt.

Descargar la versión más reciente de la aplicación “Heroku Toolbelt” para el sistema operativo que se utilice desde https://toolbelt.heroku.com/ y luego instalarla.

En Linux Mint, ingresar el comando:

$ `wget -qO- https://toolbelt.heroku.com/install-ubuntu.sh | sh`

2.1 Conectar a Heroku:

Una vez instalada la aplicación “Heroku Toolbelt”, iniciar una terminal y ejecutar el comando:

$ heroku login

2.2 Agregar las claves SSH automáticamente:

Este paso se hace por única vez, por maquina desde donde se trabaje.

$ heroku keys:add

3. Clonar el repositorio.

Posicionarse en el directorio donde desea descargar el código de la aplicación. Y luego ejecutar este comando:

$ git clone https://github.com/matiasmasca/vox.git

3.2 Crear una rama.

Este paso es necesario para poder hacer cambios o agregar contenido y hacer los commit. Ejecutar los siguientes comandos, el primero se posiciona en la carpeta “vox” y el segundo crea una nueva rama:

$ cd vox

$ git checkout -b NuevaRama

5. Crear una aplicación.

Crear una aplicación en Heroku utilizando un custom buildpack.

$ heroku login

$ heroku create --buildpack http://github.com/ruby/heroku-buildpack-ruby-jekyll.git

6. Precompilar los assets.

Precompilar los assets (sólo cuando se incluyen imágenes locales, css y javascripts)

$ rake assets:precompile

7. Enviar la aplicación.

Para crear la aplicación desde una rama especifica ejecutar:

$ git push heroku NuevaRama:master

$ heroku open

Para crear la aplicación desde la rama principal ejecutar:

$ git push heroku master

$ heroku open

8. Inicializar la base de datos.

$ heroku run rake db:setup

2.3 Integración continua: configuración de Travis CI para la aplicación Rails almacenada en GitHub y desplegada en Heroku.

A continuación se describen los pasos para vincular el repositorio de GitHub con una cuenta en Travis-CI que realiza el despliegue de la aplicación automáticamente en Heroku. Se necesita previamente tener una cuenta de GitHub con al menos un repositorio y una cuenta de Heroku.

1. Crear cuenta.

Dirigirse a https://travisci.org/ y hacer clic en 'Sign in with github' y seleccionar "allow access" de la página de GitHub que se mostrará.

2. Vincular repositorio.

Dirigirse a la página https://travisci.org/profile, donde se mostrará cada uno de los repositorios en GitHub, junto con un botón ON/OFF a su lado.

Cambiar el botón de estado OFF a ON si es necesario para el repositorio que se desea integrar con Travis-CI.

3. Configurar la cuenta de Travis-CI.

Luego hacer clic en configuración, lo cual redireccionará a una página de GitHub (https://github.com/<nombre_usuario>/<nombre_del_proyecto>/settings/hooks).

4. Vincular con Travis.

Buscar en la lista de "*Available Service Hooks"* la entrada '*travis*' y hacer clic. Esto mostrará un formulario.

4.1 Llenar los campos:

user: nombre de usuario de GitHub

token: (lo obtiene en https://travisci.org/profile/<nombre del repositorio>/profile)

4.1 Verifique que la casilla "*active"* este chequeada.

4.2 Hacer clic en "*Update settings*".

4.3 Hacer clic en "*Test Hook*".

5. Configurar los pasos de la integración.

Para aplicaciones Rails, Travis-CI utiliza los distintos *steps* estándar para hacer el despliegue en un entorno de prueba y ejecutar las pruebas.

Para indicar donde se realizará el despliegue y para tener un control sobre los pasos que se realizan, se debe configurar un archivo especial llamado ".travis.yml".

Luego se debe agregar este archivo en raíz del repositorio, dado que contiene la configuración del servicio y es allí donde se configuran las diferentes opciones que se desea se ejecuten ante cada cambio en el repositorio vinculado.

Por ejemplo, para este proyecto dicho archivo se encuentra disponible en *"CODIGO/vox/.travis.yml"* en el CD que acompaña este informe. También esta disponible en línea en https://github.com/matiasmasca/vox/blob/master/.travis.yml.

Dado los alcances del trabajo, para conocer los detalles técnicos de cómo realizar la integración continua para proyectos escritos en Ruby se puede consultar la guía de Travis-CI para Ruby en http://docs.travis-ci.com/user/languages/ruby/.

6. Realizar un commit.

Una vez agregado el archivo de configuración, debe realizar un *commit* al repositorio para que se realice la primera integración de Travis-CI.