Laboratorio de Sistemas Operativos y Redes

2do Cuatrimestre de 2016

Proyecto Taiga

Profesor: Di Biase José Luis

Alumnos: Francioni Lucio
Mancuso Emiliano

Descripción del Proyecto

En este documento se explicara cómo instalar el software “Taiga”. Explicaremos brevemente de que trata y cómo usar sus funciones básicas. También se va a detallar la especificación de la PC que utilizamos, un instructivo de su instalación, los errores que nos encontramos y cómo los solucionamos.

Introducción a la Herramienta

Taiga es una herramienta de software libre y código abierto, creada para gestionar y colaborar en proyectos ágiles, principalmente aquellos que utilizan metodología Scrum y Kanban. Su código fuente está alojado en GitHub para que sea utilizado por la comunidad.

Además, cuenta con varias opciones para trabajar. Algunas de ellas son: módulos como wiki, videoconferencia, actualización de equipo y como si fuera poco gracias a su potente API permite la integración con servicios de terceros como Slack, GitHub, GitLab, Bitbucket, HipChat, Gogs, Hall, entre otros.

Descripción de la PC Utilizada

La herramienta fue instalada en una Virtual Machine de Oracle. A continuación pasamos a detallarla:

- Sistema Operativo: Ubuntu 16.04 LTS 64bits.
- RAM: 1,5 Gb.
- Disco: 16 Gb.

Requisitos Previos

La página oficial de Taiga nos brinda un listado detallado de los requisitos previos que hay que cumplir para su correcta instalación. Estos son:

- Python igual o mayor a la versión 3.4.
- PostgreSQL igual o mayor a la versión 9.3.
- GCC & Development Headers.
- Ruby igual o mayor a la versión 2.1.
- NodeJS igual o mayor a la versión 5.0 (Junto con npm, gulp y bower).

A su vez también necesita un Hardware determinado:

- Como mínimo 0.75 Gb de RAM.
- Algo de espacio en disco, principalmente para las bases de datos.

Por último recomienda:

- Usar una imagen Ubuntu 16.04 recientemente descargada y limpia.
- La instalación debe hacerse como un usuario normal y nunca como root.

Instalación

Taiga en su página oficial cuenta con un instructivo detallado de los pasos a seguir para su instalación. Este a su vez nos brinda dos opciones: "Setup Development Enviroment" y "Setup Production Enviroment". Nosotros decidimos instalar esta última nombrada.

La instalación de la herramienta consta de tres pasos:

- Instalación del backend.
- Instalación del frontend.
- Herramientas adicionales para su correcto funcionamiento.

Además, cuenta con el instructivo para la instalación de herramientas opcionales que otorgan distintos beneficios (En este trabajo practico no hacemos uso de ellas).

1 - Instalación del Backend

1.1 - Instalación de dependencias

El backend se encuentra escrito en Phyton (3.5).

```
sudo apt-get install -y build-essential binutils-doc autoconf flex bison libjpeg-dev
sudo apt-get install -y libfreetype6-dev zlib1g-dev libzmq3-dev libgdbm-dev libncurses5-dev
sudo apt-get install -y automake libtool libffi-dev curl git tmux gettext
```

1.2 - Setup de la base de datos

Utiliza postgresql.

1.2.1 - Instalación de Postgresql

```
sudo apt-get install -y postgresql-9.5 postgresql-contrib-9.5
sudo apt-get install -y postgresql-doc-9.5 postgresql-server-dev-9.5
```

1.2.2 - Setup de la base de datos

```
sudo -u postgres createuser taiga
sudo -u postgres createdb taiga -O taiga
```

1.3 - Setup del ambiente de python

1.3.1 - Instalación de python 3.5 y virtualvwrapper

```
sudo apt-get install -y python3 python3-pip python-dev python3-dev python-pip virtualenvwrapper
sudo apt-get install libxml2-dev libxslt-dev
```

Hubo que reiniciar la consola, para que se cargue el ambiente bash con las variables y funciones del virtualvwrapper.

1.3.2 - Descarga del código de Github

```
git clone https://github.com/taigaio/taiga-back.git taiga-back
cd taiga-back
git checkout stable
```

Ver sección de problemas (1)

1.3.3 - Creación del virtualenv de nombre taiga

```
mkvirtualenv -p /usr/bin/python3.5 taiga
```

1.3.4 - Instalación de las dependencias

```
pip install -r requirements.txt
```

1.3.5 - Carga de datos básicos en la base

```
python manage.py migrate --noinput
python manage.py loaddata initial_user
python manage.py loaddata initial_project_templates
python manage.py loaddata initial_role
python manage.py compilemessages
python manage.py collectstatic --noinput
```

Esto creó un nuevo usuario de nombre "admin" y contraseña "123123".

1.3.6 - Creación de la configuración inicial

Se copió la siguiente configuración en /taiga-back/settings/local.py.

```
from .common import *

MEDIA_URL = "http://example.com/media/"
STATIC_URL = "http://example.com/static/"
ADMIN_MEDIA_PREFIX = "http://example.com/static/admin/"
SITES["front"]["scheme"] = "http"
SITES["front"]["domain"] = "example.com"
```

```
SECRET_KEY = "theveryultratopsecretkey"

DEBUG = False
TEMPLATE_DEBUG = False
PUBLIC_REGISTER_ENABLED = True

DEFAULT_FROM_EMAIL = "no-reply@example.com"
SERVER_EMAIL = DEFAULT_FROM_EMAIL

# Uncomment and populate with proper connection parameters
#  for  enable  email  sending.  EMAIL_HOST_USER  should  end  by
@domain.tld
#EMAIL_BACKEND = "django.core.mail.backends.smtp.EmailBackend"
#EMAIL_USE_TLS = False
#EMAIL_HOST = "localhost"
#EMAIL_HOST_USER = ""
#EMAIL_HOST_PASSWORD = ""
#EMAIL_PORT = 25

# Uncomment and populate with proper connection parameters
# for enable github login/singin.
#GITHUB_API_CLIENT_ID = "yourgithubclientid"
#GITHUB_API_CLIENT_SECRET = "yourgithubclientsecret"
```

1.4 - Verificación

Este paso era para asegurarnos que todo está funcionando correctamente.

```
workon taiga
python manage.py runserver
```

Ver sección de problemas (2)

A partir de este momento se pudo ver un json en la siguiente url: "http://localhost:8000/api/v1/".

2 - Instalación del Frontend

2.1 - Descarga del código de Github

```
git clone https://github.com/taigaio/taiga-front-dist.git taiga-
front-dist
cd taiga-front-dist
git checkout stable
```

2.2 - Edición de la configuración

2.2.1 - Creación del archivo

Hicimos una copia del archivo /taiga-front-dist/dist/conf.example.json en esa misma carpeta y lo renombramos como conf.json.

2.2.2 - Edición del archivo conf.json

```
{
      "api": "http://example.com/api/v1/",
      "eventsUrl": "ws://example.com/events",
      "debug": "true",
      "publicRegisterEnabled": true,
      "feedbackEnabled": true,
      "privacyPolicyUrl": null,
      "termsOfServiceUrl": null,
      "maxUploadFileSize": null,
      "contribPlugins": []
}
```

3 - Herramientas adicionales

A esta altura del instructivo contamos con la completa instalación del backend y el frontend de taiga.
El backend debe ejecutarse bajo un servidor de aplicaciones que a su vez debe ser ejecutado y supervisado por un gestor de procesos. Para esta tarea utilizaremos: circus.
Tanto el backend como el frontend deben estar expuestos al exterior, utilizando un proxy / servidor web de archivos estáticos. Para ello utilizaremos: nginx.

3.1.1 - Instalación de circus

```
sudo apt-get install circus
```

3.1.2 - Edición de la configuración de circus en el archivo /etc/circus/conf.d/taiga.ini

```
[watcher:taiga]
working_dir = /home/taiga/taiga-back
cmd = gunicorn
args = -w 3 -t 60 --pythonpath=. -b 127.0.0.1:8001 taiga.wsgi
uid = taiga
numprocesses = 1
autostart = true
send_hup = true
stdout_stream.class = FileStream
stdout_stream.filename = /home/taiga/logs/gunicorn.stdout.log
stdout_stream.max_bytes = 10485760
stdout_stream.backup_count = 4
stderr_stream.class = FileStream
```

```
stderr_stream.filename = /home/taiga/logs/gunicorn.stderr.log
stderr_stream.max_bytes = 10485760
stderr_stream.backup_count = 4

[env:taiga]
PATH = /home/taiga/.virtualenvs/taiga/bin:$PATH
TERM=rxvt-256color
SHELL=/bin/bash
USER=taiga
LANG=en_US.UTF-8
HOME=/home/taiga
PYTHONPATH=/home/taiga/.virtualenvs/taiga/lib/python3.5/site-
packages
```

Ver sección de problemas (3)

3.1.3 - Creación del directorio de registros

Taiga almacena los registros en la home del usuario, haciéndolos disponibles e inmediatamente accesibles al entrar en una máquina. Para que todo funcione, hubo que crear el directorio de registros.

```
mkdir -p /logs
```

3.1.4 - Reiniciar circus

```
sudo service circusd restart
```

3.2.1 - Instalación de nginx

Nginx se utiliza como servidor web de archivos estáticos para servir taiga-front-dist y enviar solicitudes de proxy a taiga-back.

```
sudo apt-get install -y nginx
```

3.2.2 - Configuración de nginx a través del archivo /etc/nginx/sites-available/taiga

```
server {
      listen 80 default_server;
      server_name _;

      large_client_header_buffers 4 32k;
      client_max_body_size 50M;
      charset utf-8;

      access_log /home/taiga/logs/nginx.access.log;
      error_log /home/taiga/logs/nginx.error.log;

      # Frontend
      location / {
            root /home/taiga/taiga-front-dist/dist/;
```

```
            try_files $uri $uri/ /index.html;
      }

      # Backend
      location /api {
            proxy_set_header Host $http_host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Scheme $scheme;
            proxy_set_header X-Forwarded-Proto $scheme;
            proxy_set_header                        X-Forwarded-For
$proxy_add_x_forwarded_for;
            proxy_pass http://127.0.0.1:8001/api;
            proxy_redirect off;
      }

      # Django admin access (/admin/)
      location /admin {
            proxy_set_header Host $http_host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Scheme $scheme;
            proxy_set_header X-Forwarded-Proto $scheme;
            proxy_set_header                        X-Forwarded-For
$proxy_add_x_forwarded_for;
            proxy_pass http://127.0.0.1:8001$request_uri;
            proxy_redirect off;
      }

      # Static files
      location /static {
            alias /home/taiga/taiga-back/static;
      }

      # Media files
      location /media {
            alias /home/taiga/taiga-back/media;
      }
}
```

3.2.3 - Deshabilitación de la configuración del virtualhost de nginx por default

```
sudo rm /etc/nginx/sites-enabled/default
```

3.2.4 - Habilitación de la configuración del virtualhost de nginx de taiga

```
sudo ln -s /etc/nginx/sites-available/taiga /etc/nginx/sites-
enabled/taiga
```

3.2.5 - Verificación de la configuración

```
sudo nginx -t
```

Ver sección de problemas (4)

3.2.6 - Reinicio de nginx

```
sudo service nginx restart
```

3.3.1 - Servicio HTTPS

Colocamos los certificados SSL en: /etc/nginx/ssl.
Se reemplazó la configuración original del puerto 80 para que los usuarios sean redirigidos automáticamente a la versión HTTPS.
Segundo, necesitamos generar un parámetro GHE más fuerte.

```
cd /etc/ssl
sudo openssl dhparam -out dhparam.pem 4096
```

3.3.2 - Nueva configuración de /etc/nginx/sites-available/taiga

```
server {
      listen 80 default_server;
      server_name _;
      return 301 https://$server_name$request_uri;
}

server {
      listen 443 ssl default_server;
      server_name _;

      large_client_header_buffers 4 32k;
      client_max_body_size 50M;
      charset utf-8;

      index index.html;

# Frontend
      location / {
            root /home/taiga/taiga-front-dist/dist/;
            try_files $uri $uri/ /index.html;
      }

# Backend
      location /api {
      proxy_set_header Host $http_host;
      proxy_set_header X-Real-IP $remote_addr;
      proxy_set_header X-Scheme $scheme;
      proxy_set_header X-Forwarded-Proto $scheme;
      proxy_set_header X-Forwarded-For
$proxy_add_x_forwarded_for;
      proxy_pass http://127.0.0.1:8001/api;
      proxy_redirect off;
      }

      location /admin {
            proxy_set_header Host $http_host;
```

```
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Scheme $scheme;
            proxy_set_header X-Forwarded-Proto $scheme;
            proxy_set_header X-Forwarded-For
$proxy_add_x_forwarded_for;
      proxy_pass http://127.0.0.1:8001$request_uri;
      proxy_redirect off;
      }

      # Static files
      location /static {
      alias /home/taiga/taiga-back/static;
      }

      # Media files
      location /media {
      alias /home/taiga/taiga-back/media;
      }

      add_header Strict-Transport-Security "max-age=63072000;
includeSubdomains; preload";
      add_header Public-Key-Pins 'pin-
sha256="klO23nT2ehFDXCfx3eHTDRESMz3asj1muO+4aIdjiuY="; pin-
sha256="633lt352PKRXbOwf4xSEa1M517scpD3l5f79xMD9r9Q="; max-
age=2592000; includeSubDomains';

      ssl on;
      ssl_certificate /etc/nginx/ssl/example.com/ssl-bundle.crt;
      ssl_certificate_key
/etc/nginx/ssl/example.com/example_com.key;
      ssl_session_timeout 5m;
      ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
      ssl_prefer_server_ciphers on;
      ssl_ciphers 'ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-
AES256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-DSS-AES128-GCM-
SHA256:kEDH+AESGCM:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES128-
SHA256:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-
AES256-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-
SHA:ECDHE-ECDSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES128-
SHA:DHE-DSS-AES128-SHA256:DHE-RSA-AES256-SHA256:DHE-DSS-AES256-
SHA:DHE-RSA-AES256-
SHA:!aNULL:!eNULL:!EXPORT:!DES:!RC4:!3DES:!MD5:!PSK';
      ssl_session_cache shared:SSL:10m;
      ssl_dhparam /etc/ssl/dhparam.pem;
      ssl_stapling on;
      ssl_stapling_verify on;
      }
```

3.3.3 - Actualización de /taiga-back/settings/local.py

Antes de activar el sitio HTTPS, la configuración para el front-end y back-end tiene que ser actualizada.
Cambiamos el esquema de http a https.

from .common import *

```
MEDIA_URL = "https://example.com/media/"
STATIC_URL = "https://example.com/static/"
ADMIN_MEDIA_PREFIX = "https://example.com/static/admin/"
SITES["front"]["scheme"] = "https"
SITES["front"]["domain"] = "example.com"

SECRET_KEY = "theveryultratopsecretkey"

DEBUG = False
TEMPLATE_DEBUG = False
PUBLIC_REGISTER_ENABLED = True

DEFAULT_FROM_EMAIL = "no-reply@example.com"
SERVER_EMAIL = DEFAULT_FROM_EMAIL

# Uncomment and populate with proper connection parameters
# for enable email sending.
#EMAIL_BACKEND = "django.core.mail.backends.smtp.EmailBackend"
#EMAIL_USE_TLS = False
#EMAIL_HOST = "localhost"
#EMAIL_HOST_USER = ""
#EMAIL_HOST_PASSWORD = ""
#EMAIL_PORT = 25

# Uncomment and populate with proper connection parameters
# for enable github login/singin.
#GITHUB_API_CLIENT_ID = "yourgithubclientid"
#GITHUB_API_CLIENT_SECRET = "yourgithubclientsecret"
```

3.3.4 - Reinicio de circus

```
sudo service circusd restart
```

3.3.5 - Actualización de /taiga-front-dist/dis/conf.json

```
{
        "api": "https://example.com/api/v1/",
        "eventsUrl": "wss://example.com/events",
        "debug": "true",
        "publicRegisterEnabled": true,
        "feedbackEnabled": true,
        "privacyPolicyUrl": null,
        "termsOfServiceUrl": null,
        "maxUploadFileSize": null
}
```

3.2.6 - Recarga de la configuración de nginx

```
sudo service nginx reload
```

A partir de este momento ya podemos utilizar la aplicación Taiga.

Ver sección de problemas (5)

Problemas

(1) - Cuando quisimos clonar el repositorio del backend de Github, por consola se lanzo un aviso de que no teníamos instalado git, mostrándonos el comando con el que deberíamos instalarlo:

```
sudo apt-get install git
```

Una vez instalado esto, clonamos el repositorio sin inconvenientes.

(2) - A la hora de ejecutar el archivo para correr el backend, surgió el siguiente inconveniente:

```
(taiga) lucio@lucio-VirtualBox:~/taiga-back$ python manage.py runserver
Traceback (most recent call last):
  File "manage.py", line 27, in <module>
    from django.core.management import execute_from_command_line
ImportError: No module named 'django'
```

Después de investigar, encontramos que esto ocurrió ya que teníamos instalado django en nuestra pc, pero no en el ambiente que habíamos creado. Por lo tanto lo solucionamos instalándolo de la siguiente manera dentro del "enviroment: taiga":

```
sudo apt-get install python3-django
```

Una vez solucionado esto, nos apareció el siguiente problema:

```
  File "/home/lucio/.virtualenvs/taiga/lib/python3.5/site-packages/psycopg2/__init__.py", line 164, in connect
    conn = _connect(dsn, connection_factory=connection_factory, async=async)
django.db.utils.OperationalError: FATAL:  role "lucio" does not exist
```

El problema acá fue que queríamos trabajar con el nombre de usuario de la pc (lucio) para el cual no habíamos creado un rol en nuestra base de datos. Lo solucionamos creando un rol para este usuario junto con su base de datos:

```
sudo -u postgres createuser lucio
sudo -u postgres createdb taiga -O lucio
```

(3) - Cuando quisimos crear el archivo taiga.ini nos notifica que no tenemos permiso para ello, por lo tanto utilizamos los siguientes comandos:

```
cd /etc/circus
sudo chmod 777 conf.d
```

Este comando establece permisos totales de lectura, escritura y ejecución para todos los usuarios tanto a la carpeta como a todas las subcarpetas y archivos incluidos dentro de ella, en este caso la carpeta "conf.d". De esta manera, el problema fue solucionado y pudimos crear el archivo sin inconvenientes.

(4) - Cuando quisimos verificar que la configuración de nginx era correcta nos encontramos con el siguiente error:

```
lucio@lucio-VirtualBox:~$ nginx -t
nginx: [alert] could not open error log file: open() "/var/log/nginx/error.log" failed (13: Permission denied)
2016/12/13 19:52:29 [warn] 27915#27915: the "user" directive makes sense only if the master process runs with super-user privileges, ignored i
n /etc/nginx/nginx.conf:1
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
2016/12/13 19:52:29 [emerg] 27915#27915: open() "/run/nginx.pid" failed (13: Permission denied)
nginx: configuration file /etc/nginx/nginx.conf test failed
```

Después de buscar donde podía encontrarse el error, encontramos que se debía a que lo iba a buscar a una carpeta que no correspondía. A continuación copio la configuración de archivo taiga (En rojo como era antes y en verde como es ahora corregido):

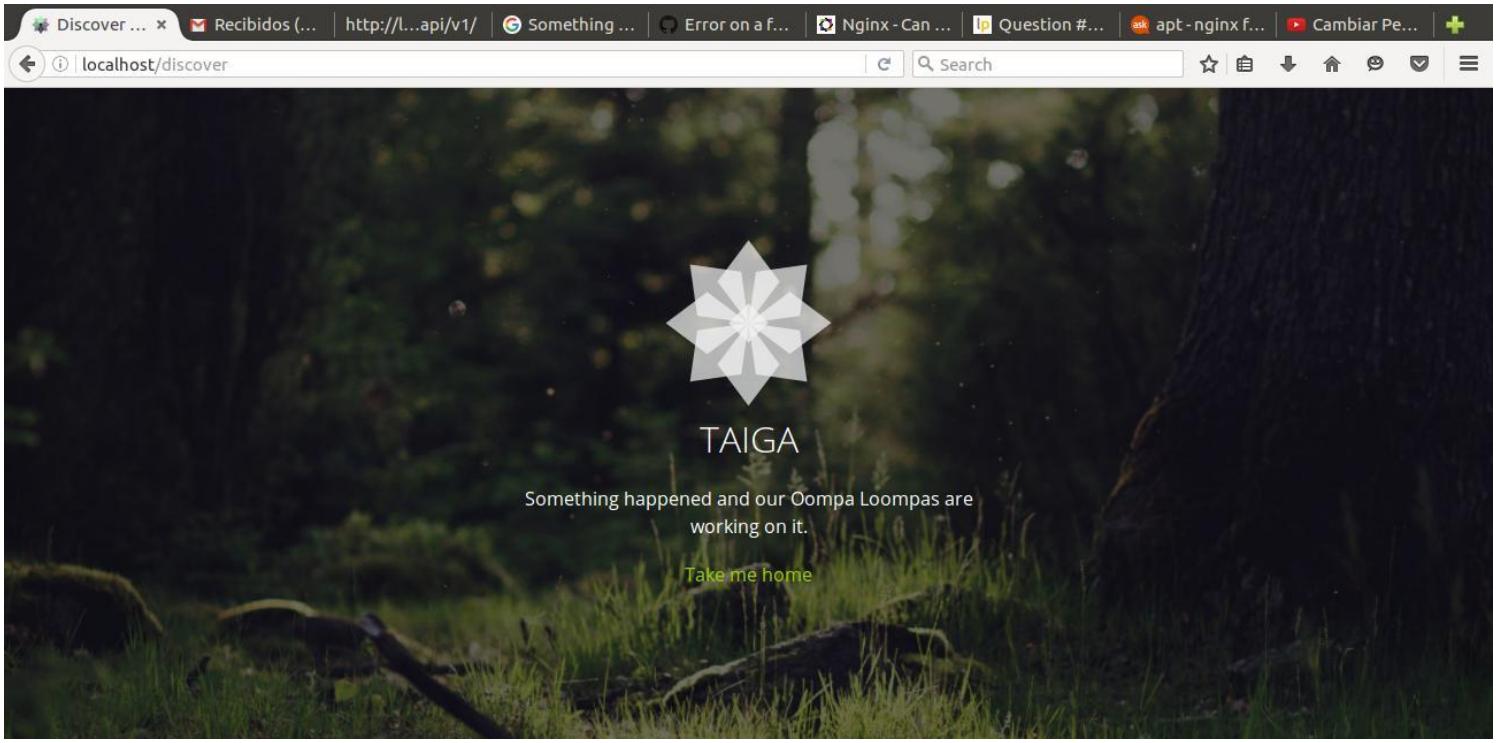
```
}
server {
      listen 80 default_server;
      server_name _;

      large_client_header_buffers 4 32k;
      client_max_body_size 50M;
      charset utf-8;

      access_log /home/taiga/logs/nginx.access.log;
      access_log /home/lucio/taiga/logs/nginx.access.log;
      error_log /home/taiga/logs/nginx.error.log;
      error_log /home/lucio/taiga/logs/nginx.error.log;

      # Frontend
      location / {
            root /home/taiga/taiga-front-dist/dist/;
            root /home/licio/taiga-front-dist/dist/;
            try_files $uri $uri/ /index.html;
      }

      # Backend
      location /api {
            proxy_set_header Host $http_host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Scheme $scheme;
            proxy_set_header X-Forwarded-Proto $scheme;
            proxy_set_header                        X-Forwarded-For
$proxy_add_x_forwarded_for;
            proxy_pass http://127.0.0.1:8001/api;
            proxy_redirect off;
      }

      # Django admin access (/admin/)
      location /admin {
            proxy_set_header Host $http_host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Scheme $scheme;
            proxy_set_header X-Forwarded-Proto $scheme;
            proxy_set_header                        X-Forwarded-For
$proxy_add_x_forwarded_for;
```

```
        proxy_pass http://127.0.0.1:8001$request_uri;
        proxy_redirect off;
}

# Static files
location /static {
        alias /home/taiga/taiga-back/static;
        alias /home/lucio/taiga-back/static;
}

# Media files
location /media {
        alias /home/taiga/taiga-back/media;
        alias /home/lucio/taiga-back/media;
}
```

(5) - Al ingresar al localhost y tratar de correr la aplicación, notamos que si bien levanta tanto el backend como el frontend, esta no se puede utilizar.

Investigando un poco, lamentablemente descubrimos que la aplicación no funciona para maquinas virtuales, problema que de haberlo sabido antes hubiéramos tenido en cuenta para buscar otra máquina donde instalarlo.

Tomamos la decisión de trabajar en una maquina virtual debido a que el instructivo de instalación recomendaba utilizar una imagen Ubuntu recién descargada y limpia, y es por esto que elegimos trabajar de esta manera sin pensar que nos encontraríamos con este problema sobre el final.



Uso de Taiga

A continuación haremos una breve explicación de la utilización de la aplicación:

Para comenzar a disfrutar de Taiga debemos registrarnos de manera gratuita por alguno de los medios que la herramienta te brinda.

El primer paso es crear un proyecto, puede ser un proyecto Kanban o Scrum. Cualquiera que elijamos se puede adaptar fácilmente a las necesidades y gustos.

Seguidamente, vamos a dar un nombre a el proyecto y a escribir una descripción para el mismo. Una vez creado el proyecto en Taiga lo primero que observamos es el BackLog, donde podemos añadir las historias de usuario del proyecto.

Cada historia de usuario se estima por lo general en puntos y se debe tener en claro que no se debe indicar el tiempo de la tarea, es importante destacar que la estimación en Taiga se puede hacer por roles**.**

Se pueden añadir tantas historias de usuario como necesite el proyecto. Al crear una nueva historia se debe colocar un título, la estimación, el estado, etiquetas y la descripción de la tarea**.** Además se puede segmentar la tarea, si es requerida por el equipo o por el cliente.

Una vez creada todas las tareas necesarias para el proyecto, debemos crear lo que en Scrum se llama Sprint. Esto es la agrupación de un conjunto de tareas que puede representar un producto funcional y que está planificado para que se realice en un período de tiempo determinado**.** Un proyecto puede tener tantos Sprint como sean necesarios y cada Sprint debe tener como resultado un prototipo.

Al sprint se le pueden agregar las tarjetas creadas. Taiga permite hacer esto de una manera fácil arrastrando y soltando cada tarea en el sprint que deseas.

Se puede priorizar las tareas con lo que se determina cuál se debe hacer primero. También, Taiga nos permite añadir miembros para que colaboren en las tareas, por ejemplo un diseñador.

Una vez creado el sprint, ya planificado y con los miembros ya listos para comenzar, nos dirigimos a. panel de tareas del Sprint, que cuenta con varias columnas, las cuales representan cada una lo siguiente:

- Historia de Usuario: Todas las historias de usuario que conforman el Sprint.
- Nueva: Cada historia de usuario se puede dividir en tareas.
- En Curso: Son aquellas tareas que se están realizando en este momento.
- Lista para Testear: Son aquellas tareas que están terminadas pero que no se han probado.
- Cerrada: Son aquellas tareas que han sido terminadas
- Necesita Información: Son aquellas tareas que se necesita una información extra para poder ser culminada.

Las tareas son tomadas por el colaborador que la va a realizar, el cual se encargará de cambiarla por el estado que le corresponda. La idea es que todo el equipo conozca el estado del proyecto y trabajen sincronizados.

El objetivo del Sprint es que todas las tareas sean concretadas, Taiga nos ofrece un gráfico que nos permite ver en todo momento como se encuentran.

Al terminar todas las tareas el sprint ha concluido.

Por último, cabe destacar, que Taiga nos ofrece una wiki del proyecto en el que podemos ir colocando información referente al proyecto. La sintaxis a utilizar es Markdown**.** También se puede ir añadiendo enlaces que necesites para tu proyecto.

Conclusiones

Podemos decir que la herramienta es muy buena, sencilla de personalizar y tomar el control de todo. Es de fácil aprendizaje y su nivel de usabilidad es excelente.

Cuenta con una documentación completa para poder profundizar en cada una de sus características. Permite dar la configuración que se desea a cada proyecto y extender sus funcionalidades gracias a sus módulos e integrar con otras herramientas.

En cuanto a su instalación, si bien esta bastante detallada en la página oficial de Taiga, cualquier inconveniente que surja resulta difícil de solucionar. Esto se debe a que al ser una herramienta relativamente nueva, no hay demasiada información respecto de sus errores, por lo que la mayoría los pudimos solucionar buscando errores similares en otras aplicaciones.

Por último, cabe destacar que es una lástima que la aplicación no funcione aun para maquinas virtuales. Debería estar comentado en la instalación que por el momento no se puede utilizar por este medio. Por lo que investigamos están trabajando en ello.