

Установка и настройка системы управления проектами Taiga

23 мая 2019 • 5506 Просмотров



Организовать - это значит сначала оценить возможность, а уже потом ставить задачу...

— *Неизвестный автор*

Искусство организовать работу и следить за ее исполнением в современном обществе является неотъемлемой частью успешного бизнеса. Всё начинается с малого. Сначала Вы один работаете над каким-нибудь проектом - всё легко, есть одна задача и вы её реализуете. Достаточно держать эту задачу в своей памяти. Но с каждым днем появляются подзадачи, добавляются заметки и приходит осознание того, что всё уже не получается запоминать. Тут можно обойтись ежедневником. Количество задач растёт и Вы понимаете, что тянуть проект в одного становится невозможно. У Вас появляются подчиненные, Вы перекладываете часть задач на своих подчиненных, но как следить за выполнением поставленных задач? Как переназначать задачи? Тут явно уже будет не хватать ежедневника.

За всю историю человечества были реализованы множества проектов от самых простых до невероятно сложных. Для достижения поставленных целей и организации управления были разработаны различные системы управления проектами, каждая имеет свои преимущества и недостатки, но, к сожалению, разбирать мы их не будем, поскольку разговор про них является очень большой отдельной темой.

В современном мире в качестве помощи и удобства менеджменту для управления проектами были разработаны различные ПО (программное обеспечение). Их довольно много, но сегодня мы рассмотрим достойное ПО системы управления проектами [Taiga](#) использующий Scrum и Kanban семейства методологий Agile. В основном это ПО используется разработчиками и дизайнерами. К сожалению, даже при бесплатности проекта в русском сегменте Интернета довольно мало информации об этом ПО. Поэтому мы - команда Syncweb, надеемся, что благодаря нашим стараниям, упростим порог входа для начала использования данного проекта.

Проверяем требования

Для начала проверим необходимые минимальные требования к Taiga.io:

- должен быть root доступ;
- Python ≥ 3.4 ;
- СУБД PostgreSQL ≥ 9.4 ;
- RabbitMQ;



- не менее 0,75 ГБ оперативной памяти (требуется для установки lxml), на всякий случай возьмем 1ГБ.
- Во время установки потребуется: Ruby >= 2.1 и NodeJS >= 7.0

Для более подробной информации можно обратиться к [официальному сайту](#).

Вообще установить можно [несколькими способами](#), но мы не ищем легких путей и воспользуемся ручным. В качестве операционной системы (ОС) будем использовать Ubuntu 18.04.

В ходе установки нужно будет изменять или создавать достаточно много файлов, и для каждой установки должны быть указаны свои параметры. Такие параметры мы будем заключать в скобки и названием на английском с пояснениями, как пример - .

Так же мы используем зачастую консольный текстовый редактор vi, Вы можете использовать любой другой. Для входа в режим вставки текста нажмите **i** и начинайте ввод. А для сохранения изменений и закрытия файла сначала нажмите **Esc**, затем **:wq**.

Установка необходимых пакетов

Для начала установим несколько удобных пакетов:

```
# apt update
# apt install -y mc net-tools
```

Создадим пользователя supp:

```
# adduser supp
```

И разрешим доступ через sudo:

```
# adduser supp
Adding user `supp' to group `sudo' ...
Adding user supp to group sudo
Done.
```



Затем отредактируем файл **/etc/ssh/sshd_config**

Раскомментируем строчку

```
PermitRootLogin no
```

И добавим

```
AllowUsers supp
```

Теперь перезапустим демона SSH:

```
# systemctl restart sshd
```

Перезайдем под пользователем supp. Предварительно установим необходимые пакеты:

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

Установка Nginx

Далее необходимо установить Nginx, в официальном репозитории имеется версия 1.14, поэтому воспользуемся инструкцией с официального сайта.

Теперь установим Nginx. Для начала установим необходимые пакеты для подключения репозитория:

```
$ sudo apt install -y curl gnupg2 ca-certificates lsb-release
```



Теперь подключим репозиторий стабильной версии:

```
$ echo "deb http://nginx.org/packages/ubuntu `lsb_release -cs` nginx" | sudo tee /e
```

Импортируем ключ для проверки подлинности пакетов:

```
$ curl -fsSL https://nginx.org/keys/nginx_signing.key | sudo apt-key add -  
OK
```

А теперь установим:

```
$ sudo apt update  
$ sudo apt install nginx
```

Инструкции для самостоятельной установки можно посмотреть на [официальной странице](#) Nginx.

Затем устанавливаем RabbitMQ и Redis:

```
$ sudo apt install -y rabbitmq-server redis-server
```

Установка PostgreSQL

Теперь установим СУБД PostgreSQL, но текущая версия в репозитории 10, а официальная последняя версия 11.3. Если посмотреть тестирования производительности различных версий PostgreSQL, например, на сайте [Habr](#), то можно увидеть увеличение производительности более новых версий по сравнению с предыдущими. Поэтому поставим с официального сайта, воспользовавшись [инструкцией](#).

Для начала добавим репозиторий:



```
$ echo "deb http://apt.postgresql.org/pub/repos/apt/ bionic-pgdg main" | sudo tee /
```

Импортируем ключ проверки подлинности пакетов:

```
$ wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-k  
OK  
$ sudo apt update
```

Продолжаем установку:

```
$ sudo apt install -y postgresql-11 postgresql-contrib-11  
$ sudo apt install -y postgresql-doc-11 postgresql-server-dev-11  
$ sudo apt install -y python3 python3-pip python3-dev virtualenvwrapper  
$ sudo apt install -y libxml2-dev libxslt-dev  
$ sudo apt install -y libssl-dev libffi-dev
```

Теперь перед дальнейшими действиями перезагрузим сервер:

```
$ sudo shutdown -r now
```

После перезагрузки сервера авторизуемся под нашим пользователем `supr` и создадим пользователя `taiga`:

```
$ sudo adduser taiga  
$ sudo adduser taiga sudo  
$ sudo su taiga  
$ cd ~
```



ВНИМАНИЕ! Дальнейшие действия необходимо производить из под пользователя Taiga



Инициализируем настройку СУБД PostgreSQL:

```
$ sudo -u postgres createuser taiga
$ sudo -u postgres createdb taiga -O taiga --encoding='utf-8' --locale=en_US.utf8 -
```

Теперь произведем настройки для RabbitMQ, где **{YOUR_PASSWORD_EVENTS}** - пароль RabbitMQ, который будет использоваться в конфигурационных файлах ниже

```
$ sudo rabbitmqctl add_user taiga {YOUR_PASSWORD_EVENTS}
Creating user "taiga"
$ sudo rabbitmqctl add_vhost taiga
Creating vhost "taiga"
$ sudo rabbitmqctl set_permissions -p taiga taiga ".*" ".*" ".*"
Setting permissions for user "taiga" in vhost "taiga"
```

Установка BACKEND

Теперь произведем настройки для RabbitMQ, где **{YOUR_PASSWORD_EVENTS}** - пароль RabbitMQ, который будет использоваться в конфигурационных файлах ниже

```
$ cd ~
$ git clone https://github.com/taigaio/taiga-back.git taiga-back
$ cd taiga-back
$ git checkout stable
Branch 'stable' set up to track remote branch 'stable' from 'origin'.
Switched to a new branch 'stable'
```

Создаем virtualenv taiga:

```
$ mkvirtualenv -p /usr/bin/python3 taiga
Already using interpreter /usr/bin/python3
Using base prefix '/usr'
New python executable in /home/taiga/.virtualenvs/taiga/bin/python3
```



Also creating executable in /home/taiga/.virtualenvs/taiga/bin/python
Installing setuptools, pkg_resources, pip, wheel...done.

И устанавливаем необходимые пакеты:

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

Теперь инициализируем базу данных:

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

После этих действий создается пользователь **admin** с паролем **123123**. Запомним, поскольку будем вводить эти данные при первичной авторизации в web-интерфейсе.

Для создания не пустой базы, например, чтобы посмотреть как создавать и работать с taiga.io или для демонстрации работы необходимо также ввести команду:

```
$ python manage.py sample_data
```

Для завершения настроек backend'а необходимо создать файл настроек **~/taiga-back/settings/local.py** со следующим содержимым, где:

- **{YOUR_DOMAIN}** - Ваш домен, например, example.ru;
- **{YOUR_SECRET_KEY}** - секретный ключ, чем сложнее и длиннее - тем лучше, его же будем использовать при конфигурации файла для событий (EVENTS) чуть ниже;
- **{YOUR_PASSWORD_EVENTS}** - пароль для пушей для событий (Events), который прописывали при настройке RabbitMQ;
- **{YOUR_GOOGLE_EMAIL}** - Ваш Email Google, поскольку в данном примере мы используем настройки от почтовика Google;
- **{YOUR_PASSWORD_EMAIL}** - пароль от Email.



Так же просим обратить внимание на протокол HTTPS

```
$ vi ~/taiga-back/settings/local.py
```

```
from .common import *
```

```
MEDIA_URL = "https://{YOUR_DOMAIN}/media/"
```

```
STATIC_URL = "https://{YOUR_DOMAIN}/static/"
```

```
SITES["front"]["scheme"] = "https"
```

```
SITES["front"]["domain"] = "{YOUR_DOMAIN}"
```

```
SECRET_KEY = "{YOUR_SECRET_KEY}"
```

```
DEBUG = False
```

```
PUBLIC_REGISTER_ENABLED = True
```

```
DEFAULT_FROM_EMAIL = "no-reply@{YOUR_DOMAIN}"
```

```
SERVER_EMAIL = DEFAULT_FROM_EMAIL
```

```
#CELERY_ENABLED = True
```

```
EVENTS_PUSH_BACKEND = "taiga.events.backends.rabbitmq.EventsPushBackend"
```

```
EVENTS_PUSH_BACKEND_OPTIONS = {"url": "amqp://taiga:{YOUR_PASSWORD_EVENTS}@localhos
```

```
# 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 = True
```

```
EMAIL_HOST = "smtp.gmail.com"
```

```
EMAIL_HOST_USER = "{YOUR_GOOGLE_EMAIL}"
```

```
EMAIL_HOST_PASSWORD = "{YOUR_PASSWORD_EMAIL}"
```

```
EMAIL_PORT = 587
```

```
# Uncomment and populate with proper connection parameters
```

```
# for enable github login/singin.
```

```
#GITHUB_API_CLIENT_ID = "yourgithubclientid"
```

```
#GITHUB_API_CLIENT_SECRET = "yourgithubclientsecret"
```

Теперь осталось запустить backend и проверить:



```
$ workon taiga
$ python manage.py runserver
Trying import local.py settings...
Trying import local.py settings...
Performing system checks...

System check identified no issues (0 silenced).
May 14, 2019 - 23:16:15
Django version 1.11.20, using settings 'settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

Установка FRONTEND

Теперь приступим к установке frontend'a.

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

Затем копируем пример конфигурационного файла:

```
$ cp ~/taiga-front-dist/dist/conf.example.json ~/taiga-front-dist/dist/conf.json
```

И изменим содержимое файла по примеру ниже, где:

- **{YOUR_DOMAIN}** - Ваш домен, например, example.ru;
- параметр **defaultLanguage: ru** - говорит о языке по умолчанию;
- параметр **publicRegisterEnabled: true** отвечает за возможность свободной регистрации на главной странице, если Вам не нужно, чтобы любой мог зарегистрироваться - поставьте значение **false**:



```
$ sudo vi ~/taiga-front-dist/dist/conf.json
```

```
{  
  "api": "https://{YOUR_DOMAIN}/api/v1/",  
  "eventsUrl": "wss://{YOUR_DOMAIN}/events",  
  "eventsMaxMissedHeartbeats": 5,  
  "eventsHeartbeatIntervalTime": 60000,  
  "eventsReconnectTryInterval": 10000,  
  "debug": false,  
  "debugInfo": false,  
  "defaultLanguage": "ru",  
  "themes": ["taiga"],  
  "defaultTheme": "taiga",  
  "publicRegisterEnabled": true,  
  "feedbackEnabled": true,  
  "supportUrl": "https://tree.taiga.io/support",  
  "privacyPolicyUrl": null,  
  "termsOfServiceUrl": null,  
  "GDPRUrl": null,  
  "maxUploadFileSize": null,  
  "contribPlugins": [],  
  "tribeHost": null,  
  "importers": [],  
  "gravatar": true,  
  "rtlLanguages": ["fa"]  
}
```

Установка EVENTS

Немного сложновата установка, но мы же сами выбрали такой путь! Теперь займемся установкой событий (Events):

```
$ cd ~  
$ git clone https://github.com/taigaio/taiga-events.git taiga-events  
$ cd taiga-events
```

Теперь установим nodejs:



```
$ curl -sL https://deb.nodesource.com/setup_8.x | sudo -E bash -  
$ sudo apt install -y nodejs
```

Установим также зависимости javascript:

```
$ npm install
```

Теперь настроим конфигурационный файл событий предварительно скопировав его:

```
$ cp config.example.json config.json
```

Примерное содержимое файла будет следующим, где:

- **{YOUR_PASSWORD_EVENTS}** - пароль, который создавали при настройке RabbitMQ;
- **{YOUR_SECRET_KEY}** - ключ, который использовали в файле конфигурации при настройке backend'a local.py:

```
$ sudo vi ./config.json
```

```
{  
  "url": "amqp://taiga:{YOUR_PASSWORD_EVENTS}@localhost:5672/taiga",  
  "secret": "{YOUR_SECRET_KEY}",  
  "websocketServer": {  
    "port": 8888  
  }  
}
```

Теперь необходимо добавить события как службу. Для этого создаем файл и приводим к виду ниже:

```
$ sudo vi ./config.json
```



```
[Unit]
Description=taiga_events
After=network.target

[Service]
User=taiga
WorkingDirectory=/home/taiga/taiga-events
ExecStart=/bin/bash -c "node_modules/coffeescript/bin/coffee index.coffee"
Restart=always
RestartSec=3

[Install]
WantedBy=default.target
```

Осталось добавить службу в автозапуск и запустить её:

```
$ sudo systemctl daemon-reload
$ sudo systemctl start taiga_events
$ sudo systemctl enable taiga_events
```

Со службой событий мы справились, но теперь необходимо сделать тоже самое и для backend'а. поэтому снова создаем файл со следующим содержимым:

```
$ sudo vi /etc/systemd/system/taiga.service
```

```
[Unit]
Description=taiga_back
After=network.target

[Service]
User=taiga
Environment=PYTHONUNBUFFERED=true
WorkingDirectory=/home/taiga/taiga-back
ExecStart=/home/taiga/.virtualenvs/taiga/bin/gunicorn --workers 4 --timeout 60 -b 1
Restart=always
RestartSec=3
```



```
[Install]
```

```
WantedBy=default.target
```

Теперь запускаем новую службу taiga и добавляем в автозапуск:

```
$ sudo systemctl daemon-reload
$ sudo systemctl start taiga
$ sudo systemctl enable taiga
```

Теперь проверим службу, что она успешно запустилась:

```
$ sudo systemctl status taiga
* taiga.service - taiga_back
   Loaded: loaded (/etc/systemd/system/taiga.service; enabled; vendor preset: en
   Active: active (running) since Fri 2019-05-17 04:09:22 +05; 3min 8s ago
 Main PID: 15247 (gunicorn)
    Tasks: 5 (limit: 1131)
   CGroup: /system.slice/taiga.service
           |-15247 /home/taiga/.virtualenvs/taiga/bin/python3 /home/taiga/.virtu
           |-15251 /home/taiga/.virtualenvs/taiga/bin/python3 /home/taiga/.virtu
           |-15252 /home/taiga/.virtualenvs/taiga/bin/python3 /home/taiga/.virtu
           |-15253 /home/taiga/.virtualenvs/taiga/bin/python3 /home/taiga/.virtu
           └-15254 /home/taiga/.virtualenvs/taiga/bin/python3 /home/taiga/.virtu
```

Соберемся, остались последние штрихи. Донастроим Nginx. Для начала удалим конфигурацию по умолчанию:

```
$ sudo rm /etc/nginx/conf.d/default.conf
```

или для некоторых случаев файл конфигурации по умолчанию может находиться в другом месте:

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

Создадим папку для логов:



```
$ mkdir -p ~/logs
```

Установка сертификатов Let's Encrypt

Поскольку мы будем использовать HTTPS для повышения безопасности нашего сайта.

Подробнее о SSL сертификатах, вы можете узнать из нашей [статьи](#). Итак, воспользуемся бесплатными сертификатами Let's Encrypt и клиентом Certbot.

```
$ sudo apt update
$ sudo apt install -y software-properties-common
$ sudo add-apt-repository universe
$ sudo add-apt-repository ppa:certbot/certbot
$ sudo apt update
$ sudo apt install -y certbot python-certbot-nginx
```

Теперь можно будет выпустить сертификат, где: **{YOUR_DOMAIN}** - Ваш домен, например, example.ru:

```
$ sudo certbot --nginx certonly -d {YOUR_DOMAIN}
```

Теперь создадим DH (Diffie Hellman) ключ:

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

Для самостоятельной установки можно воспользоваться инструкцией с официального сайта [Certbot](#).

```
$ sudo vi /etc/nginx/conf.d/taiga.conf
```

```
server {
    listen 80 default_server;
```



```

server_name _;
return 301 https://$server_name$request_uri;
}

server {
    listen 443 ssl default_server;
    server_name {YOUR_DOMAIN};

    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;

    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;
    }

    # 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;
}

# Events
location /events {
    proxy_pass http://127.0.0.1:8888/events;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
    proxy_connect_timeout 7d;
    proxy_send_timeout 7d;
    proxy_read_timeout 7d;
}

add_header Strict-Transport-Security "max-age=63072000; includeSubdomains; prel

ssl_certificate /etc/letsencrypt/live/{YOUR_DOMAIN}/fullchain.pem;
ssl_certificate_key /etc/letsencrypt/live/{YOUR_DOMAIN}/privkey.pem;
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-RS
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:ECDH
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;
}

```

Теперь проверяем конфигурацию:

```

$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok

```



```
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

Как видим - ошибок нет, а значит перезапускаем Nginx:

```
$ sudo systemctl restart nginx
```

На этом этапе уже можно пользоваться системой управления проектами Taiga (Тайга), но можно доустановить асинхронный режим. Дело в том, что по умолчанию Taiga выполняет все задачи в синхронном режиме, но некоторые задачи могут выполняться в асинхронном режиме, например, webhook или импорт/экспорт.

Установка режима Асинхронные задачи

Для начала проверим, что RabbitMQ сервер и Redis сервер установлены:

```
$ sudo apt-get install -y rabbitmq-server redis-server
```

Затем нам надо изменить строчку в файле `~/taiga-back/settings/local.py`:

```
CELERY_ENABLED = True
```

Теперь создаем файл `/etc/systemd/system/taiga_celery.service` для создания службы (демона):

```
$ sudo vi /etc/systemd/system/taiga_celery.service
```

```
[Unit]
Description=taiga_celery
After=network.target
```

```
[Service]
User=taiga
```



```
Environment=PYTHONUNBUFFERED=true
WorkingDirectory=/home/taiga/taiga-back
ExecStart=/home/taiga/.virtualenvs/taiga/bin/celery -A taiga worker --concurrency 4
Restart=always
RestartSec=3
ExecStop=/bin/kill -s TERM $MAINPID

[Install]
WantedBy=default.target
```

Теперь перезагружаем systemd, добавляем новую службу в автозапуск и запускаем:

```
$ sudo systemctl daemon-reload
$ sudo systemctl start taiga_celery
$ sudo systemctl enable taiga_celery
$ sudo systemctl restart taiga
```

На этом установка системы управления проектами Taiga закончена. На самом деле - нет. Мы не настроили firewall!

Настройка firewall

В ubuntu 18 используется ufw, для начала проверим его статус:

```
$ sudo ufw status
Status: inactive
```

Как видим - firewall не активен. Теперь посмотрим список заранее созданных шаблонов (в логике ufw это app от application - приложение):

```
$ sudo ufw app list
Available applications:
OpenSSH
```

Как видно, по умолчанию присутствует только настройки для ssh. Теперь шаблоны для 80 и 443 портов:



```
$ sudo vi /etc/ufw/applications.d/www
>[WWW]
title=WWW
description=World Wide Web.
ports=80/tcp
$ sudo vi /etc/ufw/applications.d/www-ssl
[WWWSSL]
title=WWW ssl
description=World Wide Web ssl.
ports=443/tcp
```

Проверим:

```
$ sudo ufw app list
Available applications:
  OpenSSH
  WWW
  WWWSSL
```

Теперь активируем

```
$ sudo ufw allow OpenSSH
Rules updated
Rules updated (v6)
$ sudo ufw allow WWW
Rules updated
Rules updated (v6)
$ sudo ufw allow WWWSSL
Rules updated
Rules updated (v6)
$ sudo ufw enable
```

Проверяем:

```
$ sudo ufw status
Status: active

To Action From
```

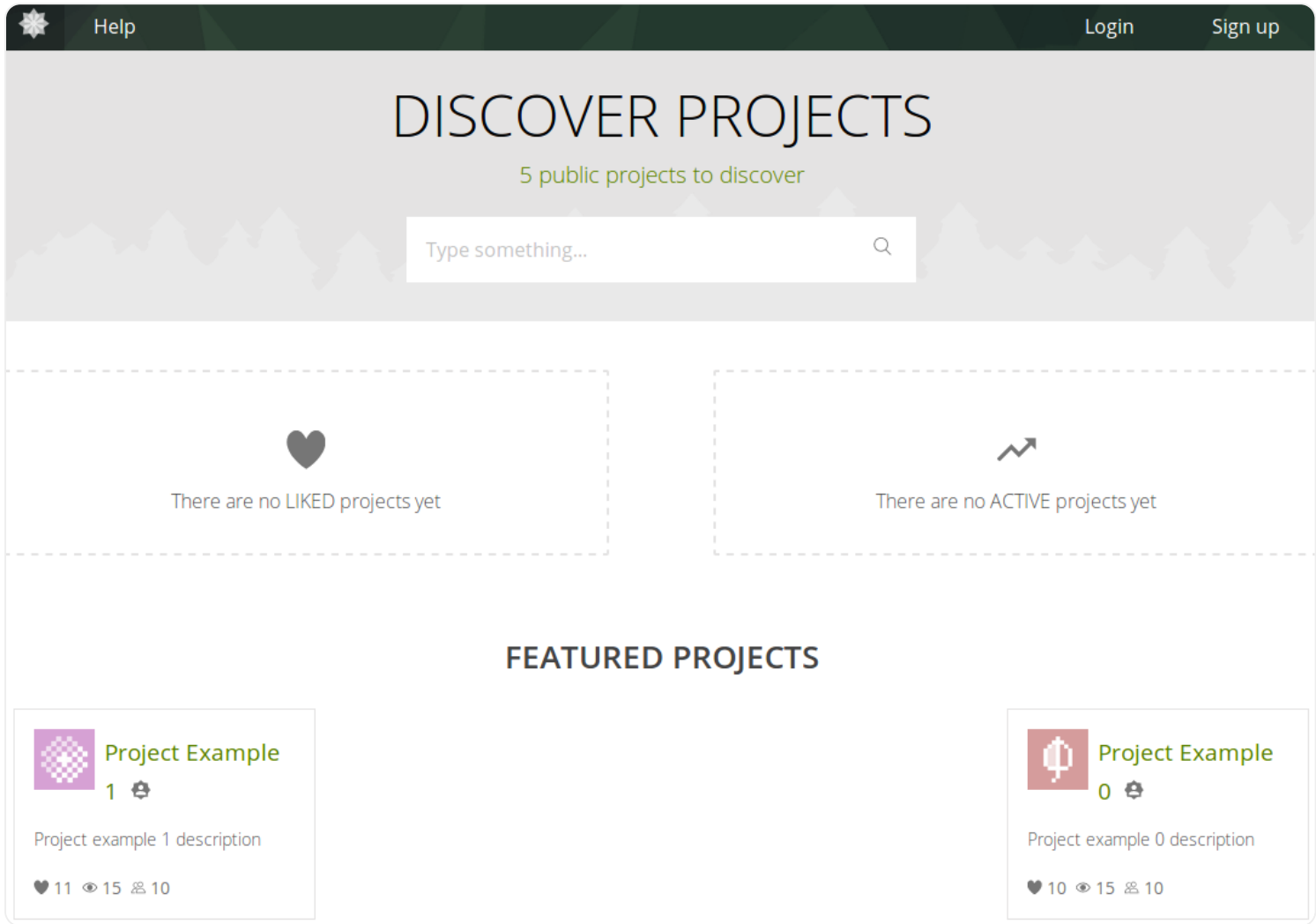


OpenSSH	ALLOW	Anywhere
WWW	ALLOW	Anywhere
WWWSSL	ALLOW	Anywhere
OpenSSH (v6)	ALLOW	Anywhere (v6)
WWW (v6)	ALLOW	Anywhere (v6)
WWWSSL (v6)	ALLOW	Anywhere (v6)

На этом теперь точно установка закончена.

Обзор Taiga.io

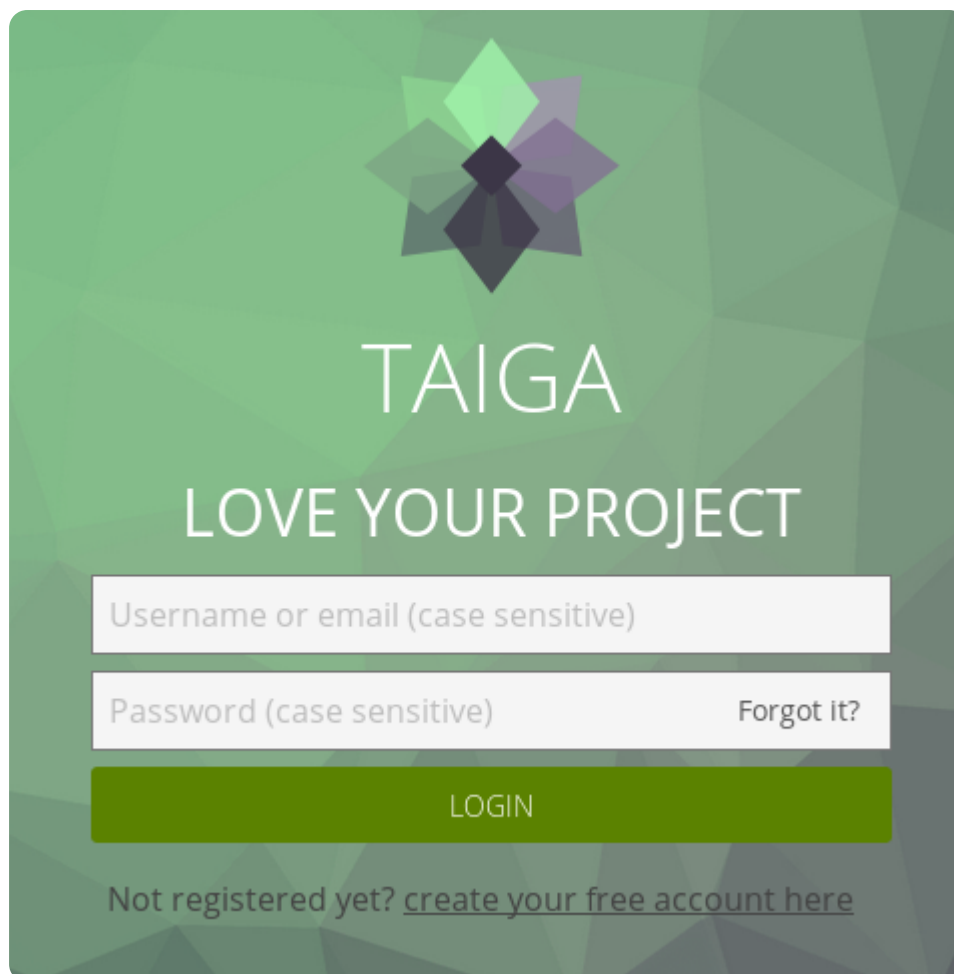
Теперь пришло время посмотреть на результаты своего труда. Заходим по нашему адресу https://{YOUR_DOMAIN}:



Не пугайтесь о наличии на изображении уже существующих данных. Если вспомним - в инструкции мы указали о возможности заполнить базу для демонстрации возможностей. Справа

вверху авторизуемся нажав кнопку "Login":





Помним пользователя **admin** и пароль **123123**? Вводим и не забываем сменить его в настройках пользователя. Там же можно выбрать язык интерфейса пользователя. После входа мы оказываемся на рабочем столе (Dashboard):



Help

Administrator

PROJECTS DASHBOARD

WORKING ON

Project Example 6 EPIC Ready

#48 Add tests for bulk operations

Project Example 6 ISSUE Needs Info

#46 Create the user model

Project Example 6 USER STORY New

#32 Migrate to Python 3 and milk a beautiful cow

Project Example 6 USER STORY Ready for test

#28 Fixing templates for Django 1.6.

Project Example 6 TASK Needs Info

#16 Lighttpd support

Project Example 6 TASK Needs Info

#11 Experimental: modular file types

Project Example 3 ISSUE Needs Info

#96 get_actions() does not check for 'delete_selected' in actions

Project Example 3 ISSUE Postponed

#92 Feature/improved image admin

Project Example 3 TASK New

Project Example 6

Project example 6 description

10
 15
 10

Project Example 5

Project example 5 description

9
 14
 10

Project Example 4

Project example 4 description

5
 12
 10

Project Example 3

Project example 3 description

6
 15
 10

Project Example 2

В верхней части выбираем проект (в данном примере выбран проект "Project example 0"), после чего мы оказываемся в управлении проектом. Посмотрим как выглядит Kanban в реализации Taiga.io:

Помощь

Administrator

PROJECT EXAMPLE 0 КАНБАН

НОВАЯ 4

Не назначен

#21 Lighttpd x-sendfile support

Очки 23.5

▲ 3 ● 1 ● 4

Не назначен

#30 Add setting to allow regular users to create folders at the root level.

Очки 25

● 1 ● 1 ● 3

Vanesa Torres

#36 get_actions() does not check for 'delete_selected' in actions

Очки 31

▲ 4 ● 4 ● 1 ● 2

Не назначен

#41 Implement the form

Очки 48

● 6 ● 1 ● 3

ГОТОВО 3

Virginia Castro

#6 Lighttpd x-sendfile support

Очки 26.5

▲ 9 ● 1 ● 1 ● 2

Francisco Gil

#32 Feature/improved image admin

Очки 19

▲ 8 ● 4 ● 1 ● 3

Не назначен

#39 Support for bulk actions

Очки 58.5

▲ 4 ● 3 ● 1 ● 4

В ПРОЦЕССЕ 5

Не назначен

#13 Support for bulk actions

Очки 110

▲ 6 ● 6 ● 1 ● 4

Angela Perez

#16 Add tests for bulk operations

Очки 17

▲ 5 ● 2 ● 1 ● 3

Miguel Molina

#35 Added file copying and processing of images (resizing)

Очки 43

▲ 3 ● 5 ● 1 ● 2

Vanesa Torres

#38 Create testsuite with matrix builds

Очки 30.5

▲ 1 ● 1 ● 1 ● 4

Mohamed Ortega

#40 Added file copying and processing of images (resizing)

Очки 10

▲ 7 ● 7 ● 1 ● 2

МОЖНО ПРОБ... 5

Begoña Flores

#1 Exception is thrown if trying to add a folder with existing name

Очки 44

▲ 1 ● 1 ● 1 ● 3

Angela Perez

#10 get_actions() does not check for 'delete_selected' in actions

Очки 53.5

▲ 6 ● 2 ● 1 ● 3

Mohamed Ortega

#24 Lighttpd x-sendfile support

Очки 46

▲ 7 ● 3 ● 1 ● 2

Virginia Castro

#34 Experimental: modular file types

Очки 18

▲ 4 ● 3 ● 1 ● 2

Mohamed Ortega

#37 Create the html template

Очки 23.5

▲ 4 ● 5 ● 1 ● 3

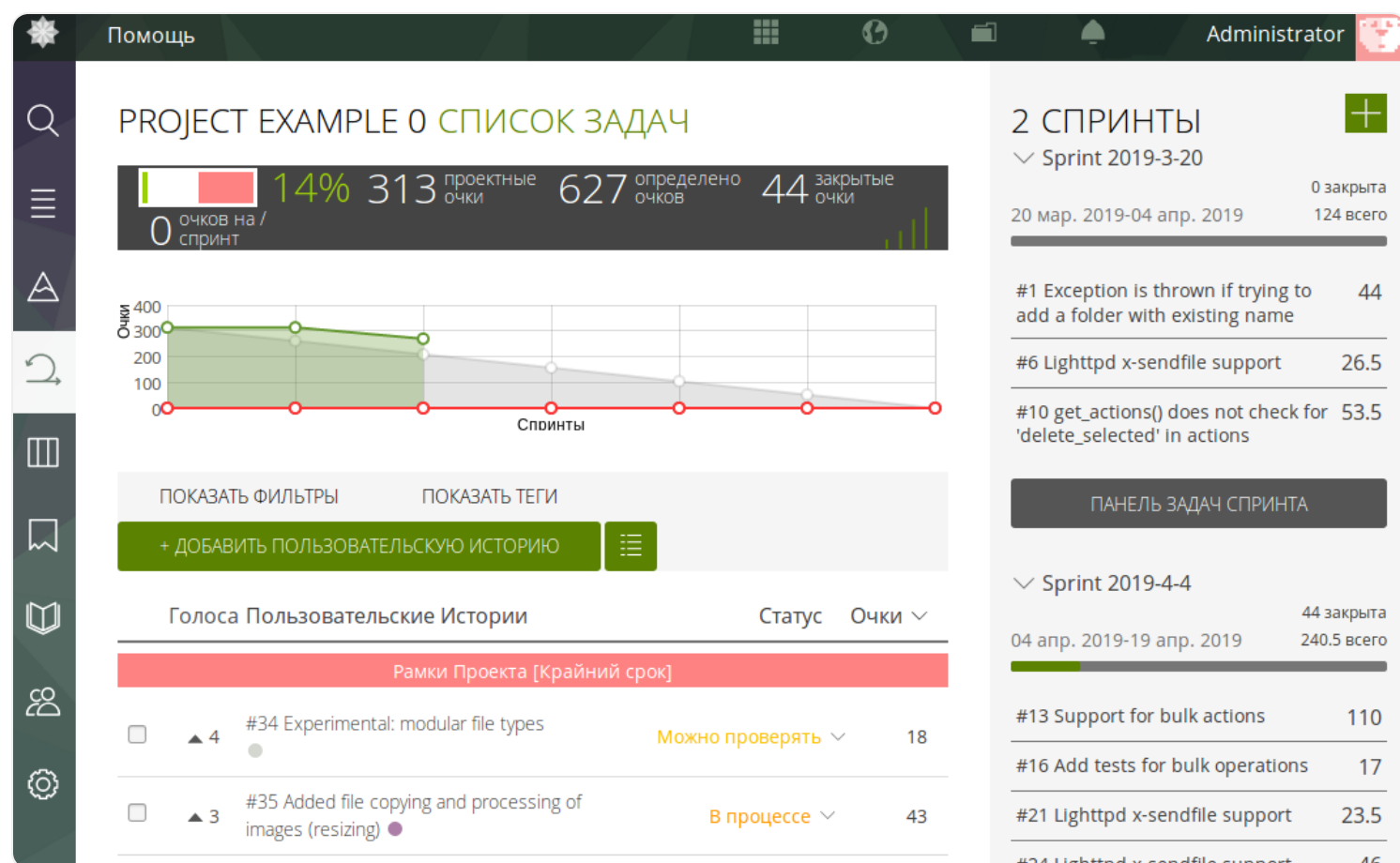
ЗАВЕРШЕНА

АРХИВИРОВАНА

Пользовательские истории в этом статусе скрыты по умолчанию



Теперь посмотрим как будет выглядеть Scrum:



В каждом проекте можно добавлять или удалять пользователей, распределять роли:



Помощь

Администратор

ФИЛЬТРЫ

Искать по имени

ВСЕ

ЮЗАБИЛИТИ

ДИЗАЙНЕР

ФРОНТЕНД РАЗРАБОТЧИК

БЭКЕНД РАЗРАБОТЧИК

ВЛАДЕЛЕЦ ПРОДУКТА

ЗАИНТЕРЕСОВАННАЯ СТОРОНА

PROJECT EXAMPLE 0 КОМАНДА

Мр. Употребляющий
Вульф

Сервантес
иокаин

Охотник
за
ошибками

Ночная
смена

Могущество

Administrator
Back

Покинуть проект

6

Команда Все

Angela Perez
Stakeholder

4

Begoña Flores
UX

8

Catalina
Fernandez
Back

1

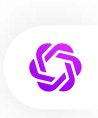
Enrique Crespo
Product Owner

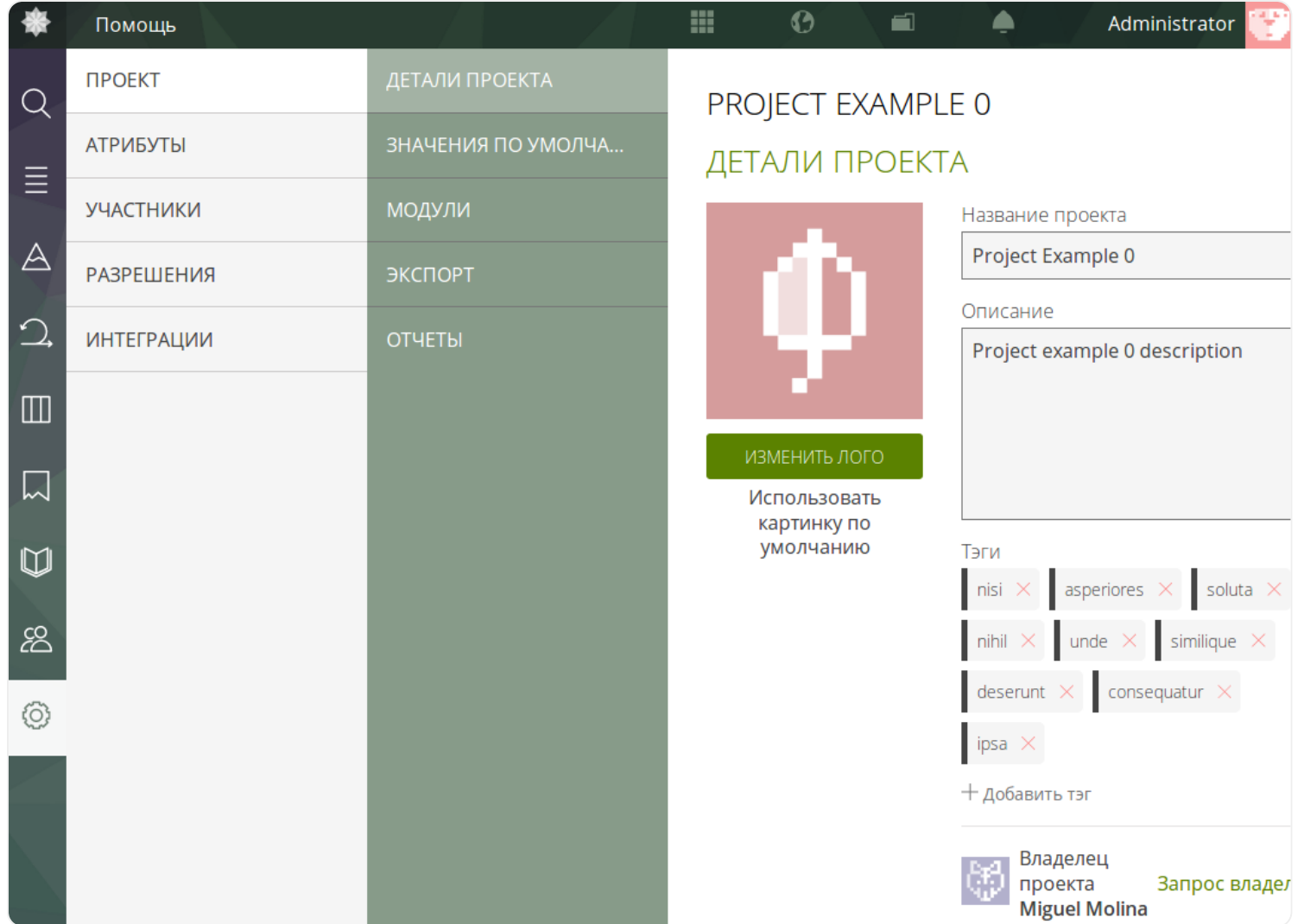
1

Francisco Gil
Front

5

Конечно же есть и различные настройки самого проекта:





Если Вы знакомы с методологией Agile, то слышали об Epics. Следующее изображение показывает как выглядит это в Taiga.io:



Помощь

Administrator

🔍

☰

🏠

↶

📅

🔖

📖

👤

⚙️

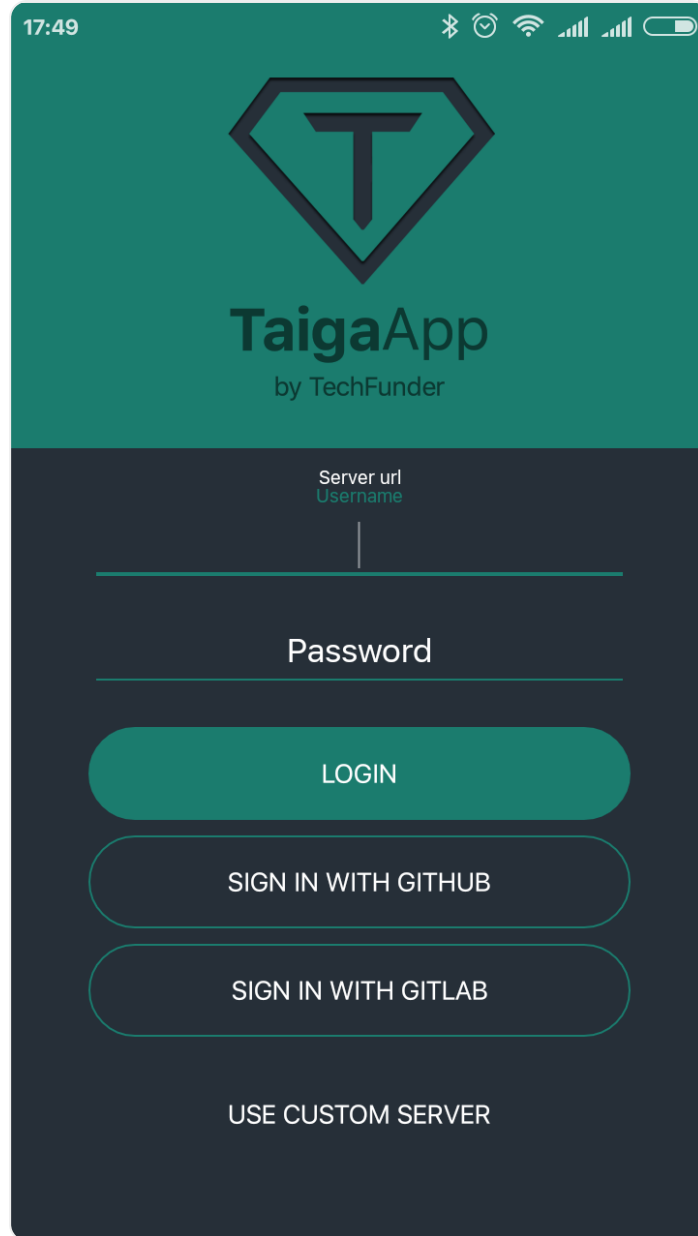
PROJECT EXAMPLE 0 ЭПИКИ

+ ДОБАВИТЬ ЭПИК

Голоса	Имя	Проект	Спринт	Назначен	Статус	Прогресс
▲ 3	#64 Added file copying and processing of images (resizing) ЭПИК ^				Можно проверять ▾	<div><div></div></div>
▲ 9	#65 Experimental: modular file types ЭПИК ▾				Новая ▾	<div><div></div></div>
▲ 4	#37 Create the html template ●●●●○				Можно проверять	<div><div></div></div>
▲ 3	#35 Added file copying and processing of images (resizing) ●				В процессе	<div><div></div></div>
▲ 5	#16 Add tests for bulk operations ●●●○		Sprint 2019-4-4		В процессе	<div><div></div></div>
▲ 1	#66 Added file copying and processing of images (resizing) ЭПИК ^				Можно проверять ▾	<div><div></div></div>

На этом удобства использования системы управления проектами Taiga не заканчивается. Здесь присутствует ПО для мобильных платформ, в частности в кратце рассмотрим приложения для устройств на базе [OC Android](#):






Данное приложение имеет платные функции, но для большинства хватит и бесплатных возможностей. Для подключения к своему серверу выбираем **use custom server** и вводим наши данные:



17:50

←



TaigaApp
by TechFunder

Server url

Username

admin

Password

.....

LOGIN

После входа мы также видим рабочий стол (dashboard):



WORKING ON

Еще статьи из блога



Issue Требуется подробности
#46 Create the user model, Project Example 6

Установка и настройка
Confluence на
собственный сервер

7 ОКТЯБРЯ 2019

Установка и настройка
системы управления
проектами Taiga

23 МАЯ 2019

Что такое SSL
сертификат и для чего
он нужен

12 НОЯБРЯ 2018

Многосайтовость 1С-
Битрикс

4 СЕНТЯБРЯ 2020

Битрикс Сайты 24

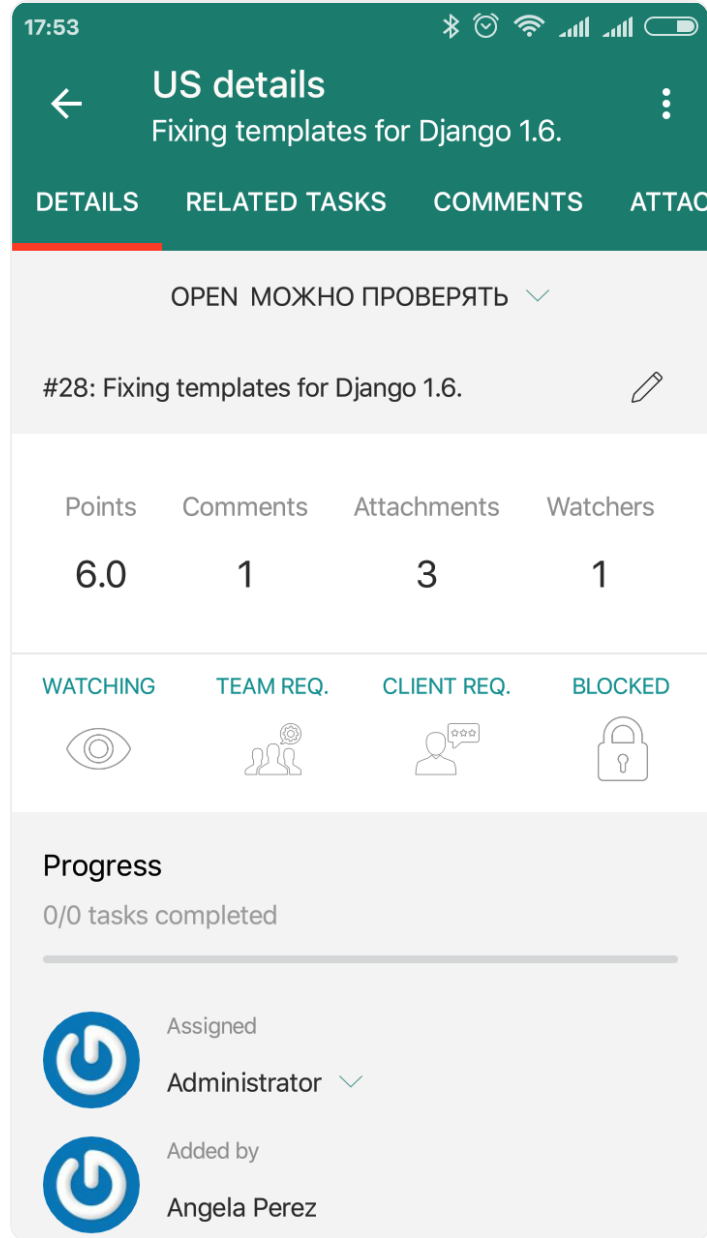
11 НОЯБРЯ 2021

Хостинговые услуги Сайты Инфраструктура Битрикс

Компания Дата-центры Документы Новости Партнёры Вакансии
Партнёрская программа Отзывы

sync|web





С мобильных устройств можно заходить через web-интерфейс в Taiga, но, к сожалению, не все браузеры всегда адекватно обрабатывают некоторые возможности. Например, в Kanban перенос задач не всегда работает во многих браузерах. В конце концов - Taiga имеет собственное API, позволяющее использовать весь функционал.

Заключение

Мы с Вами рассмотрели ПО системы управления проектами Taiga работающая с Kanban и Scrum семейства Agile методологии. Конечно, полноценно научиться работать с данной системой - дело не минутное, да и в этой статье мы не затрагивали эту тему, но если представить открывающиеся возможности после изучения и внедрения данного ПО- то это несомненно принесёт пользу Вашему бизнесу и будущим проектам. Главным примером являемся мы сами. Наша компания Syncweb активно и успешно использует её в своих внутренних проектах.



К тому же при использовании на своих серверах Taiga бесплатна. Наличие различных возможностей интеграции, API, активная разработка(проект не брошен, регулярно выходят обновления), активное сообщество только даёт плюсы в сторону использования Taiga. К главным минусам мы бы отнесли сложность в установке и обновлении и малом количестве русскоязычных ресурсов по данному ПО, что несколько повышает порог знаний системных администраторов для установки и внедрения в организацию, но мы надеемся, что наша инструкция поможет снизить данное требование. С нами было ведь не сложно, правда?

Надежные VDS сервера для установки Taiga



Смотреть



