Первоначальная настройка.

1. Настройка SSO (Используется библиотека passport.js: <https://www.passportjs.org/packages/passport-oauth2/> )
   1. Все необходимые для работы токены/URL добавить в ./auth/.env по аналогии с ./auth/.env.example. Токены, которые могут быть вынесены в эти переменные (примеры для Яндекса можно найти в .env.example): CLIENT\_ID, CLIENT\_SECRET, AUTHORIZATION\_URL, TOKEN\_URL  
      Необязательно ограничиваться только этими переменными, при необходимости можно вынести любые, для обращения к ним использовать process.env.VARIABLE\_NAME
   2. Скопировать файл ./auth/src/middleware/passport/customYandexStrategy.ts в ту же папку (./auth/src/middleware/passport) и переименовать классы или работать непосредственно с существующими.
   3. Заменить this.name = “yandex” на необходимое (Далее не используется, можно выбрать любое. Является названием стратегии по умолчанию, если название не указано явно)
   4. Обратить внимание на строку this.\_oauth2.setAccessTokenName(“oauth\_name”) и комментарии над ней. Если в ней нет необходимости, удалить.
   5. Заменить функцию userProfile и вызов userService.findOrCreate.   
      userProfile делает запрос для получения данных о пользователе и возвращает объект, который далее передаётся в колбэк (вызывающий userService) как переменная profile. В userService.findOrCreate передать один аргумент - объект с двумя свойствами:

* Email
* name (полное ФИО)
  1. В файле ./auth/src/middleware/passport/passport.ts заменить  
     import customYandexStrategy from “./customYandexStrategy”  
     на имена созданных стратегии и файла
  2. Заменить  
      passport.use(“sso-strategy”, customYandexStrategy);

на   
 passport.use(“sso-strategy”, \*strategyName\*);

1. Установка nginx
   1. sudo apt update
   2. sudo apt install nginx
   3. Изменить default.conf (в ./etc/nginx/conf.d/default.conf) на default.conf из корня репозитория
   4. Изменить (в default.conf) server\_name на необходимое доменное имя
2. Установка и настройка Certbot
   1. sudo apt install certbot python3-certbot-nginx
   2. sudo certbot --nginx -d example.com -d www.example.com (example.com заменить на необходимое доменное имя, указанное в default.conf)
3. Заполнение переменных среды

Создать и заполнить файлы:

* **./.env** - по аналогии с ./.env.example
* **./client./.env.local** - по аналогии с ./client/.env.example
* **./server/.env** - по аналогии с ./server/.env.example
* **./auth/.env** - по аналогии с ./auth/.env.example
* **./strapi/.env** - по аналогии с ./strapi/.env.example

Если используется дамп БД, то необходимо ввести значения API-токенов, используемых ранее (для исходного дампа значения указаны внизу этого документа).

Если не используется дамп БД (неизвестны API токены), то следует написать произвольные, которые затем следует создать (с введёнными ранее произвольными значениями) при помощи админ-панели strapi.

API-токены нужны для взаимодействия процессов “server” и “strapi”, их отсутствие не будет мешать работе админ-панели процесса “strapi”.

Для упрощения можно во всех API-токенах указать один и тот же, а в админ-панели strapi создать его с полными правами на доступ ко всем сущностям.

К API-токенам относятся:

* PROJECTS\_TOKEN
* USER\_TOKEN
* REQUESTS\_TOKEN
* PLATFORM\_TOKEN
* FORM\_TOKEN
* UPLOAD\_TOKEN
* CATEGORY\_TOKEN

1. Импорт БД

В файлах 01\_structure.sql и 02\_contents.sql находится дамп базы данных с уже заранее сгенерированными API токенами и данными по некоторым проектам и пользователям.  
  
Их следует скопировать в **./postgres/01\_structure.sql** и **./postgres/02\_structure.sql**  
Также должны быть указаны в volumes в ./docker-compose.yaml для сервиса “postgres” (по умолчанию уже указаны).   
  
Далее все данные по базе данных будут автоматически сохраняться в ./postgres/data, дампы нужно будет удалить из volumes в docker-compose.yaml

1. Установка Docker Compose Plugin
   1. sudo apt update
   2. sudo apt install docker-compose-plugin
2. Запуск (в папке с docker-compose.yaml)
   1. docker compose build
   2. docker compose up -d

Перезапуск одного процесса (на примере server).

Имя процесса (здесь server) берётся из docker-compose.yaml

1. Загрузить файлы с обновлённым исходным кодом
2. docker compose down server
3. docker compose build server
4. docker compose up server

Перезапуск всех процессов. (БД уже существует).

1. Убрать дампы БД из volumes для процесса postgres (01\_structure.sql, 02\_contents.sql)
2. docker compose down
3. docker compose build
4. docker compose up

Значения API-токенов в дампе БД

PROJECTS\_TOKEN=dfcea4fcfc6c4b40a4772700af3cdcbcc8866f4254ea12b7c7ceb6de5e1a6b1c605bd510f148025462d50f67802fb739112aa1232a2ff66fd48b5fde234101d37be45ac62afb248076ad622b38b9c2051f8dd48cc221a198f7f7a9dd6be871ab9bf90a9320ca63889f0bb2bd5ab6c44d8086b6313bbf5cfff574673f1bbd5d4b

USER\_TOKEN=fcf56c55d1def8f1069fb16bad1bad1f59d77127caaefe0b7e3ff8ae213a25b1468b4ab367b707c553ffbaa935769d85034a3a7326e834d814b87359510be7635039ce8d118e226e4cb847c8d6261cb08a1873c449c655d6f968ec921b0eb92d3c8fbebdd8785ee551c45a0b04040003b0e11f303bd736894103402c7dc90a17

REQUESTS\_TOKEN=b55644b09eb9b37ef87df16a8a8984654b1eead4d7f839e214f510b965880592498eaf33c533f734b918eb8066e65e84cbdc75594696835b865a3ffd0c81ae87bd84da0a104550da15273280b0bdd637381171014178b2fe6dd25d26bb758551a83b6160ca73e55fe64a5c2c6633aab09fcab05f2bab73a20d1e669e75e4782b

PLATFORM\_TOKEN=e994f9edd76abe30e058a7e1754a9b066ba0a9370ab6fdaa7d952f8d7e228f93eb84bb7dff63467547aa67b6380dca56001e61eaa900164370a12dc3da062aa947f829a675024dc634f57b807f543ff1f5c019ddc5526cc647ef684e41258a8a5a63815d46b5f9151b6621576908a78c3b769e6942dc3f8e251fee19da1d174c

FORM\_TOKEN=ca6e40b3f5f1ed0a477e16defe17362093c4301e7b9d86acaf1abc8143cc7160a64b5b1112fccaf6e8280c517cd7c676294f6f9a0efe49320a7ccd7963af766fcd28c4a25cf606304cbd91d095de8fa1c1e138a99367c843378bf756002f4b106294e8e7e4c507ae4d0593edd8fbb826a3376860e8af141e91445d941d1ecb9e

UPLOAD\_TOKEN=c11e60c9af45d6d33705163b917052417e925848109b84617aba867c201cc5e04c3e3cec9b4ae67e2ccb29c5d6397ecb8eeef69f6adef7a1a1147f796aa9f10699985c397cca499e2b15437976cbd7bf179b9182c58d14dfe89a794ac6adeffe99e1d514ab35f8471e19912c1f3ffad3d4900d2d5eb6ee173bcaeee88bdc3c52

CATEGORY\_TOKEN=19750355984891d34bdbf1b28b3319d6f3b718883b95a40fc73651ae399017852a42ca9aa02527cfb7da0a40646f27ed387ce6cac6d39f4be62f5ac44dbdabaf3fa09421a1ec517595f8ff5b23170245fab85acd287ecd8895c71628ede468fc6be01234801cc183f4b945c0c6d6c67432abf48240158a3e91e25adb26536050

Создание API-токенов в админ-панели Strapi

Права, которые нужно дать каждому токену на сущности:

| Имя токена | Content-Type | Права |
| --- | --- | --- |
| CategoryToken | Category | find, update, delete |
| Project | find, findOne |
| Tag | find, findOne |
| FormToken | Form | find, findOne, create, update, delete |
| Student | find, findOne, update |
| PlatformToken | Platform | find, findOne, create, update, delete |
| ProjectsToken | Platform | find, findOne |
| Project | find, findOne, create, update, delete |
| Request | find, findOne |
| Student | find, findOne |
| Tag | find, findOne, create, update, delete |
| Team | find, findOne |
| User-in-team | find, findOne |
| Upload | find, findOne, destroy, upload |
| RequestsToken | Project | find, findOne, create, update, delete |
| Request | find, findOne, create, update, delete |
| Student | find, findOne |
| Team | find, findOne, create, update, delete |
| User-in-team | find, findOne |
| UploadToken | Upload | find, findOne, destroy, upload |
| UserToken | Form | find, findOne |
| Student | find, findOne, create, update, delete |
| Team | find, findOne, create, update, delete |
| User-in-team | find, findOne, create, update, delete |