Branching strategy



**Guideline for developers**

1. Checkout a new branch according git flow naming conventions: feature/super-feature-N or bugfix/super-fix-N where N is the number of CTP **task** from **YouTrack**
2. After you’ve done your coding and testing create a **pull request** on our BitBucket from **your branch** to current **release branch** (if it doesn’t exist ask @aleksei\_badyaev (Unlicensed) to create one)
3. Update **YouTrack task** and pass it to code-review (put a **link** to the **pull request** in the task description)

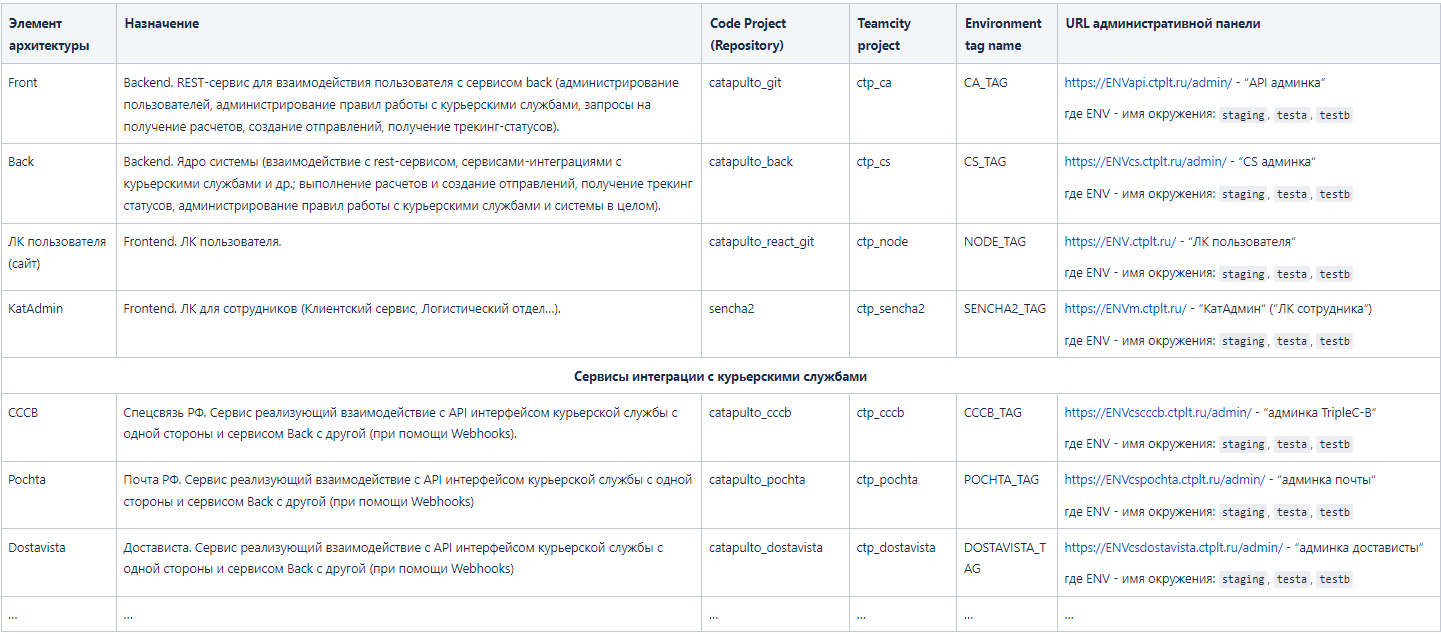
Тестирование задач производится в окружениях staging, testN.

## Основные элементы архитектуры системы

Для настройки окружения и проведения тестирования требуется понимать архитектуру продукта и знать следующую таблицу соответствий имен:

**…ctplt…** - это staging

**…catapulta…** - это прод



* **Test a b c d** - здесь нет трекинга. Но отправления все равно нужно отменять
* **dev a** – для разрабов
* **dev b** – для разрабов
* **dev demo** – здесь крутится демо версия прилаги
* **staging** - наиболее функциональный и приближен к проду. Если ты его забираешь, то отписываешь в Rocket Chat > Environment : taking staging

**Доступ к серверу окружения**

Для тестирования задачи необходимо иметь доступ по протоколу SSH (22 порт) к серверу окружения. Доступ предоставляет DevOps-инженер по согласованию с руководителем команды разработки и тестирования (TeamLead). Для получения доступа необходимо сгенерировать пару публичный\закрытый ключ ssh-rsa и передать DevOps инженеру открытый ключ

1. Подключаемся за юзера (которому закинут ключ подключения) к серверу. В Putty в настройках SSH закидываем ppk ключ для подключения. После подключаемся к серверу (по сути не важно какой testN мы введем, так как это просто элайс (ссылка). Если мы введем testa или testb, все равно перекинет на один и тот же сервер

aivanov@testd.ctplt.ru - тестовые

aivanov@staging.ctplt.ru - стейджинг

1. Когда подключимся, то логинимся под админом

sudo su - ctpadmin

1. На сервере будут папки - каждая папка это окружение (a b c d). Открываем нужное окружение. В данной директории располагаются файлы с локальными настройками всех приложений. Например:

* **catapulto/settings\_local.py** - файл для настройки сервиса front)
* файлы с описанием docker-compose сервисов (**yaml**-файлы),
* локальные конфигурации веб-сервера nginx (в директории **nginx/conf.d**/)
* файл **.env** с именами тегов докер образов, которые будут использоваться при разворачивании проекта.

cd home/ctpadmin/test

Переходим на нужное окружение (мое окружение - d)

cd d

1. Для доступа к личному кабинету пользователя (ЛК клиента и ЛК сотрудника) и административной панели API сервиса Front требуется, находясь на сервере в домашней директории окружения под пользователем ctpadmin (ctpadmin@dev:~/test/a$), создать супер-пользовател

./run.sh exec front python manage.py createsuperuser --username username --email [username@catapulto.ru](mailto:username@catapulto.ru)

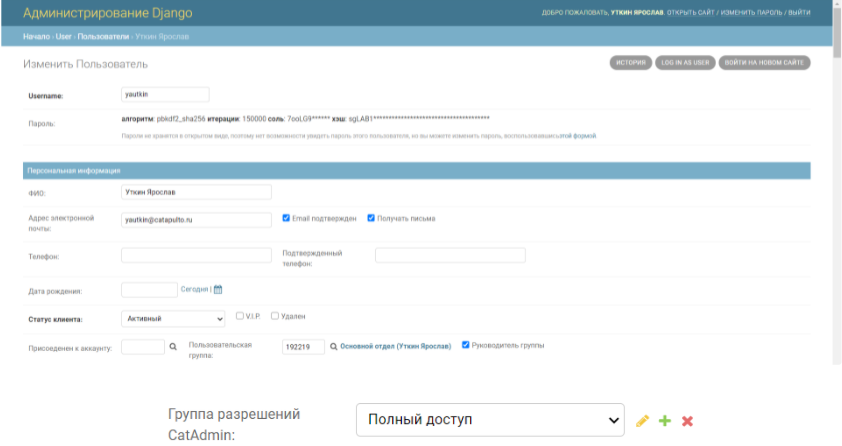
*Для доступа к административной панели CS сервиса Back и административным панелям модулей интеграции с курьерскими службами:*

* ./run.sh exec back python manage.py createsuperuser --username username --email username@catapulto.ru
* ./run.sh exec cccb python manage.py createsuperuser --username username --email username@catapulto.ru
* ./run.sh exec pochta python manage.py createsuperuser --username username --email username@catapulto.ru

Логин и Email делаем одинаковыми для простоты

Пароль qwerty123

1. Затем требуется авторизоваться в административной панели API сервиса Front настраиваемого окружения, используя username и пароль c предыдущего шага (пример URL адреса https://testaapi.ctplt.ru/admin/). Открыть модель “Пользователи”, используя форму поиска найти username созданный на предыдущем шаге, указать ФИО, отметить Email подтвержден, указать статус клиента “Активный“, в поле “Группа разрешений CatAdmin“ выбрать “Полный доступ“. (для того что бы была возможность зайти в CatAdmin), указать что доступен наложенный платеж



В CatAdmin заходим под тем же юзером что и в ЛК. Мы этим юзером сможем зайти только в то окружение, где мы его создали (например test d)

Если нам нужно зайти на другое тестовое окружение (test a, test b) то нужно создавать юзера для этого тестового окружения

Если нам нужно зайти на другое окружение (staging, dev ,…) то нужно заходить на сервер с нужным окружением и там создавать юзера

Если нам нужно зайти на Back Backend, или на микросервис (часть КС реализована через микросервисы), то так же нужно зайти на сервер по SSH и создать там юзера, а после уже зайти на веб интерфейс сервиса используя креды. Если это Back Backend, то **testdcs.ctplt.ru/admin/**

Изменения на тестовом окружении не влияют на продакшн окружение,но нельзя удалять телефоны или email. Так как пошлется оповещение

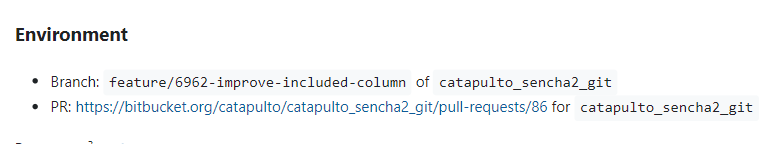
Но если создать отправление, то оно уйдет в курьерку

**Настройка Окружения**

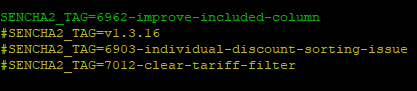
1. Изменяем настройки конфигурации откуда качать какие билды (для обновления)

Версии (тэги образов Docker-контейнеров в Nexus) программных компонент указываются в файле .env, находящемся в папке окружения (например, папка TESTA-окружения находится в папке **/home/ctpadmin/test/N**).

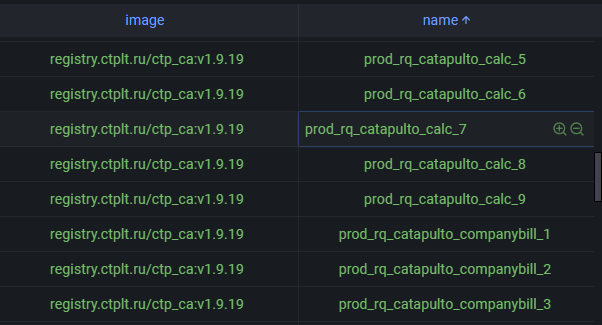
В нем мы меняем версию тэга, которую мы можем посмотреть в release (для той задачи которую мы делаем - feature/ветка). Мы прописываем версию, а в yaml файлах указан путь откуда качать



Так как в задаче были изменения с CatAdmin, то меняем тэг Cat Admin. Закоменчиваем стабильную ветку, и создаем свою, где прописываем путь



Остальные ветки ставить стабильные - то что на проде (их можно посмотреть через Grafana > Urgent overview catapulto). Смотрим версию



или в TeamCity. Последняя версия будет билд с числом. Так же смотрим на дату (так как релизы каждую неделю, дата должна быть начиная с даты прошлого релиза)



1. После внесения версий в файл тэгов окружения необходимо обновить конфигурацию окружения, выполнив в папке окружения серию команд:
   1. ./run.sh pull front back node dostavista catadmin …- Скачиваем ветку
   2. ./run.sh up -d - Перезапускаем сервисы

**Если у нас Stage**

* 1. ./run.sh exec nginx nginx -s reload
  2. docker exec nginx nginx -s reload

1. Проверить миграцию (обновление) базы данных. Например что если новая задача подрзумевают добавление нового поля, смена значения, ..., то нужно обновить и БД

* ./run.sh exec {проект} python manage.py migrate
* Front
* Back
* Eshop
* Dostavista

Для CatAdmin , Node нет миграции

Как зайти за другого пользователя (например что бы не создавать отправления, если нам нужно что то проверить с ними. Можно зайти за юзера у кого уже есть созданные отправления)

1. Залогиниться в **API админке > пользователи** и найти пользователя с отправками. Выбираем крупных юр лиц (Дайсон, Щелково агрохим, Арциус)
2. Переходим в карточку пользователя и нажимаем "войти на новом сайте"

Если разраб внес изменения в код (исправил найденный дефект), то он вливает изменения в эту же ветку, по этому нам нужно будет просто запулить изменения (шаг 2), и не вносить изменения в тэги

Может быть такое, что разраб разрабатал, пофиксил что-то, и мне на тестирование пришла задача через какое то время, когда другие компоненты обновились. И если я в .env файле прописываю новые версии других компонентов, но разработаная фича/ пофикшенный баг висит на старой бранче (какой то компонент), то может не завестись функциональность. Для этого просить разраба подлить свежие изменения в его ветку. Что бы проверить точно ли баг связан с изменениями – нужно посмотреть когда разраб подлил свои изменения (создал ветку). Так как у нас релизы каждую неделю, то если он сделал задачу более недели назад, значит ветка точно устарела. Но не надо всегда бегать к разрабу и просить подлить, нужно проверить а связаны ли новые изменения компонентов (микросервисов) с теми, которые сделал разраб (в другом микросервисе). Если эти два микросервиса общаются между собой, то нужно подлить новые изменения в ветку разраба

Если была проблема с билдером, то нужно просто пересобрать ветку (нажать Run около названия нужной ветки)

