В данном проекте мы используем двухэтапную схему работы над задачами (выполнение-проверка).

Процесс выполнения задачи:

• Перед началом выполнения задачи тщательно продумать, что не понятно, какая нужна информация, все ли формулировки ясны и задать необходимые вопросы коллегам.

• В процессе выполнения коммитить результаты в отдельную ветку (**НомерЗадачи***краткое*название\_задачи), фиксировать полезную информацию в самой задаче.

• Когда задача готова:

• создать pull-request из ветки задачи в ветку master;

• добавить комментарий к pull-request’у, в котором указать сценарий проверки (что нужно сделать с проектом, чтобы убедиться что задача выполнена) и ссылку на issue;

• добавить pull-request’у метку (label) ready*for*check.

• Варианты после проверки:

• Нет замечаний - PR мержится в master.

• Есть небольшие замечания - PR мержится, но создается отдельная задача для исправления замечаний.

• Есть существенные замечания - PR не мержится, с задачи снимается метка **ready*for*check**, замечания фиксируются в задаче в виде комментария, который необходимо исправить (в том же PR) и повторно отметить задачу как **ready*for*check**.

• issue закрываются только проверяющим.  
Welcome to the report\_generator wiki!

Создайте в */var/www/.ssh/* файл *id*rsa\_. Перенесите туда свой закрытый ключ. Скорее всего ваш закрытый ключ будет находиться в *~/.ssh*, если вы раннее не меняли маршрут для ssh ключа

Сделайте *id*rsa\_ c доступом 600 и для www-data:*chown www-data:www-data id\_rsa  
chmod 600 id\_rsa*

От www-data сгенерируйте файл *know\_hosts*, используя:*sudo su -s /bin/bash www-data  
ssh -T git@github.com*

Вернитесь в пользователь *root*:*exit*

*service apache2 restart*

Закрытый ключ не должен запрашивать пароля  
 Тут будет сравнение с аналогами, например pandoc, and so on

Запуск проекта

Перейдите в *report\_generator/web-service/src*

Сгенерируйте папку *python3.6 -m venv venv*

Активация виртуального окружения *source venv/bin/activate*

*pip3.6 install -r ../scripts/requirements.txt*

Сам запуска сайта *python3.6 start\_service.py*

Основной принцип

Wiki-страницы должны быть оформлены по правилам markdown разметки, например:  
В файле было написано так: *try:  
 do\_smt()  
 except:  
 it\_is\_bad()*Данная конструкция превратится при отображении в:  
 try:  
 do*smt()  
 except:  
 it*is\_bad()

**Вывод:** Для сохранения форматирования из файла используйте соответствующие возможности markdown('''*some*text\_```)

Инструкция

**Добавьте в колабараторы PRsender**

в папке src создать файл *oauth.txt* с токеном для гитхаба

в папке src создать файл *github\_data.txt<client\_ID>  
<client\_secret>*

*./build\_and\_run\_docker.sh*

Зайти в докер: *docker exec -it report\_generator\_doc bash*

Вставить в файл */var/www/.ssh/id\_rsa* ssh ключ

*bash /report\_generator/scripts/init.sh* + согласиться на добавление know\_hosts

Уже на вашей машине в */etc/hosts* прописать *127.0.0.1 report-generator*

Перейти на *report-generator:8077*

Список аргументов

• *general\_font* - шрифт обычного текста

• *general\_size* - кегль обычного текста

• *code\_font* - шрифт кода

• *code\_size* - кегль кода

• *for\_h1* - кегль для заголовка 1 уровня

• *for\_h2* - кегль для заг. 2 ур.

• *for\_h3* - кегль для заг. 3 ур.

• *for\_h4* - кегль для заг. 4 ур.

• *for\_h5* - кегль для заг. 5 ур.

• *for\_h6* - кегль для заг. 6 ур.

• *teacher* - ФИО преподавателя

• *student* - ФИО студента

• *number\_group* - номер группы

• *theme* - тема работы

• *discipline* - предмет

• *cathedra* - кафедра

• *min\_pages* - мин. кол-во страниц

• *date\_start* - дата выдачи задания

• *date\_finish* дата окончания

• *date\_defend* - дата защиты

• *repo\_name* - ccылка на репозиторий (ssh)

• *wiki\_name* - ccылка на wiki (https)

• *branch\_name* - название ветки

все пробелы заменять на символ *\_*

Пример ссылки

*http://<domen>?general\_font=Times\_New\_Roman&general\_size=14&code\_font=Consolas&code\_size=10&for\_h1=20&for\_h2=18&for\_h3=16&for\_h4=14&for\_h5=12&for\_h6=10*

http://report-generator.moevm.info/?general*font=Times*New*Roman&general*size=14&code*font=Consolas&code*size=10&for*h1=16&for*h2=14&for*h3=13&for*h4=12&for*h5=11&for*h6=10&teacher=Чайка*К.В.&discipline=Программирование&cathedra=МО*ЭВМ&type=lr&repo*name=git@github.com:OSLL/report*generator.git&wiki*name=https://github.com/OSLL/report*generator.wiki.git  
К каждому блоку комментариев прикреплен коммит, при котором были оставлены комментарии. Предполагаем, что все исправления комментариев были совершенны в следующем коммите. Поэтому сначала парсим комментарии в пул реквесте через github api, после высчитываем следующий коммит и берем дифф коммитов, полученный дифф - есть исправления комментариев

Для работы программы вам необходимо иметь

• Репозиторий, в котором у вас есть comitt-доступ.

• *wiki*, можно брать *wiki* даже **не** с вашего репозитория.

Перед запуском вам необходимо настроить конфигурационный файл(*settings.json*)*settings.json*

Краткая настройка

Основные поля:

• *student* - ФИО студента.

• *teacher* - ФИО преподавателя.

• *download* - список имен файлов, которые будут добавлены в отчет в приложение.

• *PDF* - *bool* переменная, отвечающая за формат файла( *.pdf - True, .docx - False*).

• *pages\_of\_wiki* - список страниц из *wiki*, которые надо включить в отчет.

Пример как протестировать данную программу

Если нет репозитория, в котором вы можете сделать *commit*, то можете сделать *fork*[репозитория](https://github.com/light5551/test_gen)

Запустите программу как указано [здесь](https://github.com/OSLL/report_generator#report_generator)

Для тестирования wiki можно брать [отсюда](https://github.com/light5551/test_gen/wiki), также для примера там будет ветка **test\_branch**

Также если программа выдала что-то похожее на *Unoconv error: Command '['/usr/bin/python3', '/usr/bin/unoconv', '-f', 'pdf', 'mytestproject/ready\_project.docx']' returned non-zero exit status 251.* , то просто запустите программу заново, это ошибка значит, что утилита *unoconv* не смогла сконвертировать .docx в .pdf

Инструмент для генерации отчетов

Аннотация

Статья посвящена актуальным на сегодняшний день проблемам: хранение отчетов на кафедрах вуз’ов, большая трата времени людей, которые соответственно и пишут отчеты. Также в статье есть и сравнение с иными путями генерации отчетов. Была реализована автоматизированная система генерации сложных отчетов с гибкими настройками, которая на основе данных, полученных с веб-сервиса github.com, генерирует отчет. Также статья будет интересна людям, которым требуется писать отчеты, которые бы удовлетворяли некоторым требованиям к оформлению. В статье описан способ эффективного и быстрого написания отчета, который будет удовлетворять требованиям, требуемые к нему. Эта статья представляет интерес в первую очередь для студентов, которые тратят большое количество времени не только на выполнение самой работы, но и на написание отчетов по выполненной работе с обеспечением форматирования, которое тоже тратит большое количество времени. Также статья интересна преподавателям, которые должны проверять отчеты и предоставлять их на проверку комиссиям. Была произведена сравнительная характеристика различных путей генерации отчетов. Автором предложена реализация описанной системы в виде веб-приложения.

Количество слов

152