

**Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Московский государственный технический
университет имени Н.Э. Баумана»**

Факультет «Робототехника и комплексная автоматизация»
Кафедра «Системы автоматизированного проектирования»

***Разработка подсистемы автоматической вёрстки
отчетной документации о ходе научно-образовательной
деятельности по различным направлениям***

Студент РК6-82Б: Идрисов М.Т.

Научный руководитель: доцент кафедры РК6,
к.ф.-м.н., Соколов А.П.

Актуальность работы

При огромном количестве документов не представляется возможным использовать ручные способы автоматизации. Для решения существующей проблемы необходимо программное обеспечение для автоматизации процесса обработки и сбора постоянно формируемой отчетной документации с последующим их объединением в единые документы.

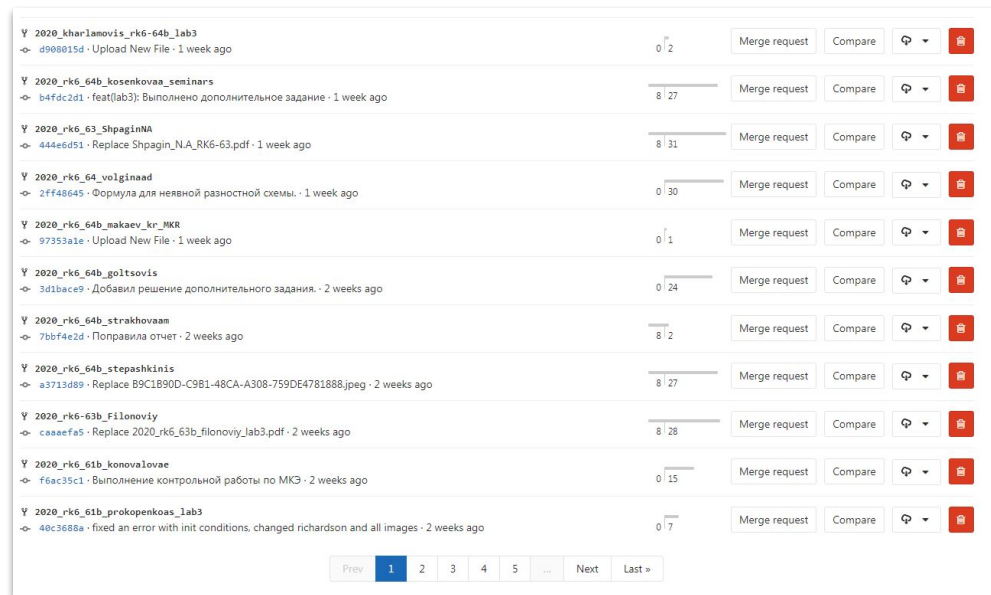


Рисунок 1 Множество веток репозитория comwps с располагаемым в каждом множество документов

Обзор существующих решений

Среди существующих разработок по созданию отчетов следует выделить наиболее популярные и многофункциональные: **Tableau**, **Microsoft Power BI**, **SAS Visual Analytics**

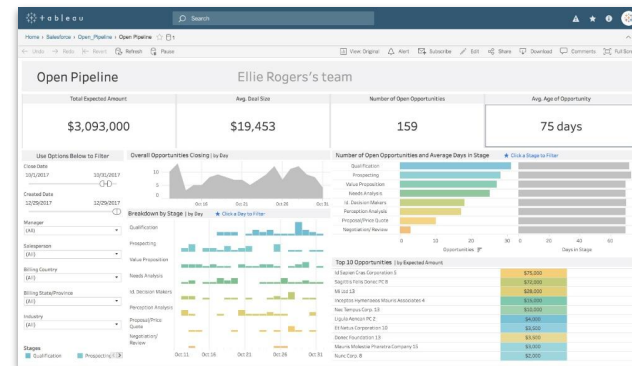


Рисунок 2. Интерфейс Tableau

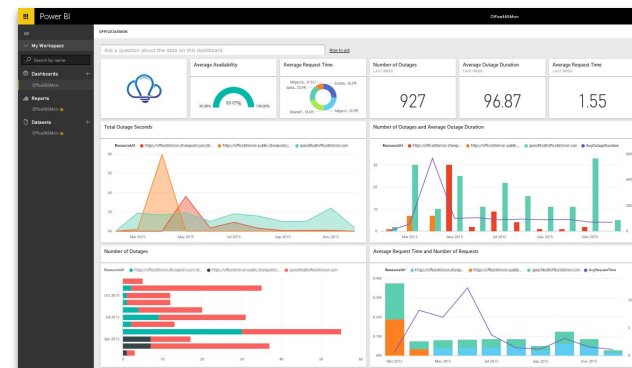


Рисунок 3. Интерфейс Power BI

Выбор архитектуры

Два популярных типа архитектур: **монолитная архитектура** и **микросервисная архитектура**.

Минусы **монолитной архитектуры**:

- высокая сложность
- длинный путь от сохранения изменений до их развертывания
- длительное тестирование
- трудности с масштабированием
- сложно добиться надежности приложения
- зависимость от постепенно устаревающего стека технологий

Плюсы **микросервисной архитектуры**:

- сервисы получаются небольшими и простыми в обслуживании
- сервисы развертываются независимо друг от друга
- сервисы масштабируются независимо друг от друга
- позволяет экспериментировать и внедрять новые технологии

Создание базового docker-образа

Docker - программное обеспечение для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации.

Образ (image) - «сущность» или «общий вид» (union view) стека слоев только для чтения.

Контейнер (container) - «сущность» или «общий вид» (union view) стека слоев только для чтения с верхним слоем для записи

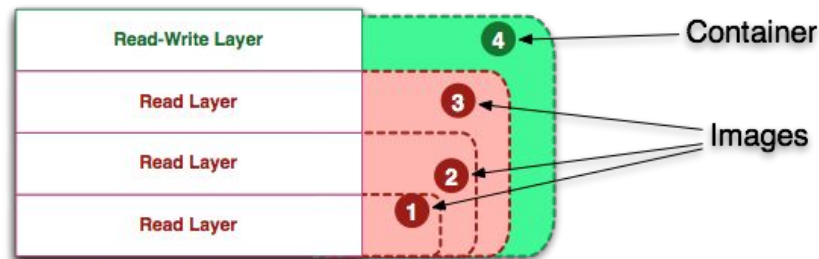


Рисунок 4. Схематическое представление слоев образа и контейнера

Создание базового docker-образа

Базовый образ **python:3.8-slim-buster** был дополнен пакетами и утилитами для компиляции TeX файлов в pdf.

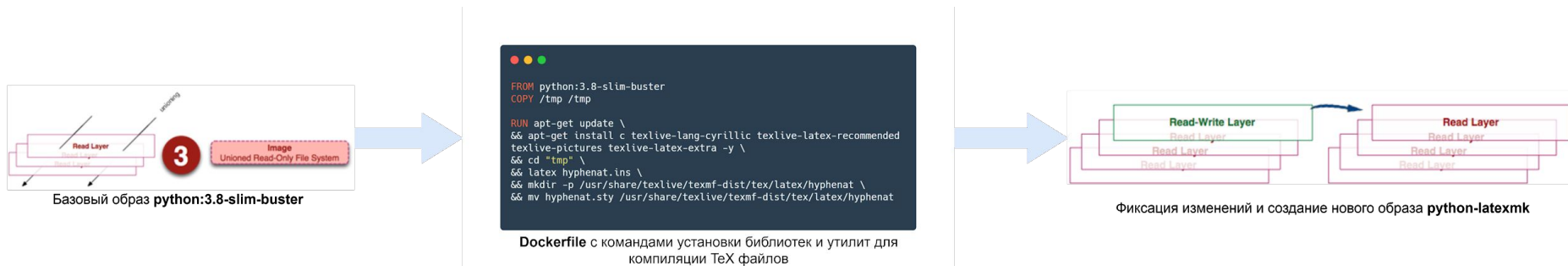


Рисунок 5. Дополнение базового образа требуемыми пакетами и утилитами

Используемые модули и сервисы

Django — свободный фреймворк для веб-приложений на языке Python.

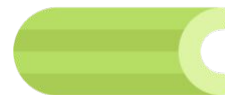
python-gitlab — обеспечивает доступ к API сервера GitLab, используя python-вызовы.

jinja — современный и удобный язык шаблонов для Python, созданный по образцу шаблонов Django.

Celery — асинхронная очередь задач, основанная на передаче сообщений.

Flower — встроенный в Celery веб-инструмент для мониторинга и администрирования задач.

RabbitMQ — брокер сообщений на основе стандарта AMQP.



Основной цикл работы

Непрерывно работающий скрипт проверяет изменения в ветках и в случае изменений запускает в работу задачу на создание/обновление отчета.

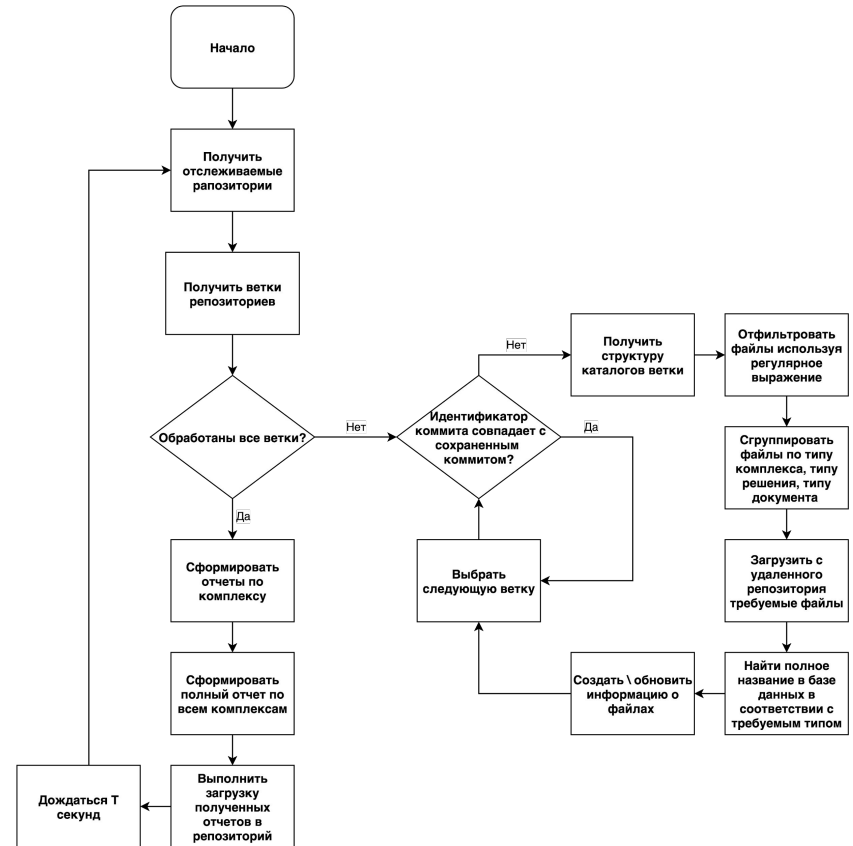


Рисунок 6. Блок-схема основного цикла программы

Схема взаимодействия микросервисов

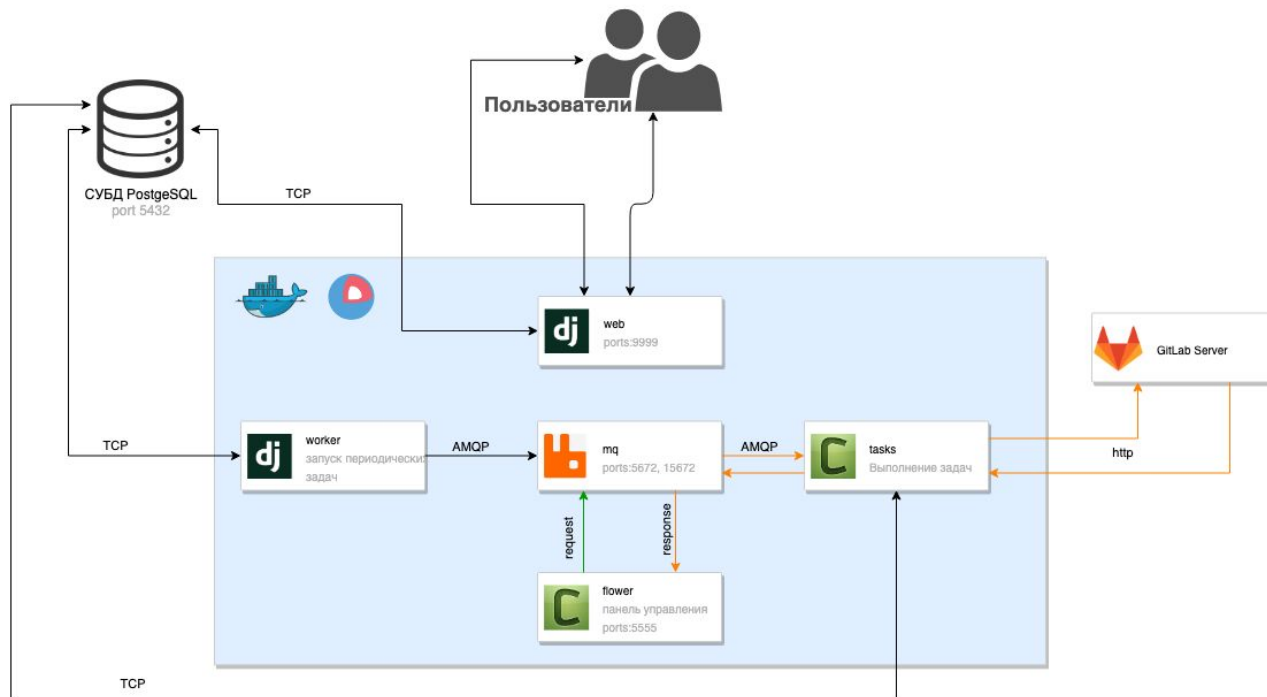


Рисунок 7. Схема взаимодействия контейнеров

Тестирование

Два метода тестирования:

- **модульное тестирование** — метод, позволяющий проверить на корректность отдельные модули исходного кода программы
- **ручное тестирование** — метод для проверки путем моделирования действий пользователя

Модульное тестирование

Pytest — фреймворк для тестирования, который позволяет писать тесты с использованием языка Python.

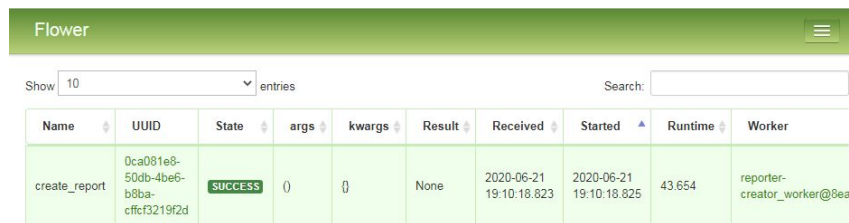
Fixtures — функции, выполняемые pytest до фактических тестовых функций.

pytest-django — плагин, который позволяет тестировать приложение Django с помощью pytest

```
def test_group_by_type(self):
    notes = [
        {
            'id':
            'f0b4b21c607290d52c35af3ca6b36270a4d92fc9',
            'name': 'rndhpc_rem_2020_04_09_n01.tex',
            'type': 'rem',
            'path': 'rddoc/rndhpc_rem_2020_04_09_n01.tex',
            'mode': '100644',
            'complex': 'rndhpc',
            'year': '20',
            'month': '04',
            'day': '09',
            'title': 'n01'
        },
        {
            'id':
            '78e41117f4663ac886c01c4790ccf5524e194fc9',
            'name': 'rndhpc_rem_2025_12_10_214124.tex',
            'type': 'rem',
            'path':
            'rddoc/rndhpc_rem_2025_12_10_214124.tex',
            'mode': '100644',
            'complex': 'rndhpc',
            'year': '25',
            'month': '12',
            'day': '10',
            'title': '214124'
        }
    ]
    grouped_notes = self.worker.group_by_type(notes)
    assert grouped_notes == {
        'Разработка ресурсоемкого ПО инж.анализа': {
            'Заметка общего назначения': [
                {
                    'complex': 'rndhpc',
                    'day': '10',
                    'id':
                    '78e41117f4663ac886c01c4790ccf5524e194fc9',
                    'mode': '100644',
                    'month': '12',
                    'name': 'rndhpc_rem_2025_12_10_214124.tex',
                    'path':
                    'rddoc/rndhpc_rem_2025_12_10_214124.tex',
                    'title': '214124',
                    'type': 'rem',
                    'year': '25'
                },
                {
                    'complex': 'rndhpc',
                    'day': '09',
                    'id':
                    'f0b4b21c607290d52c35af3ca6b36270a4d92fc9',
                    'mode': '100644',
                    'month': '04',
                    'name': 'rndhpc_rem_2020_04_09_n01.tex',
                    'path':
                    'rddoc/rndhpc_rem_2020_04_09_n01.tex',
                    'title': 'n01',
                    'type': 'rem',
                    'year': '20'
                }
            ]
        }
    }
```

Рисунок 8. Пример модульного теста

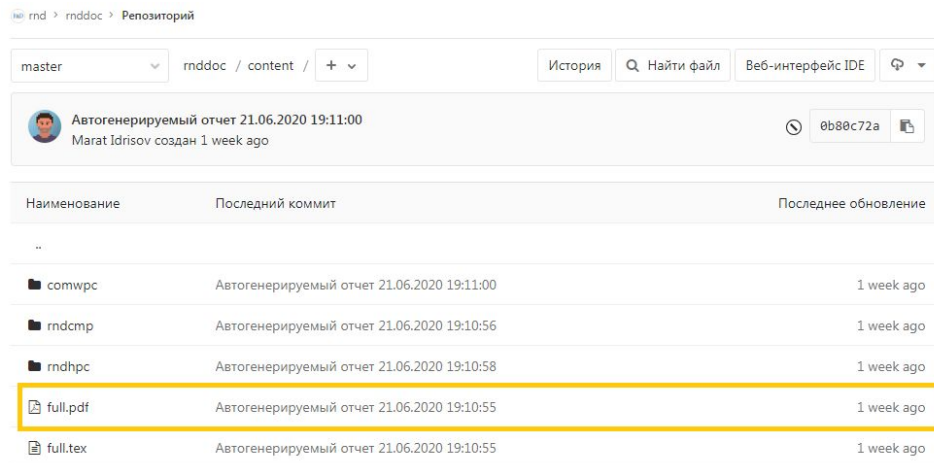
Ручное тестирование



The screenshot shows the Flower web interface. At the top, there's a green header with the word "Flower" and a menu icon. Below it, there's a "Show" dropdown set to "10" and a "Search:" input field. The main part of the interface is a table with columns: Name, UUID, State, args, kwargs, Result, Received, Started, Runtime, and Worker. One row is visible, showing a task named "create_report" with a green "SUCCESS" state.

| Name | UUID | State | args | kwargs | Result | Received | Started | Runtime | Worker |
|---------------|-------------------------------------|---------|------|--------|--------|-------------------------|-------------------------|---------|-----------------------------|
| create_report | 0ca081e8-50db-4be6-b9ba-cffc3219f2d | SUCCESS | () | {} | None | 2020-06-21 19:10:18.823 | 2020-06-21 19:10:18.825 | 43.654 | reporter-creator_worker@8ea |

Рисунок 9. Успешно выполненная задача в панели мониторинга Flower



The screenshot shows the GitLab web interface for a repository named "rmdoc". The breadcrumb path is "rmd > rmdoc > Репозиторий". The current branch is "master". There are buttons for "История", "Найти файл", "Веб-интерфейс IDE", and a fork icon. A commit by "Marat Idrisov" is shown, dated "21.06.2020 19:11:00". Below this is a table of files in the repository.

| Наименование | Последний коммит | Последнее обновление |
|--------------|--|----------------------|
| .. | | |
| comwpc | Автогенерируемый отчет 21.06.2020 19:11:00 | 1 week ago |
| rndcmp | Автогенерируемый отчет 21.06.2020 19:10:56 | 1 week ago |
| rndhpc | Автогенерируемый отчет 21.06.2020 19:10:58 | 1 week ago |
| full.pdf | Автогенерируемый отчет 21.06.2020 19:10:55 | 1 week ago |
| full.tex | Автогенерируемый отчет 21.06.2020 19:10:55 | 1 week ago |

Рисунок 10. Сгенерированные отчеты в репозитории GitLab

Заключение

В результате работы удалось:

- сравнить различные виды архитектур
- спроектировать приложение на основе микросервисной архитектуры
- предложить и программно реализовать подходы создания отчетов
- описать требования для генерации отчетов
- заложить основы для дальнейших разработок генерации более сложных персонализированных отчетов