**Федеральное государственное образовательное бюджетное учреждение высшего образования**

**«Финансовый университет при Правительстве Российской Федерации»**

**Колледж информатики и программирования**

ПМ.03 Участие в интеграции программных модулей

Группа: 4ПКС-115

**Отчет по производственной практике**

**(по профилю специальности)**

**Руководитель практики от предприятия**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Руководитель практики от колледжа**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Студент**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Оценка** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.2019

2019

СОДЕРЖАНИЕ

[Введение 3](#_Toc630384)

[1 Общая часть 5](#_Toc630385)

[1.1 Описание структурного подразделения организации 5](#_Toc630386)

[1.2 Технические и программные средства 5](#_Toc630387)

[1.3 Процесс выполнения работ 6](#_Toc630388)

[Заключение 17](#_Toc630389)

[Список литературы 18](#_Toc630390)

# Введение

АО КБ «РУСНАРБАНК» основан в 2001 году и успешно работает на рынке с 2002 года на основании Лицензии Центрального Банка РФ № 3403

АО КБ «РУСНАРБАНК» является универсальным коммерческим банком, оказывающим услуги юридическим и физическим лицам. Головной офис Банка располагается в г. Москве. Сеть отделений Банка включает филиал в г. Белгороде, 3 дополнительных офиса в г. Москве и 2 дополнительных офиса в г. Белгороде.

В апреле 2009 года состоялось объединение АО КБ «РУСНАРБАНК» и ЗАО «БелДорБанк» в форме присоединения последнего к АО КБ «РУСНАРБАНК».

В ноябре 2016 года произошла смена акционеров Банка - Банк перешел под контроль АО «РЕГИОН Эссет Менеджмент» Д.У. ЗПИФ долгосрочных прямых инвестиций «Ритм» (Группа Компаний «РЕГИОН») с долей владения 100%.

Группа Компаний «РЕГИОН» – одна из крупнейших в России частных инвестиционных групп, охватывающая основные направления финансовой деятельности и ориентирующаяся на комплексное обслуживание корпоративных клиентов и институциональных инвесторов.

В мае 2017 года Совет Директоров Банка принял решение усилить команду топ-менеджеров, в результате чего в Правление Банка вошли 5 новых членов правления, в том числе Председатель Правления Синицын Александр Владимирович.

Рейтинговым агентством Эксперт РА Банку присвоен долгосрочный кредитный рейтинг по национальной шкале на уровне ruBBB- прогноз «Стабильный», Агентством Agentur Expert RA GmbH (Германия) присвоен кредитный рейтинг на уровне B+ по международной шкале.

В июле 2017 года Советом Директоров Банка утверждена новая Стратегия Банка до 2020 года, направленная на построение универсального банка с фокусом на развитие кредитования малого и среднего бизнеса и ритейл, а также усиление работы на рынке ценных бумаг. В основу стратегического планирования заложено построение среднего динамично развивающего Банка с темпом прироста Активов выше среднего уровня, перед Банком поставлены задачи по формированию и развитию широкой продуктовой линейки, получение Банком аккредитации в программах рефинансирования Корпорации МСП, АИЖК и др. структурах, внедрение новой масштабной банковской IT системы, позволяющей предоставлять современные банковские услуги, реализация индивидуальной тарифной политики для различных клиентских сегментов, развитие региональной сети продаж с учетом повышения общей эффективности бизнеса.

# 1 Общая часть

## 1.1 Описание структурного подразделения организации

Практика проходилась в подразделении автоматизации бизнес-процессов АО КБ «РУСНАРБАНК». Основные задачи подразделения:

* Поддержка онлайн-сервисов банка;
* Эффективная поддержка оперативной деятельности банка, организация учета и контроля;
* Подготовка любых документов для партнеров, включая накладные, счет-фактуры, акты сверки и деловые предложения;
* Быстрое получение отчетов о состоянии дел в банке за любой период времени;
* Оптимизация затрат на персонал, увеличение эффективности использования рабочего времени путем освобождения сотрудников от рутинной работы;
* Сведение к минимуму негативного влияния «человеческого фактора» на важнейшие бизнес-процессы;
* Безопасное хранение информации о клиентах банка;
* Повышение качества обслуживания клиентов.

## 1.2 Технические и программные средства

От практиканта было необходимо:

* Проектирование алгоритмов, основанных на базовых математических алгоритмах компьютерного зрения;
* Проектирование и разработка комплекса модулей для системы распознавания финансовой отчетности по формам бухгалтерского баланса ОКУД 0710001 и 0710002;
* Интеграция решения с технологией Docker для оперативной развертки системы;

Практиканту было предоставлено:

* Рабочий компьютер;
* Возможность использовать любые open source решения для решения поставленных задач;

## 1.3 Процесс выполнения работ

В ходе практики была создана система для распознавания финансовой отчетности с применением принципа ООП со следующими модулями:

* DictionaryGetter - модуль для хранения данных словарей associated\_dictionary и month\_dictionary, является аналогом статического класса;
* MatrixToJson - модуль для преборазования матрицы finalmatrix из OKUD в выходной JSON;
* OKUD - модуль для формирования матрицы для документов 1-5 типа (OKUD);
* Opencv - основной модуль программы для распознавания PDF документов;
* TableRecognition – модуль, содержащий всю логику распознавания таблицы;
* UtilModule – модуль с простыми статическими методами для общей работы программы;

Разберем каждый модуль подробнее.

Методы модуля DictionaryGetter:

* GetCodesAndNames() - процедура для чтения словаря associated\_dictionary из файла CodesAndNames.yaml, CodesAndNames.yaml содержит сопоставление кода бухгалтерского баланса с наименованием в отчетности;
* GetMonthAndNumbers() - процедура для чтения словаря month\_dictionary из файла MonthAndNumbers.yaml. MonthAndNumbers.yaml содержит сопоставление месяца с его кодом, т.е: "Январь" -> "01".

Классы модуля MatrixToJson:

* SuperVisor - содержит логику получения qc (quality control), значения qc: 0 -> все отлично, 1 -> есть некоторые вопросы, но нет смысла беспокоиться, 2 -> есть серьёзные проблемы, отдаваемый результат неправильный, 3 -> программа упала с ошибкой.
* ToJSON - используется для преобразования данных матрицы в выходной JSON;

Методы модуля MatrixToJson:

* GetCodeQCStatus - метод для определения qc кода бухгалтерского баланса: ищет полное сопоставление в словаре associated\_dictionary, получает код бух. баланса – code, получает наименование, сопоставленное с кодом бухгалтерского баланса – text, получает значение кода – content;
* GetMainDocStatus - метод для определения qc всего документа: получает list с индексами кодов бухгалтерского учета – coords, получает qc, который мы передали с OKUD (на стороне OKUD базовая проверка на адекватность данных), отдаёт новый qc;
* GetSmallTableQCStatus - метод для определения qc верхней правой таблицы (smalltable) на документах 1 и 3 типа: получает list table, отдает число от 0 до 2 – qc;
* FiveDocProcessing - метод с логикой формирования JSON для документов 5 типа (документов, которые нельзя идентифицировать по маркеру): формирует из квадратной матрицы n на m JSON c текстом, берет header'ы из предыдущего объекта JSON;
* Processing - метод с логикой формирования JSON для документов отличных от 5 типа, формирует из квадратной матрицы n на m JSON c текстом;
* SmallTableProcessing - метод для формирования smalltable для некоторых типов документов (малая верхняя правая таблица): обращается к self.small\_table, вставляет форматированный JSON в self.small\_table;
* Конструктор класса ToJSON. В зависимости от типа документа: вызывает метод SmallTableProcessing(), если есть smalltable (в 1 и 3 типе), вызывает метод FiveDocProcessing(), если документ 5 типа, вызывает метод Processing(), если документ не 5 типа, получает list small\_table, содержащий данные верхней правой таблицы, получает oldjson, содержащий предыдущий json в итерации для подстановки header'ов из него, елси документ 5 типа, получает list matrix с данными основной таблицы, получает тип документа doctype, получает qc документа, который выставляется в проверке в OKUD – docqc;

Классы модуля OKUD:

* OKUD\_0710001 - класс для работы с 1 страницей формы 0710001;
* OKUD\_0710001\_CONTINUEPAGE - класс для работы с последующими страницами формы 0710001;
* OKUD\_0710002 - класс для работы с 1 страницей формы 0710002;
* OKUD\_0710002\_CONTINUEPAGE - класс для работы с последующими страницами формы 0710002;
* OKUD\_ALTERCLASS - класс для процессинга 5 типа документов;
* OKUD\_PROCESSING - класс с логикой формировнаия матрицы для последующей отдачи в MatrixToJson.TOJSON;

Методы модуля OKUD:

* Конструктор класса OKUD\_0710001: получает предыдущий JSON – oldjson, получает изображение в numpy array – image, вызывает \_\_OKUD\_0710001\_SMALLTABLE, вызывает \_\_OKUD\_0710001\_MAINPAGE;
* Конструктор класса OKUD\_0710001\_CONTINUEPAGE: получает предыдущий JSON – oldjson, получает изображение в numpy array – image, вызывает \_\_OKUD\_0710001\_CONTINUEPAGE;
* Конструктор класса OKUD\_0710002: получает предыдущий JSON – oldjson, получает изображение в numpy array – image, вызывает \_\_OKUD\_0710002\_SMALLTABLE, вызывает \_\_OKUD\_0710002\_MAINPAGE;
* Конструктор класса OKUD\_0710002\_CONTINUEPAGE: получает предыдущий JSON – oldjson, получает изображение в numpy array – image, вызывает \_\_OKUD\_0710002\_CONTINUEPAGE;
* Конструктор класса OKUD\_ALTERCLASS: получает предыдущий JSON – oldjson, получает изображение в numpy array – image, вызывает \_\_OKUD\_ALTERCLASS;
* Конструктор класса OKUD\_PROCESSING, принимает следующие параметры: изображение в numpy array – thisimage, координаты прямоугольника с данными на изображении – maxobj, данные малой таблицы smalltable – tableobj, данные предыдущего JSON (для header'ов в 5 типе документов) – oldjson, тип обрабатываемого доумента – doctype, используемый метод обрезки (по MinAreaRect или BoundingRect) – cropmethod. Вызывает \_\_SmallTableProcessing, если есть данные в smalltable и \_\_MainMatrixProcessing, пока self.MainMatrixProcessingFlag == True c разными коэффициентами NoizeRemoverKoff;

Классы модуля opencv (main):

* GetDocumentType - класс для определения типа исходной страницы документа, определение происходит путем получения координат прямоугольника c текстом с сортировкой по возрастанию оси ординат(y);
* ParserClass - класс для парсинга данных определенного типа документа, каждый метод - каждый тип документа;
* ScanerFixClass - класс для препроцессинга изображения путем его выравнивания и обрезки;

Методы модуля opencv (main):

* Конструктор класса GetDocumentType: принимает в качестве параметра изображение в виде numpy array, объявляет все ассоциации документов в self.dictionary\_getter, получает self.textboxes с помощью статического UtilModule.UtilClass.GetText, вызывает свой единственный метод get\_type\_by\_text;
* get\_type\_by\_text - метод для получения типа страницы документа: ищет вхождение строки ассоциативного массива self.dictionary\_getter c распознанным текстом text с точек прямоугольника, хранящихся в self.textboxe, обращается к self.dictionary\_getter для ассоциации, к self.image для обрезки изображения с исходного, к self.textboxes для получение координат прямоугольников с текстом. Отдает номер распознанного типа документа в self.doctype;
* Конструктор класса ParserClass - вызывает методы парсинга на основе типа распознанного документа в GetDocumentType.get\_type\_by\_text. За ассоциацию типа документа и методы отвечает self.method\_number\_list. Получает изображение в виде numpy array – image, номер типа документа - method\_number. Объявляет поля для отдачи текста self.OUTPUT\_OBJ и self.TABLE\_OUTPUT\_OBJ;
* CropAroundCenter - процедура для отсечения от изображения лишних контуров. От центра изображения вычисляет параметры, полученные от RotatedRectWithMaxArea. Обращается к self.RectWithMaxArea для получения ширины и высоты прямоугольника, к изображению self.image. Отдает новое обрезанное изображение в self.image;
* DetectBox - процедура вторичной для обрезки изображения. Используется для обрезки белых областей со всех сторон изображения, необходима для правильной работы класса определения типа документа. Обращается к изображению self.image, отдает новое изображение;
* HoughCheck - процедура для выпрямления угла изображения на 90 градусов. Использует преобразование Хафа (Hough Transform) для вычисления угла наклона исходного изображения и imutils для выпрямления изображения на угол отклонения. Обращается к self.image, отдает новое изображение в self.image и угол отклонения в градусах self.degree;

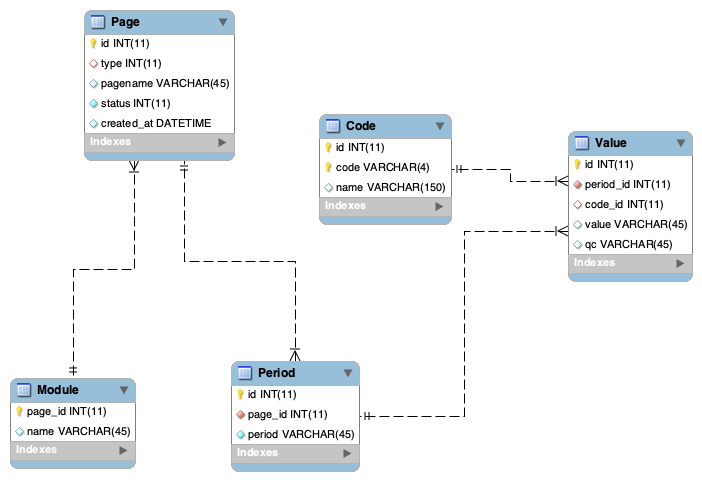
БД для хранения данных распознанных документов отчетности банка c архитектурой на рисунке 1 ниже:

Рисунок 1 – Схема разработанной базы данных

БД содержит в себе следующие таблицы:

* Page для хранения основных данных о каждой странице документа;
* Module для хранения данных о модулях программы, обрабатывающих какой-либо тип документа;
* Period для хранения данных о периодах на странице документа
* Value для хранения значения, связанного с периодом;
* Code для ассоциации текста и кодов строк в формах бухгалтерской отчетности, указываемых в годовой отчетности организации.

Для нагрузочного тестирования БД было необходимо написать программу для заполнения тестовыми данными, а также провести ручное тестирование текущей архитектуры. Стуруктура программы, написанная на языке программирования Python изображена на рисунке 2 ниже.

Рисунок 2 – Структура по методам программы для заполнения БД тестовыми данными

Программа для тестирования БД содержит в себе два модуля:

* DB – исполняемая программа;
* DataGenerator – модуль генерации данных для текущей архитектуры БД.

Программа содержит следующие классы:

* DB.ListGeneratorClass используется для получения сгенерированных данных методом SQLGenerator;
* DatabaseLogicClass содержит все методы для работы с БД;
* PageClass – генерация данных для таблицы Page, содержит единственный метод Getter;
* PeriodClass – генерация данных для таблицы Period, содержит единственный метод Getter;
* ValueClass – генерация данных для таблицы Value, содержит единственный метод Getter;

В качестве отличительной особенности разработанной программы можно отметить использование механизма транзакций.

Транзакция — это совокупность одной или нескольких SQL-инструкций, имеющих начало и конец. В конце транзакции происходит либо ее отмена, либо подтверждение. Отмена транзакции называется откатом (rollback) (пример с deadlock’ом показан на рисунке 3), так как идет последовательная отмена всех сделанных изменений. Подтверждение транзакции называется фиксацией (commit).

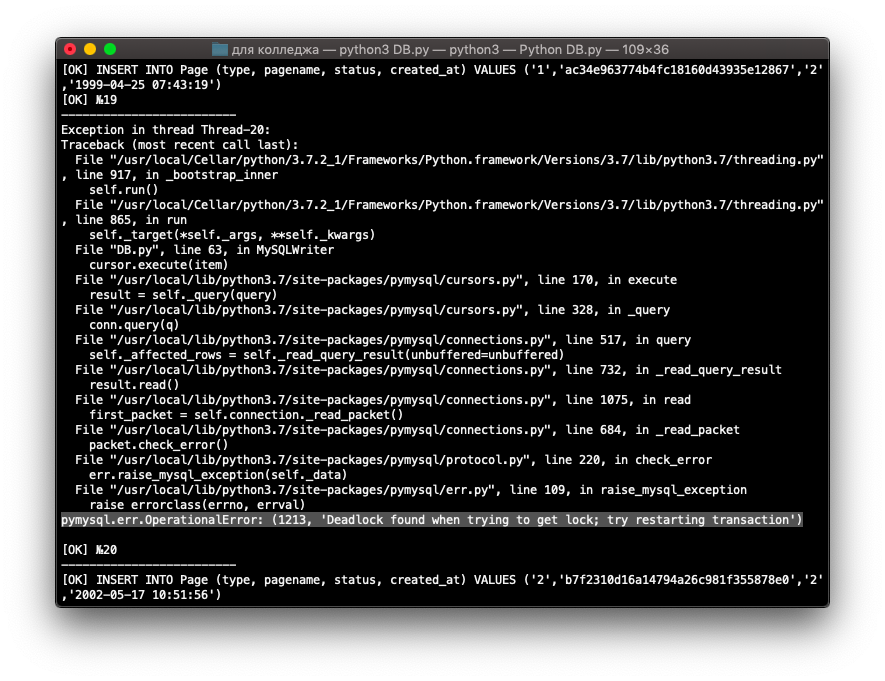


Рисунок 3 – Пример Deadlock’a и отката транзакции

Считается, что правильная транзакция обладает следующими свойствами:

* Атомарностью;
* Согласованностью;
* Изолированностью;
* Устойчивостью.

Процесс работы приложения для заполнения БД тестовыми данными показан на рисунке 4 ниже, нагрузочное тестирование БД на рисунке 5.

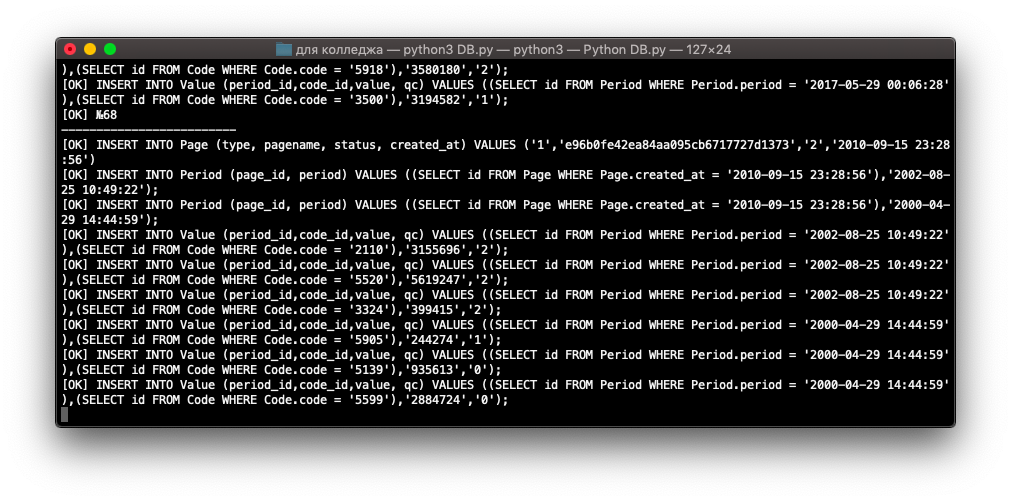


Рисунок 4 – Работа приложения для заполнения разработанной БД тестовыми данными

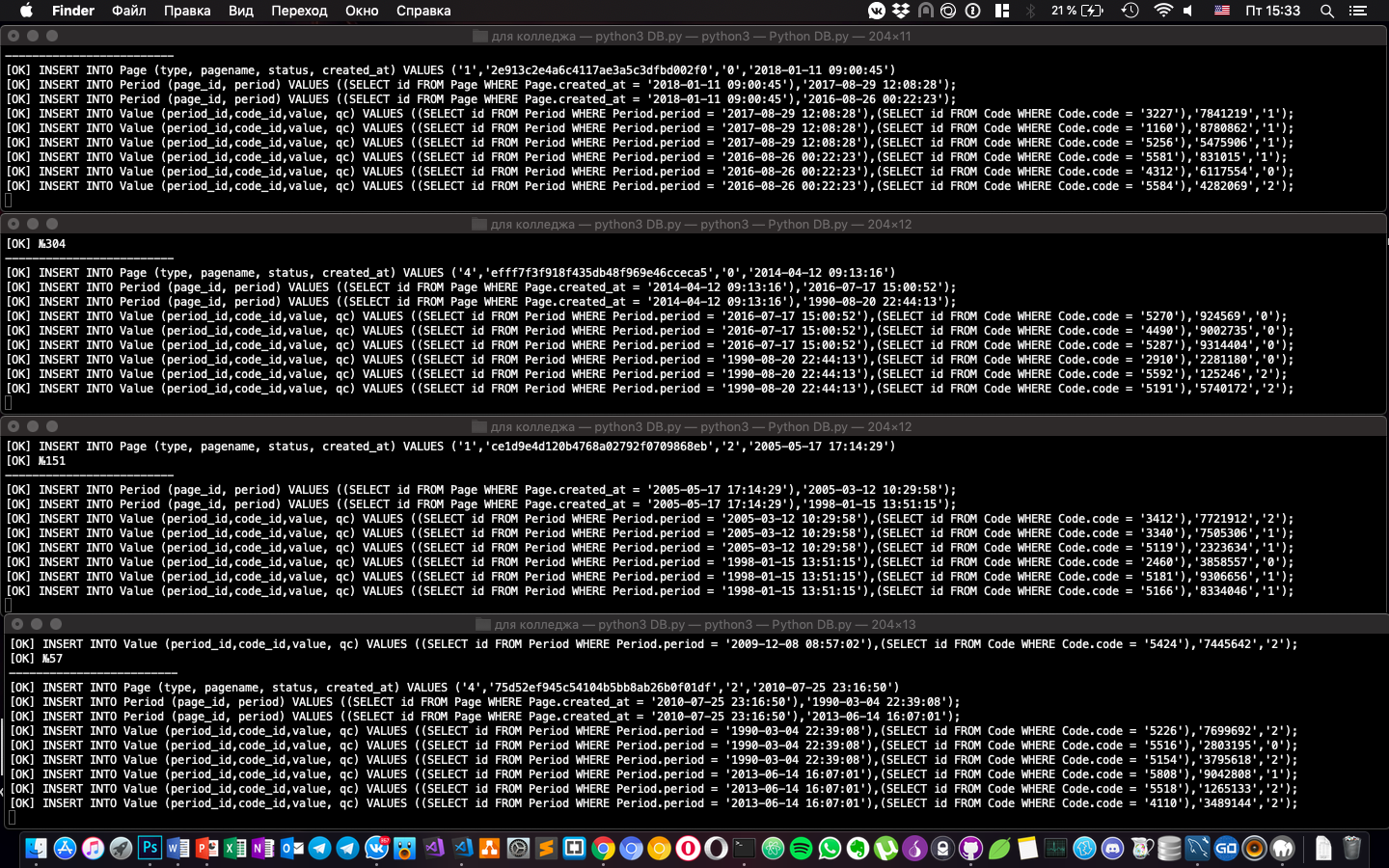


Рисунок 5 – Нагрузочное тестирование БД

Спроектированная и заполненная БД с тестовыми данными содержит в себе следующее количество записей по таблицам.

* Таблица Value – 50322 записей;
* Таблица Period – 16774 записей;
* Таблица Page – 8387 записей;
* Таблица Module – 4 записи;
* Таблица Code – 571 запись;

Результаты выполнения запросов находятся на рисунке 6 ниже:

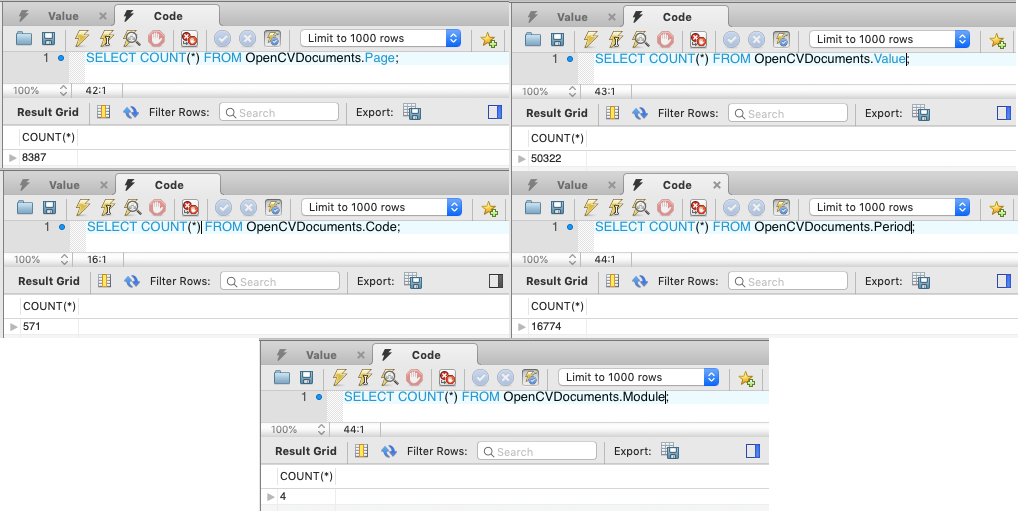


Рисунок 6 – Результаты выполнения запросов COUNT в БД

Пример данных в таблицах можно наблюдать на рисунках 7-9 ниже:

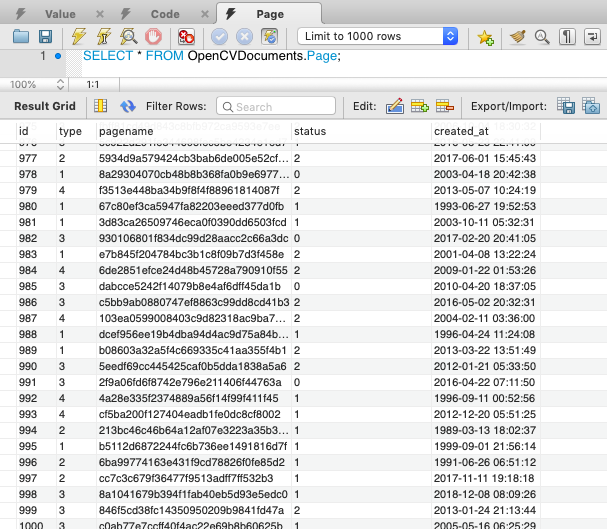
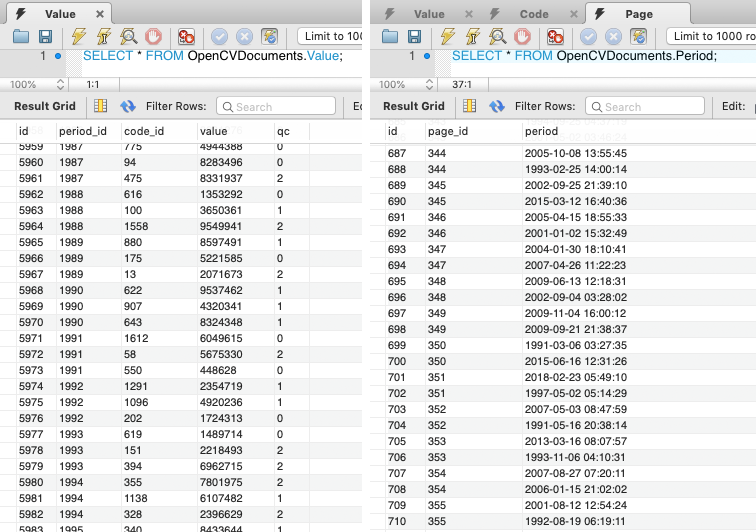


Рисунок 7 – Пример данных в таблице Period

Рисунок 8 – Пример данных в таблице Value/Period

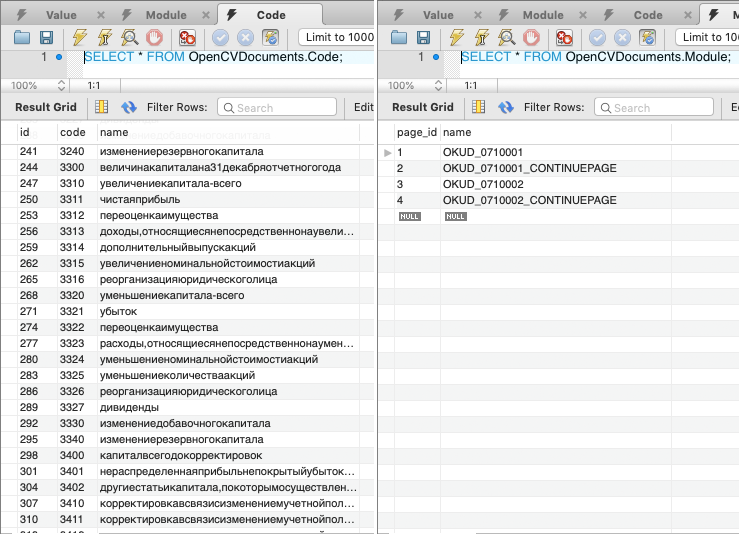


Рисунок 9 – Пример данных в таблицах Code/Module

Теперь идет работа и развертка docker для контейнеров Python+Flask API/MySQL/PHPMyAdmin/Redis. Сначала необходимо было поставить сам пакет:

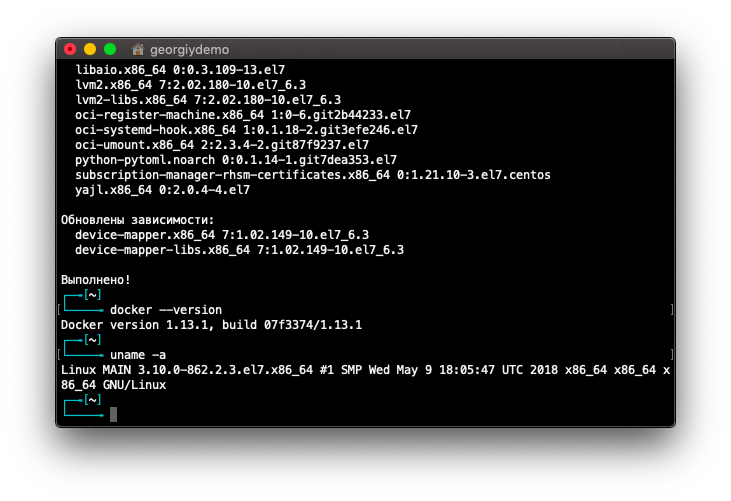


Рисунок 10 – Установленный пакет Docker на дистрибутив CentOS

Далее была проделана работа по написанию docker-compose со связкой двух контейнеров:

* Python:3.7-alpine;
* Redis:alpine.

Также в compose был выполнен проброс порта 5000 для доступа к Flask API вне контейнера. Порт 6379 используется для Redis между контейнерами (рисунки 11 – 12).

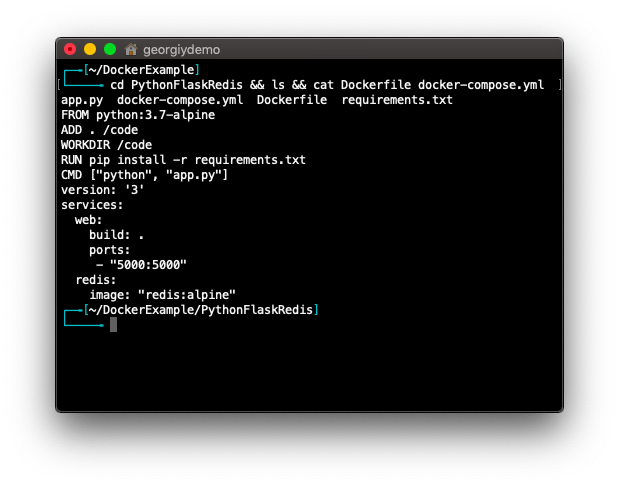


Рисунок 11 – Созданный docker-compose с Python+Flask+Redis



Рисунок 12 – Запуск созданного docker-compose

Также был создан docker-compose cо связкой MySQL + PHPMyAdmin. Используются контейнеры:

* Php:7.2.6-apache + mysql;
* Phpmyadmin/phpmyadmin.

Выполнены следующие пробросы портов:

* 8001:80 – используется для PHPMyAdmin;
* 8000:80 – также используется для PHPMyAdmin (альтернатива);
* 3306:3306 – используется для подключения к БД MySQL.

Файл compose и запуск контейнера показаны на рисунках 13 - 14 ниже:



Рисунок 13 – Созданный docker-compose с PHP+Apache+MySQL+PHPMyAdmin

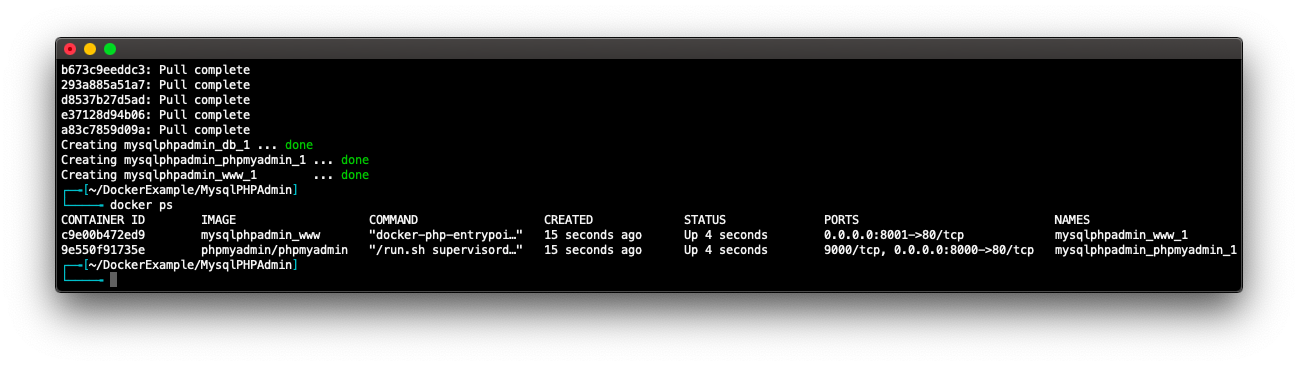


Рисунок 14 – Запуск созданного docker-compose

В итоге получили полностью развернутую систему для хранения и взаимодействия с данными на рисунке 15.



Рисунок 15 – Список всех развернутых контейнеров на сервере

# Заключение

В ходе практики были выполнены все поставленные задачи:

* Проектирование и создание объектов базы данных MySQL для предприятия;
* Разработка приложений БД и заполнение разработанной БД тестовыми данными
* Управление и разграничение доступа к БД;
* Использование стандартных методов защиты БД на основе связи таблиц по foreign key
* Развертка docker-контейнера с MySQL/Apache/PHP/PHPMyAdmin на сервере предприятия;
* Развертка docker-контейнера с Redis/Python+Flask на сервере предприятия;

В результате прохождения практики по модулю ПМ.02 была проделана существенная работа по закреплению моих теоретических и практических навыков.

# Список литературы

1. Docker-compose.yml for mysql and phpmyadmin – [Электронный ресурс]. – Режим доступа: https://gotechnies.com/docker-compose-yml-mysql-phpmyadmin/
2. Get started with Docker Compose – [Электронный ресурс]. – Режим доступа: https://docs.docker.com/compose/gettingstarted/
3. How to install Docker and deploy a LAMP Stack – [Электронный ресурс]. – Режим доступа: https://www.linode.com/docs/applications/containers/how-to-install-docker-and-deploy-a-lamp-stack/
4. Использование NoSQL Redis в качестве основного хранилища – [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/178525/
5. Docker + Flask. Simple Tutorial– [Электронный ресурс]. – Режим доступа: https://medium.com/@mtngt/docker-flask-a-simple-tutorial-bbcb2f4110b5
6. Транзакции, блокировки и многопользовательский доступ к данным. – [Электронный ресурс]. – Режим доступа: http://www.mstu.edu.ru/study/materials/zelenkov/ch\_4\_9.html
7. Основы правильного проектирования баз данных в веб-разработке – [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/115777/
8. Using Transactions in Python Programs – [Электронный ресурс]. – Режим доступа: https://www.oreilly.com/library/view/mysql-cookbook-2nd/059652708X/ch15s08.html
9. Use Commit and Rollback to Manage MySQL Transactions in Python – [Электронный ресурс]. – Режим доступа: https://pynative.com/python-mysql-transaction-management-using-commit-rollback/
10. Транзакции в mysql и SELECT FOR UPDATE – [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/238119/
11. Блокировки и уровни изоляции транзакций InnoDB в MySQL – [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/238513/
12. GraphQL. Введение. – [Электронный ресурс]. – Режим доступа: https://habr.com/post/326986/
13. Проектирование Web API в 7 шагов – [Электронный ресурс]. – Режим доступа: https://habr.com/ru/company/geekfamily/blog/256495/