**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](#_Toc3745384)

[1 Общая часть 5](#_Toc3745385)

[1.1 Описание структурного подразделения организации 5](#_Toc3745386)

[1.2 Технические и программные средства 5](#_Toc3745387)

[1.3 Процесс выполнения работ 6](#_Toc3745388)

[Заключение 17](#_Toc3745389)

[Список литературы 18](#_Toc3745390)

# 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 обработки изображений, 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;
* Сервер CentOS7 с возможностью использовать root-доступ по протоколу ssh;
* 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;
* RotatedRectWithMaxArea - процедура для вычисления повернутого прямоугольника с максимальной площадью. Обращается к изображению self.image и градусам поворота изображения self.degree. Отдает ширину и высоту self.RectWithMaxArea;
* Конструктор класса ScanerFixClass: вызывает последовательно все методы класса, менять вызов методов местами нельзя. Принимает в качестве параметра изображение в виде numpy array;

Методы модуля TableRecognition:

* compute\_cell\_hulls - запускает find\_table\_cell\_polygons() и вычисляет прямоугольник, охватывающий ячейку (для каждой ячейки). Для большинства (4-точечных) ячеек это эквивалентно исходному пути, однако при этом удаляется, небольшие неровности и дополнительные точки из более крупных, 5 + - точечных ячеек (в основном объединенные ячейки);
* compute\_cell\_polygons - вычисляет список полигонов ячеек из контуров и связанных с ними угловых кластеров. Генерирует list c ячейкми, который хранится в self.cell\_polygons;
* compute\_contour\_bounding\_boxes - вычисление повернутых bounding boxes с минимальной площадью для каждого контура;
* compute\_contour\_centers - вычисление центра ячеек каждого контура прямоугольника, используя meanCenter();
* compute\_filtered\_missing\_cell\_contours - метод для вычисления отфильтрованных отсутствующих контуров ячеек;
* compute\_missing\_cells\_mask вычисление бинарной маски масштаба изображения (img-scale mask);
* does\_page\_have\_valid\_table - смотрит, есть ли таблица на изображении;
* draw\_cell\_hull() - метод для рисования одной ячейки;
* extract\_cell\_from\_image - извлечение из изображения секции, соответствующий ячейке таблицы. Параметры: img - массив типа 2D, table\_coords - координаты таблицы;
* filter\_contours() фильтрация контуров: которые меньше, чем определенная область (square px), с менее чем n узлами (обычно 4);
* find\_empty\_cells - узнаёт, какие ячейки пусты;
* find\_fine\_table\_corners - находим точные углы таблицы. Эта функция работает с координатами кластера, возвращает те углы, у которых самые экстремальные координаты, смешивая координаты X/Y;
* remove\_non\_table\_nodes(self) - определение самого верхнего узла таблицы ("supernode") (т. е. узел с самыми прямыми дочерними элементами) и удаление каждого узла (и аннулировать любой связанный контур);
* separate\_supernode - удаление supernode из контуров и сохранение его отдельно. Это означает, что только ячейки таблиц и артефакты оставляются в виде контуров;
* visualize\_contours - раскраска информации таблицы на основе контуров. Красный цвет – контур, зеленый цвет – клетки;
* is\_inside\_table - определяет, находится ли данный полигон полностью внутри. Это работает с помощью проверки на нахождение всех точек внутри полигона (или на краю). Ввозвращает True, если прямоугольник корректный;
* transitive\_closure(clusters). Берет карту <номер узла> => <список узлов> и вычисляет все транзитивные замыкания, используя быстрый алгоритм на основе множеств. Возвращает список наборов (каждый набор содержит одно транзитивное замыкание) и таблица поиска идентификатора закрытия узла;

Методы модуля UtilClass:

* CropImager\_BoundingRect - метод для обрезки изображения на точки прямоугольника, возвращаемого cv2.rectangle. Получает numpy массив изображения – img и объект от BoundingRect – item. Возвращает обрезанное изображение в numpy array;
* CropImager\_MinAreaRect - метод для обрезки изображения на точки прямоугольника, возвращаемого cv2.minAreaRect. Получает numpy массив изображения – img, объект от minAreaRect – rect. Возвращает обрезанное изображение в numpy array;
* GetGroupAlterValues - метод для альтернативной группировки значений в list, получает list – d, группирует элементы внутри list между собой, возвращает количество групп;
* GetGroupValues - метод для группировки значений в list, получает list – d, группирует элементы внутри list между собой, возвращает количество групп;
* GetMaxCounter - метод для нахождения максимального элемента в collections.Counter. Получает Counter – InputCounter, возвращает максимальной элемент в InputCounter;
* GetText - метод для определения текста на изображении. Получает numpy массив изображения. Возвращает list, содержащий координаты прямоугольников с текстом;
* NoizeRemover - метод для удаления шума на изображении. Использует cv2.erode и cv2.multiply, получает numpy массив изображения – image, коэффициент rate (обычно от 1.0 до 1.6). Возвращает обработанное изображение без шума в виде numpy array;
* PdfToPng - конвертация PDF -> PNG, использует модуль pdf2image, DPI = 300 (желательно не менять), получает в качестве аргумента имя фала PDF. Возвращает list с именами сохраненных файлов PNG в этой же директории;
* TableRecognation - метод для определения ячеек таблицы. Использует модуль TableRecognition с основным алгоритмом работы. Получает numpy массив изображения. Возвращает list, содержащий координаты прямоугольников с текстом;

В итоге программа после интеграции всех модулей отдает преобразованный из PDF текст в формате JSON (рисунок 1-2).

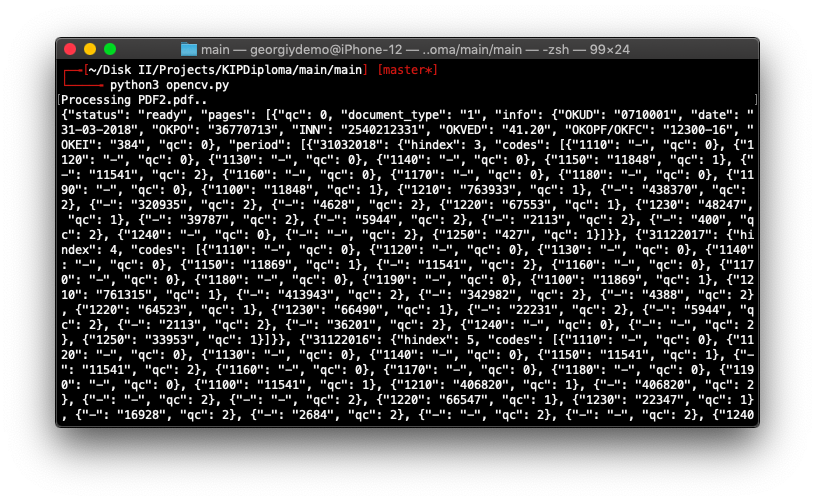


Рисунок 1 – Отдача JSON программой

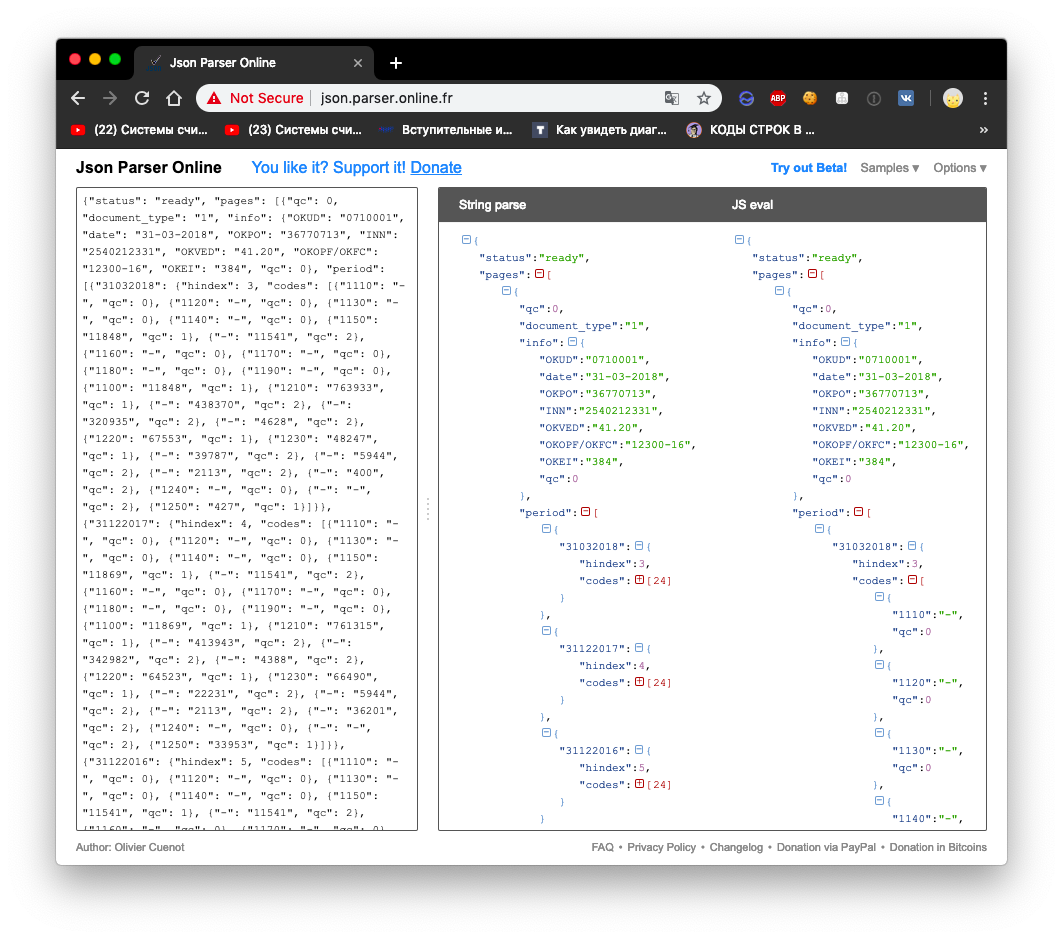


Рисунок 2 – Парсинг JSON веб-сервисом json.parser.online.fr

Также был написан скрипт docker-compose для оперативной развертки разработанного решения (рисунки 3-4)

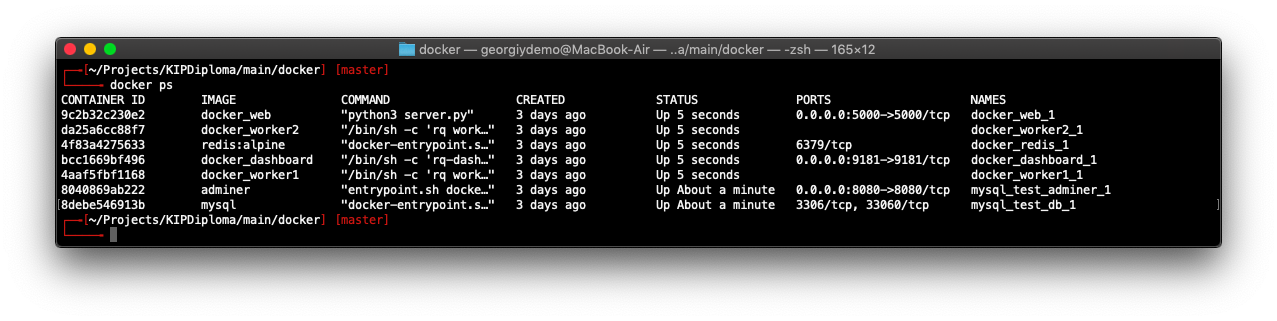


Рисунок 3 – Запущенные docker-контейнеры решения

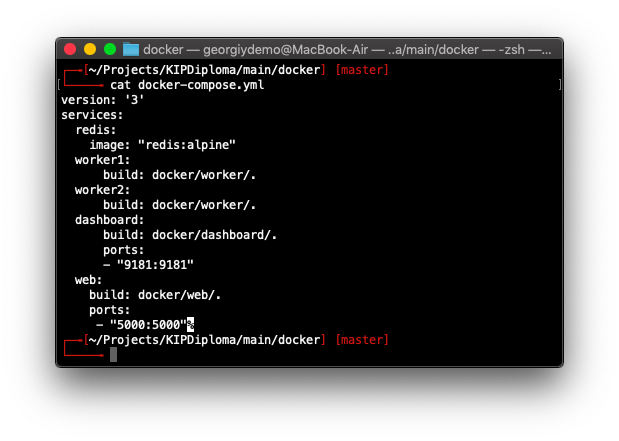


Рисунок 4 – Файл конфигурации docker-compose

# Заключение

В ходе практики были выполнены все поставленные задачи:

* Проектирование алгоритмов обработки изображений, 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;
* Написание сценария docker-compose для управления контейнерами docker;

В результате прохождения практики по модулю ПМ.03 была проделана существенная работа по закреплению моих теоретических и практических навыков.

# Список литературы

1. [ScanDoc] предобработка сканов – [Электронный ресурс]. – Режим доступа: https://habr.com/company/tinkoff/blog/278073/;
2. OpenCV шаг за шагом. Обработка изображения - морфологические преобразования 2 – [Электронный ресурс]. – Режим доступа: http://robocraft.ru/blog/computervision/327.html;
3. Алгоритмы выделения контуров изображений – [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/114452/;
4. Базовые алгоритмы нахождения кратчайших путей во взвешенных графах – [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/119158/
5. Введение в OpenCV применительно к распознаванию линий дорожной разметки – [Электронный ресурс]. – Режим доступа: https://habr.com/company/newprolab/blog/328422/
6. Декораторы в Python – [Электронный ресурс]. – Режим доступа: https://pythonworld.ru/osnovy/dekoratory.html
7. Инструкция по работе с TensorFlow Object Detection API – [Электронный ресурс]. – Режим доступа: https://habr.com/ru/company/nixsolutions/blog/422353/