МИНОБРНАУКИ РОССИИ

Федеральное государственное автономное образовательное   
учреждение высшего образования   
«Южный федеральный университет»

Институт математики, механики   
и компьютерных наук им. И. И. Воровича

Рахилова Екатерина Олеговна

СОЗДАНИЕ СЕРВИСА ДЛЯ АВТОМАТИЗИРОВАННОГО КОНТРОЛЯ СРОКОВ ИСПОЛНЕНИЯ ДОКУМЕНТОВ

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА  
по направлению подготовки  
02.04.02 – Фундаментальная информатика и информационные технологии,  
направленность программы  
«Разработка мобильных приложений и компьютерных игр»

**Научный руководитель** –   
 доц., к. ф.-м. н. Махно Виктория Викторовна

**Рецензент** –   
к. ф.-м. н. Кузнецова Елена Михайловна

Допущено к защите:  
руководитель   
образовательной программы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Демяненко Я.М.

Ростов-на-Дону – 2022

Оглавление

[Введение 3](#_Toc104812988)

[1.Application Programming Interface (API) 5](#_Toc104812989)

[1.1 Существующие виды API 7](#_Toc104812990)

[1.2 Протоколы и архитектуры API 8](#_Toc104812991)

[2.Выбор инструментов для разработки API 11](#_Toc104812992)

[3. Используемые средства для разработки API на облачной платформе. 16](#_Toc104812993)

[3.1 Понимание сервис-ориентированных архитектур 17](#_Toc104812994)

[3.2 Универсальность и современность Heroku 20](#_Toc104812995)

[3.3 Принцип работы облачной платформы 21](#_Toc104812996)

[4.Используемые средства для хранения данных для API 23](#_Toc104812997)

[5.Создание API для автоматизированного контроля сроков исполнения документов. 28](#_Toc104812998)

[5.1 Используемые фреймворки и библиотеки. 28](#_Toc104812999)

[5.2 Создание программы для сканирования документов 32](#_Toc104813000)

[5.3 Отправка информации об отсканированных документах в базу  
данных 36](#_Toc104813001)

[5.4 Создание API, используя фреймворк Flask 37](#_Toc104813002)

[5.5 Выгрузка API на облачную платформу Heroku 38](#_Toc104813003)

[Заключение 40](#_Toc104813004)

[Список литературы 41](#_Toc104813005)

**Введение**

В современном мире существует множество программ для планирования задач, которые очень помогают в повседневной жизни. Как правило, их функционал довольно примитивен. Ниже приведены стандартные возможности:

* Добавление/удаление задачи;
* Установка напоминания;
* Добавление приоритетов;
* Сохранение текущих задач (Необходимо, чтоб при обновлении программы все задачи сохранялись);

Для бытовых дел данного функционала должно вполне хватить, но его будет недостаточно, если, например, у пользователя программы на работе предусмотрены сроки сдачи документов, которых может быть очень много. При заведении задач по документам с напоминанием о сроке сдачи в силу человеческого фактора какой-нибудь из документов окажется пропущенным, а это в свою очередь может повлечь за собой немало трудностей. Для решения данной проблемы был создан удобный сервис для сбора полезной информации, которая необходима пользователю. Данный сервис расширит функционал программ для планирования задач и еще больше поможет пользователям. Обмен данными между программой для планирования задач и созданным в рамках данной работы сервисом будет происходить по средствам API (программного интерфейса).

# Цели и задачи работы

Целью данной выпускной магистерской работы было провести исследование в разработке API, а также создать API, содержащее данные с полезной информацией необходимой пользователю для отображения в программах планирования задач и опубликовать его на облачной платформе.

Задачами данной работы было:

* Исследовать функционал API;
* Настроить облачную платформу Heroku, для публикации API;
* Реализовать программу на Python, собирающую данные о документах из рабочей папки на компьютере пользователя и формирующее API;
* Подключить программу к базе данных MySQL;
* Опубликовать готовое API на облачную платформу Heroku;

# 1.Application Programming Interface (API)

API - это набор функций, позволяющих приложениям получать доступ к данным и взаимодействовать с внешними программными компонентами, операционными системами или микросервисами. Прежде всего, необходимо разобраться, по какому принципу происходит обмен данными (рис. 1).

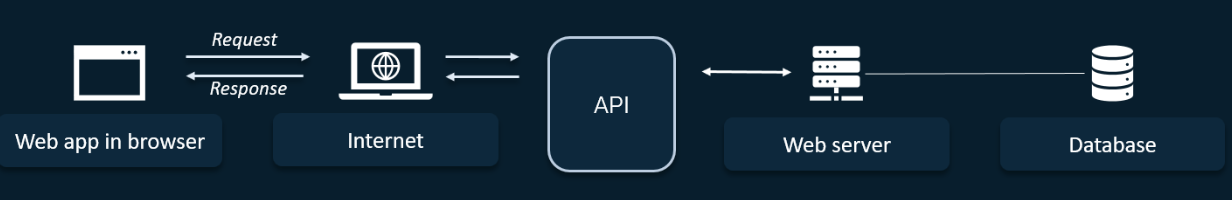
****

Рис. 1. Схема обмена данными

API состоит из двух частей:

* Техническая спецификация, описывающая возможности обмена данными между решениями, причем спецификация выполняется в виде запроса на обработку и протоколов доставки данных.
* Программный интерфейс, написанный в соответствии с представляющей его спецификацией.

Другими словами, API передает ответ пользователя системе и отправляет ответ системы обратно пользователю. Пользователь нажимает кнопку "подписаться на популярного писателя" в популярной социальной сети Твиттер; API сообщает сайту, что он планирует подписаться на другого объекта; сайт обновляет информацию, добавляет пользователю в подписки нового объекта и транслирует в ленту его популярные публикации.

Ранее, при определении понятия API, был упомянут термин «микросервисы». Однако API и микросервис это не одно и то же. Микросервисы — это стиль или архитектура, которая разделяет функциональность внутри веб-приложения. В то время как API — это основа, с помощью которой разработчики взаимодействуют с веб-приложением.

API позволяет разработчику сделать определенный "запрос", чтобы отправить или получить информацию. Эта связь осуществляется с помощью текстового формата данных под названием "JSON".

JavaScript Object Notation (JSON) — это легкий и простой в разборе текстовый формат для обмена данными. Каждый файл JSON содержит коллекции пар имя/значение и упорядоченные списки значений. Поскольку это универсальные структуры данных, формат может быть использован с любым языком программирования (Листинг 1.1).

Значения ключа могут быть любыми:

* число
* строка
* массив
* другой объект

Листинг 1.1. Пример JSON файла

{

    "name": "Ivan Ivanov",

    "id": 7,

    "personal\_data": {

        "age": 27,

        "username": "ivanov\_ivan",

        "email": "ivanov@newbox.com",

    },

    "subscribers": 1500,

    "subscriptions": 200,

    "publications": 138,

    "is\_banned": false,

}

## 1.1 Существующие виды API

С точки зрения политики безопасности, API могут быть частными, партнерскими и публичными. Рассмотрим каждый из них отдельно.

1. Частные API.

Частные API предназначены только для использования внутри предприятия для соединения систем и работы с данными. Например, частный API может соединять системы расчета заработной платы и управления персоналом организации.

1. Партнерские API.

Чаще всего партнерские API облегчения деятельности между предприятиями. Одна сторона формирует данные, а вторая приобретает доступ к этим данным у первой. Компания, создающая API, может контролировать, как используются их сервис и следить за тем, обеспечивают ли сторонние решения, использующие их API, достойный пользовательский опыт. Партнеры имеют четкие права и лицензии на доступ к таким API. По этой причине партнерские API обычно включают в себя достаточно надежные механизмы аутентификации, авторизации и безопасности.

1. Публичные API.

Публичный API является открытым и доступным для использования любым сторонним разработчиком или предприятием. Публичные API обычно предполагают умеренную аутентификацию и авторизацию. Предприятие также может попытаться монетизировать API, установив стоимость каждого вызова для использования публичного API.

## 1.2 Протоколы и архитектуры API

API обмениваются командами и данными, и это требует четких протоколов и архитектур — правил, структур и ограничений, которые регулируют работу API. Сегодня существует множество архитектурных стилей API и самые популярные из них это REST и SOAP. Их можно назвать "форматами", каждый из которых обладает уникальными характеристиками и компромиссами и используется для различных целей. Рассмотрим каждуюспецификаций, чтобы выяснить, какая из них больше всего подходит для решения поставленных задач:

1. SOAP API

Simple Object Access Protocol (SOAP) — это протокол веб-коммуникаций, который используется для обмена информацией и структурированными данными через HTTP/HTTPS. В отличие от REST API, SOAP поддерживает только формат данных XML. В основном обмен данными в Интернете осуществляется через REST, но это не означает, что SOAP скоро уйдет в прошлое, поскольку он высоко стандартизирован, более безопасен и в некоторых случаях позволяет автоматизировать процесс.

XML или eXtensible Markup Language — это независимый от программного или аппаратного обеспечения инструмент для хранения и транспортировки данных. Он имеет набор правил для структурирования сообщений в виде удобных как для человека, так и для машины записей. Его структура похожа на HTML, но в языке XML нет предопределенных тегов, поэтому автор должен определить как теги, так и структуру документа (Листинг 1.2).

Листинг 1.2. Пример структуры XML файла

<?xml version="1.0"?>

<catalog>

   <book id="book1">

      <author>Ivanov, Ivan</author>

      <title>About XML Format</title>

      <price>10.15</price>

      <genre>Computer</genre>

      <publish\_date>2020-10-01</publish\_date>

      <description>How to create an XML file</description>

   </book>

</catalog>

1. REST API

Архитектура репрезентативной передачи состояния (REST) является, пожалуй, самым популярным подходом к созданию API. REST опирается на подход клиент/сервер, который разделяет внешний и внутренний интерфейс API, и обеспечивает значительную гибкость в разработке и реализации. REST не является статичным, что означает, что API не хранит данные или статус между запросами. API REST, обычно называемые "RESTful API", также могут взаимодействовать напрямую или работать через промежуточные системы, такие как API-шлюзы и распределители нагрузки. Эти API используют HTTP-запросы для работы с ресурсами.

Существует четыре основных метода запроса:

* GET - Сбор информации (Получение всех данных).
* PUT - Обновление части данных (Обновление цен на товары)
* POST – Добавление нового элемента (Создание новой категории продукта).
* DELETE – Удаление элемента (Удаление записи в блоге).

В отличие от SOAP, REST не ограничивается использованием XML для предоставления ответа. Мы можем получить информацию в различных форматах, таких как JSON, XML, CSV, простой текст, HTML, которые легче разбираются в зависимости от языка, который вы используете для своего приложения.

Способность поддерживать множество форматов для хранения и обмена данными - одна из причин, по которой была выбрана архитектура REST в качестве создания API для нашей программы.

# 2.Выбор инструментов для разработки API

Для выполнения одной из поставленных целей выпускной работы, был выбран язык программирования Python.

Python — это язык программирования общего назначения, интерпретируемый и высокоуровневый. Он ориентирован на удобочитаемость и простой синтаксис. Python предлагает все функциональные возможности, которые могут понадобиться для решения задач программирования.

Сегодня это один из самых распространенных языков программирования в мире. Он находится на первом месте в списке лучших языков программирования по версии IEEE Spectrum (журнал, который издается Институтом инженеров электротехники и электроники) и в индексе TIOBE (индекс, который оценивает распространенность языков программирования, подсчитывая результаты поисковых запросов, содержащих название языка). Он используется для различных приложений, и это очень популярный язык.

Приведем ряд преимуществ и недостатков данного языка.

Преимущества Python:

1. Легко читать, изучать и писать код.

Python — это язык программирования высокого уровня, который имеет англоподобный синтаксис. Это облегчает чтение и понимание кода. Его очень легко освоить, поэтому многие рекомендуют его на начальном этапе. Для выполнения одной и той же задачи требуется меньше строк кода по сравнению с другими основными языками, такими как C/C++ или Java.

1. Повышенная продуктивность.

Данный язык программирования достаточно продуктивный. Благодаря его простоте можно сосредоточиться сразу на решении проблемы. Не нужно тратить слишком много времени на понимание синтаксиса или поведения языка программирования.

1. Интерпретируемый язык.

Python — интерпретируемый язык, что означает, что он напрямую выполняет код строка за строкой. В случае какой-либо ошибки он останавливает дальнейшее выполнение программы и сообщает об этом. Python показывает только одну ошибку, даже если в программе их несколько. Это в разы облегчает отладку.

1. Динамически типизированный.

Python не знает тип переменной, пока мы не запустим код. Он автоматически присваивает тип данных во время выполнения. Программисту не нужно беспокоиться об объявлении переменных и их типов данных.

1. Свободный и открытый исходный код.

Python поставляется по лицензии OSI с открытым исходным кодом. Это делает его свободным для использования и распространения. Можно скачать исходный код, изменить его и даже распространять свою версию Python. Это полезно для организаций, которые хотят изменить некоторые специфические функции и использовать свою версию для разработки.

1. Поддержка обширных библиотек.

Стандартная библиотека Python огромна, в ней можно найти практически все функции, необходимые для поставленной задачи. Таким образом, не придется зависеть от внешних библиотек. А если они понадобятся, то существует менеджер пакетов Python (pip), который облегчает процесс импорта из индекса пакетов Python (PyPi). Он состоит из более чем 200 000 пакетов.

1. Мобильность.

Во многих языках, таких как C/C++, для запуска программы на разных платформах необходимо изменять код. С Python дело обстоит иначе. Достаточно написать код только один раз и запустить её где угодно. Однако следует быть осторожным, чтобы не включить в программу какие-либо системно-зависимые функции.

Недостатки Python:

1. Неэффективность памяти.

Чтобы обеспечить простоту в использовании, Python вынужден идти на небольшой компромисс. Данный язык программирования использует большой объем памяти. Это может быть недостатком при создании приложений, в которых необходимо соблюсти оптимизацию памяти.

1. Доступ к базе данных.

Когда необходимо взаимодействовать с базой данных, ресурсов Python недостаточно. Уровень доступа к базе данных в Python примитивен и недостаточно развит по сравнению с такими популярными технологиями, как JDBC и ODBC. Огромные предприятия нуждаются в беспрепятственном взаимодействии со сложными унаследованными данными, поэтому для таких задач Python редко используют.

1. Ошибки во время выполнения.

Как известно, Python - динамически типизированный язык, поэтому тип данных переменной может измениться в любой момент. Переменная, содержащая целое число, в будущем может стать строкой, что может привести к ошибкам времени выполнения. В связи с этим необходимо проводить тщательное тестирование приложений.

Из всех перечисленных преимуществ и недостатков можно сделать вывод, что Python — это простой, универсальный и полноценный язык программирования. Он отлично подходит как для начинающих, так и для профессионалов. Хотя он и имеет некоторые недостатки, мы можем заметить, что преимущества их превышают. Даже компания Google сделала Python одним из своих основных языков программирования.

Для редактирования исходного кода программы была выбрана программа Atom.

Atom был разработан и выпущен компанией GitHub. Этот бесплатный редактор кода с открытым исходным кодом получил название «взламываемый текстовый редактор 21 века». И он действительно является взламываемым. Atom позволяет разработчикам полностью настраивать внешний вид, возможности и требования для ускорения рабочего процесса.

Однако Atom все еще позволяет разработчикам продуктивно использовать его, никогда не прикасаясь к конфигурационным файлам. Только что загруженная версия поставляется с восемью темами для подсветки синтаксиса и четырьмя UI: двумя светлыми и двумя темными. Но если ни одна из предустановленных тем пользовалтеля не заинтересовала, Atom позволяет легко и быстро установить пользовательские темы, созданные сторонними разработчиками, или создать их самостоятельно.

Поскольку наш проект будет опубликован на платформе Github и, чтобы упростить взаимодействие с данной платформой, был выбран бесплатный клиент под названием SourceTree.

Sourcetree — это мощный клиент для взаимодействия с Git, который Atlassian бесплатно предоставляет как для Windows, так и для Mac. Использование Sourcetree позволяет упростить взаимодействие с кодом, включая визуализацию, улучшая способ управления вашими репозиториями. Благодаря доступным подробным диаграммам ветвления можно легко следить за каждым обновлением, которое было сделано.

Для любой организации, которой требуется лучший контроль и взаимодействие с репозиториями кода и процессами сборки, тестирования и управления релизами, Sourcetree предоставляет необходимые средства контроля. Если компания начинает реализацию стратегии непрерывной доставки для конкретного продукта или нуждается в дополнительном понимании того, как меняется код во время коммитов и слияний, Sourcetree поможет достичь этих целей.

# 3. Используемые средства для разработки API на облачной платформе.

В качестве сервиса для хранения API был выбрана Heroku. Heroku — это облачная PaaS (Platform as a Service) платформа, популярность которой выросла в последние годы. Heroku настолько проста в использовании, что является лучшим выбором для многих проектов по разработке.

С особым вниманием к поддержке приложений, ориентированных на клиента, она обеспечивает простую разработку и развертывание приложений. Поскольку платформа Heroku управляет оборудованием и серверами, то компании, использующие её, могут сосредоточиться на совершенствовании своих приложений, а не на инфраструктуре, которая их поддерживает.

Прежде чем в полной мере оценить преимущества Heroku, необходимо понять историю облачных вычислений и то, какое место в ней занимает Heroku.

Облачные вычисления — это информационно-технологическая парадигма. Это модель для обеспечения повсеместного доступа к общим ресурсам, таким как компьютерные сети, серверы, хранилища, приложения и сервисы.

Облачные сервисы могут быть быстро предоставлены с минимальными усилиями по управлению со стороны предприятия. Облако - привлекательный вариант для предприятий с самыми разными потребностями. Однако особенности бизнес-модели и потребности проекта определят, какого поставщика облачных сервисов они в итоге используют.

Heroku, платформа как услуга, в целом проста в использовании. Но она наиболее полезна для предприятий в конкретных ситуациях. У Heroku есть бесплатная модель обслуживания для небольших проектов. Кроме того, существуют многоуровневые пакеты услуг для случаев, когда необходимо решить более сложные бизнес-задачи.

## 3.1 Понимание сервис-ориентированных архитектур

Сервис-ориентированные могут быть разделены по определенному назначению. Наиболее примечательными являются Infrastructure-as-a-Service (IaaS, «инфраструктура как услуга»), Platform-as-a-Service (PaaS, «платформа как услуга») и Software-as-a-Service (SaaS, «программное обеспечение как услуга»).

Рассмотрим каждый из них подробнее:

1. Infrastructure-as-a-Service.

Infrastructure-as-a-Service— это мгновенная вычислительная инфраструктура, которая обслуживает, управляет и контролирует через Интернет. Она может изменяться в зависимости от требований, и клиент должен платить только за то, что он использовал. IaaS может увеличиваться и уменьшаться в зависимости от требований, поэтому клиент не платит никаких дополнительных сборов.

IaaS снижает нагрузку на управление и обслуживание серверов, поскольку инфраструктура предоставляется компанией. Каждый ресурс имеет отдельный компонент, и клиент может арендовать его в соответствии с требованиями.

Полное управление осуществляется поставщиком облачных услуг. Установка, настройка и управление программным обеспечением полностью возлагаются на клиента.

Рассмотрим примеры данной инфраструктуры:

* Бизнес-сети.

В бизнес-сети объединенные серверы и сетевые ресурсы, которые использует бизнес, будут хранить информацию и запускать приложения. Растущие предприятия будут масштабировать свою инфраструктуру в соответствии с ростом.

* Облачный хостинг.

В облачном хостинге интернет-сайты размещаются на виртуальных серверах, которые поддерживают ресурсы из базовых физических серверов.

* Виртуальный центр обработки данных.

Существует виртуальная сеть подключенных серверов, которая улучшает возможности облачного хостинга, ИТ-инфраструктуру предприятия или интегрирует операции.

Также, отличным примером такой архитектуры является популярный Microsoft Azure.

1. Platform-as-a-Service

Платформа как услуга (PaaS) — это полноценная, масштабируемая среда разработки и развертывания, которая предоставляется по подписке. Основное преимущество модели PaaS заключается в том, что она позволяет пользователям получить доступ к аппаратному и программному обеспечению, которое можно использовать для разработки и запуска приложений без необходимости приобретения, установки и обслуживания инфраструктуры. Аналогичным образом, организация может использовать PaaS для расширения или изменения архитектуры существующих приложений в облаке.

1. Software-as-a-Service

SaaS является одним из трех основных компонентов облачных вычислений, которые являются одним из основополагающих элементов цифровой трансформации. Услуги SaaS предлагают пользователям использовать приложения разработчика, работающие на облачной инфраструктуре. Приложения доступны с различных клиентских устройств через клиентский интерфейс, такой как веб-браузер, мобильные приложения или программный интерфейс.

В некоторых случаях пользователи имеют ограниченный доступ к изменению файла настроек приложений. Но они все так же не могут изменять общую, базовую архитектуру и код приложений SaaS. Доступ к услугам в группе SaaS гораздо более ограничен, чем доступ к услугам в группах PaaS или IaaS.

Важными примерами SaaS, используемых многими компаниями, являются HubSpot, Google Apps и GoToMeeting.

## 3.2 Универсальность и современность Heroku

Платформа облачных сервисов Heroku основывается на системе управляемых контейнеров (в парадигме Heroku они называются dynos). У Heroku есть бесплатная модель обслуживания для небольших проектов. Кроме того, существуют многоуровневые пакеты услуг для случаев, когда необходимо решить более сложные бизнес-задачи.

Heroku поддерживает несколько языков программирования, которые используются в качестве модели развертывания веб-приложений. Будучи одной из первых облачных платформ, Heroku находится в разработке с июня 2007 года. Тогда она поддерживала только язык программирования Ruby. Но сейчас она также поддерживает множество языков программирования, таких как: Java, Node.js, Python и так далее. Это означает, что самые разные разработчики могут обратиться к Heroku за недорогим способом масштабирования своих приложений, независимо от предпочитаемого языка разработки.

Heroku предоставляет возможность создания собственных билдов, в которых пользователи могут разместить приложения на любом из других языков программирования. По этой причине Heroku является универсальной платформой. Она позволяет разработчикам создавать, запускать и масштабировать приложения одинаковым образом на всех языках программирования.

Полиморфизм и масштабируемость — это причины, по которым Heroku часто рассматривается как предпочтительная платформа среди разработчиков.

## 3.3 Принцип работы облачной платформы

Приложения, запускаемые на Heroku, обычно имеют уникальные доменные имена, которые используются для маршрутизации HTTP-запросов к нужному контейнеру. Приложения как сервисы используют контейнеры приложений.

Контейнеры предназначены для упаковки и запуска сервисов. Каждый из контейнеров приложений представляет собой интеллектуальный контейнер на надежной, полностью управляемой среде выполнения. Контейнеры приложений - называемые "dyno (дино или диносы)" в контексте платформы Heroku - распределены по "dyno-сетке". Она состоит из нескольких серверов. Менеджер Dyno поддерживает и управляет созданными dynos.

Все это означает, что поскольку Heroku управляет и запускает приложения, нет необходимости управлять операционными системами или другими внутренними конфигурациями системы.

Проекты, созданные в Heroku, привязаны к репозиториям в GitHub. Интеграция Heroku с GitHub обеспечивает автоматическую сборку и развертывание последней версии кода.

GitHub — это надежное хранилище исходного кода, поэтому интеграция Heroku с GitHub и другими инструментами помогает разработчикам оптимизировать свои усилия и экономить время и деньги заинтересованных сторон в ходе реализации проектов разработки.

# Используемые средства для хранения данных для API

В качестве хранения информации об отсканированных документах была выбрана система хранения баз данных MySQL.

MySQL — это РСУБД (расшифровывается, как реляционная система управления базами данных), реализующая SQL. Первоначально она была разработана для использования с базами данных малого и среднего размера, но теперь она может обрабатывать даже очень большие объемы хранимых данных. MySQL написана на языке C/C++ и в основном соответствует стандарту SQL. Однако в него добавлено множество расширений, а скорость и надежность важнее идеального соответствия. Более подробное обсуждение соответствия MySQL и SQL можно найти в документации MySQL по стандартам соответствия.

Базовая версия MySQL распространяется корпорацией Oracle и доступна бесплатно по лицензии с открытым исходным кодом. Текущий релиз MySQL можно использовать на любом дистрибутиве Linux и на большинстве других платформ таких, как Windows или же MacOS. Он является важным компонентом стека LAMP с открытым исходным кодом, наряду с Apache и языком программирования PHP. Стек LAMP является краеугольным камнем разработки веб-приложений с открытым исходным кодом на Linux. MySQL может использоваться как часть системы клиент/сервер или как часть встроенной системы.

Как и все приложения РСУБД, MySQL является реляционной базой данных. Администраторы и пользователи определяют отношения внутри и между таблицами в базе данных. Различные столбцы могут быть отмечены как обязательные или необязательные и могут служить первичным ключом или указателем на другую таблицу. MySQL является стабильной, надежной и простой в использовании.

Приведем некоторые конкретные преимущества MySQL:

* MySQL — это развитый, популярный и хорошо зарекомендовавший себя продукт. Он имеет большую поддержку сообщества, чем другие базы данных, а также содержит отличные справочные материалы и документацию.
* MySQL может быть настроен на ACID-совместимость при использовании с системой хранения InnoDB. Аббревиатура ACID расшифровывается как Atomicity, Consistency, Isolation, and Durability. Это означает, что она может гарантировать достоверность данных, несмотря на любые ошибки или сбои в работе.
* Данная система известна своей высокой производительностью благодаря таким функциям, как оптимизация библиотек классов, сжатие, распределение памяти и хэш-таблицы. Она поддерживает многопоточное исполнение процессов ядра для более эффективной работы на системах с несколькими процессорами.
* Поддерживаются очень большие базы данных, хранящие до сотен миллионов записей и до 64 индексов на таблицу.
* MySQL предоставляет расширенные средства безопасности, включая шифрование всех паролей.
* Позволяет осуществлять репликацию и резервирование данных для повышения надежности.
* MySQL обеспечивает поддержку алиасов для таблиц и столбцов, а также полную поддержку множества различных наборов символов.
* MySQL хорошо работает с PHP, который широко используется в веб-разработке.
* Работает со многими различными компиляторами и на многих платформах и разработан для переносимости между системами. Клиентские программы могут быть написаны на многих языках.
* MySQL предоставляет API для C/C++, PHP, Java, Python, Ruby и Perl.
* MySQL доступен в виде библиотеки, которая может быть встроена в отдельные приложения.
* MySQL поставляется в комплекте с несколькими удобными клиентскими утилитами, включая mysqldump и mysqladmin. Пользователи могут проверять, оптимизировать и исправлять таблицы с помощью программы mysqlcheck.
* Лицензия MySQL с открытым исходным кодом позволяет разработчикам настраивать данную систему и изменять исходный код в соответствии со своими требованиями.

MySQL — это универсальная СУБД для работы с набором данных любого размера. Ее можно использовать в любой момент, когда приложению необходимо хранить и извлекать данные. Первоначально MySQL была разработана для малых и средних конфигураций с одним сервером. Но благодаря недавним улучшениям производительности и масштабируемости, ее можно использовать практически везде в приложениях любого размера. MySQL используют даже крупные компании, включая Uber, Airbnb и Shopify.

Другие распространенные области применения MySQL включают хранилища данных, обработку транзакций, системы бронирования, электронную коммерцию и веб-базы данных. Например, база данных MySQL может поддерживать список товаров и инвентаризацию для интернет-магазина.

Для полноценного использования MySQL нам необходимо разобраться в её клиент-серверной части. Клиент MySQL и сервер MySQL — это два различных компонента, которые работают вместе в сетевой архитектуре. Обычно имеется один центральный сервер и один/несколько клиентов. Сервер базы данных MySQL устанавливается на хосте, часто как часть стека LAMP. Этот сервер хранит конфигурацию базы данных и данные, а также отвечает на запросы клиентов. Он также обеспечивает безопасность и любую систему контроля доступа, реплицирует и архивирует данные по мере необходимости. Один и тот же сервер может содержать несколько баз данных для разных клиентов.

Клиент MySQL позволяет пользователям подключаться к серверу MySQL, находящемуся в той же системе или на другом сервере. Клиент посылает SQL-запросы на сервер для чтения из базы данных или записи в нее. Клиент MySQL также администрирует, обслуживает и обеспечивает безопасность клиента. Автономный клиент рекомендуется для пользователей, которые хотят подключаться к удаленной базе данных только для выполнения запросов. Администраторы, которые хотят разместить базу данных на сервере, должны установить полный пакет сервера MySQL. Установка сервера включает в себя клиент для создания и администрирования базы данных.

Стандартная клиентская утилита командной строки MySQL называется mysql. Она может быть установлена без серверного компонента с помощью команды «yum install mysql» или «apt-get install mysql-client». Чтобы получить доступ к клиенту MySQL, достаточно ввести команду «mysql <имя\_базы\_данных>». Имя пользователя, пароль и IP-адрес сервера могут быть указаны с помощью дополнительных параметров.

Когда пользователь успешно войдет в систему, клиент отобразит страницу MySQL. После этого пользователь может выполнять команды SQL.

# 5.Создание API для автоматизированного контроля сроков исполнения документов.

## 5.1 Используемые фреймворки и библиотеки.

Одной из основных фреймворков Python, используемых в разработке API, является Flask.

Flask занимает одно из первых мест в списке самых востребованных веб-фреймворков согласно Stack Overflow и это замечательный результат для того, кого называют микрофреймворком. Почему именно так? По сути, потому что для своей работы он не зависит от других инструментов или библиотек. Flask чуть больше 11 лет, он примерно такого же возраста, как и более известный веб-фреймворк на языке Python - Django.

Flask — это так называемый фреймворк WSGI. В переводе означает "интерфейс шлюза веб-сервера". По сути, это способ для веб-серверов передавать запросы веб-приложениям или фреймворкам. Flask опирается на внешнюю библиотеку WSGI, а также на механизм шаблонизатора Jinja2.Перечислим несколько преимуществ и недостатков Flask.

Преимущества Flask:

1. Масштабируемый.

Статус Flask как микрофреймворка и его размер означают, что можно использовать его для невероятно быстрого роста такого технологического проекта, как веб-приложение. Если необходимо создать приложение, которое начинается с малого, но имеет потенциал для быстрого роста в различных направлениях, которые еще не до конца проработаны, то это идеальный выбор. Простота использования и небольшое количество зависимостей позволяют ему работать без сбоев даже при увеличении масштаба.

1. Гибкий.

Это основная особенность Flask и одно из его самых больших преимуществ. Перефразируя один из принципов Zen of Python, можно сказать, что простота лучше сложности, потому что ее можно легко переставлять и перемещать. Это не только полезно с точки зрения того, что ваш проект может легко двигаться в другом направлении, но и гарантирует, что структура не разрушится при изменении какой-либо части. Минимальный набор компонентов Flask и его пригодность для разработки небольших веб-приложений означает, что он даже более гибок, чем сам Django.

1. Прост в навигации.

Как и в случае с Django, возможность легко ориентироваться в нем является ключевым фактором, позволяющим веб-разработчикам сосредоточиться на быстром кодировании, не увязая в нем. По своей сути микрофреймворк прост для понимания веб-разработчиков, что не только экономит их время и усилия, но и дает им больше контроля над своим кодом и возможными возможностями.

1. Легковесный.

Когда мы используем этот термин по отношению к инструменту или фреймворку, мы говорим о его конструкции - в нем мало составных частей, которые нужно собирать и пересобирать, и он не зависит от большого количества расширений для функционирования. Такая конструкция дает веб-разработчикам определенный уровень контроля. Flask также поддерживает модульное программирование, то есть его функциональность может быть разделена на несколько взаимозаменяемых модулей. Каждый модуль действует как независимый строительный блок, который может выполнять одну часть функциональности. В совокупности это означает, что все составные части структуры сами по себе являются гибкими, подвижными и тестируемыми.

1. Документация.

Следуя теории создателя о том, что "хороший дизайн документации заставляет вас действительно писать документацию", пользователи Flask найдут большое количество примеров и советов, расположенных в структурированном виде. Это стимулирует разработчиков к использованию фреймворка, поскольку они могут легко ознакомиться с различными аспектами и возможностями инструмента. Документацию по Flask можно найти на официальном сайте.

Недостатки Flask:

1. Недостаточно инструментов разработки.

Легкость этого микрофреймворка неизбежно имеет некоторые недостатки. Главным из них является то, что в отличие от Django, Flask не имеет большого набора инструментов. Это означает, что разработчикам придется вручную добавлять расширения, такие как библиотеки. И если вы добавите огромное количество расширений, это может привести к замедлению работы самого приложения из-за множества запросов.

1. Сложность ознакомления с большим приложением, написанным на Flask.

Из-за того, что разработка веб-приложения с использованием Flask может принимать различные повороты и изгибы, веб-разработчику, пришедшему в проект на середине пути, будет трудно понять, как он был разработан. Модульная природа микрофреймворка, о которой мы говорили ранее, может снова преследовать программистов, которым придется знакомиться с каждой составной частью.

Взвесив все преимущества и недостатки данного микрофреймворка, можно сделать вывод, что с Flask легко начать работу, потому что он не требует больших усилий для обучения. Кроме того, он очень понятен, что повышает читабельность.

Для парсинга дат сдачи в документе в формате .docx был использован python пакет под названием docx2txt. Данный пакет позволяет извлекать текст и изображения из документов Word. Мы можем прочитать документ, используя метод пакета под названием process, который принимает имя файла в качестве входных данных. Обычный текст, элементы списка, текст гиперссылок и текст таблицы будут возвращены в виде одной строки.

А для документов в формате .pdf была использована библиотека PyPDF2. Используя данную библиотеку, можно совершать множество различных типов операций с PDF.

Используя этот пакет, можно:

* Извлекать нужные данные из файла
* Менять ориентацию страниц
* Объединять файлы
* Разделять файлы
* Добавлять водяные знаки
* Шифровать файл

Также были задействованы такие python модули, как «OS» и «RE (regular expression)».

Модуль OS в Python является частью стандартной библиотеки языка программирования и поставляется с множеством различных функций, которые позволяют разработчикам взаимодействовать с операционной системой (Macintosh, Windows, Linux), над которой они в данный момент работают.

Модуль "RE" как раз предоставляет набор функций, позволяющих искать совпадения в строке по регулярным выражениям, в которых прописана последовательность искомых символов или шаблонов.

## 5.2 Создание программы для сканирования документов

Перед тем, как непосредственно начать писать программу на языке Python, при помощи командной строки были установлены пакеты, библиотеки и микрофреймворк, описанные ранее. Для простоты установки каждого из инструментов, в Python существует пакетный менеджер под названием Pip. Данный пакетный менеджер создан для того, чтобы устанавливать сторонние пакеты и библиотеки, которые изначально не входят в Python.

Перечислим команды, которые использовались для установки сторонних пакетов, модулей и библиотек:

* Для docx2txt — «pip install docx2txt»
* Для PyPDF2 — «pip install PyPDF2»
* Для Flask — «pip install flask» и «pip install flask-restful»
* Для MySQL — «pip install mysql-connector-python»

После установки был создан файл под названием «document-service» с расширением «.PY».

Для поддержания актуальности версий программы была настроена связь нашего локального проекта с Github. Для этого создается репозиторий в личном кабинете. После создания репозитория, предоставляется возможность «склонировать» его на собственный компьютер используя команду «git clone github.com/ {имя\_пользователя}/ document-service.git».

После клонирования репозитория автоматически создается папка, в которую нам нужно положить Python файл, который мы создали ранее.

Также к локальному проекту был настроен клиент SourceTree для удобства взаимодействия с GitHub. После добавления репозитория, данный клиент покажет, что в нем произошли изменения, поскольку ранее в папку был перенесен файл под названием «document-service».

Для отправки внесенных изменений в удаленный репозиторий, необходимо перенести изменения из состояния «Unstaged» в «Staged». Далее нажимаем на кнопку «Commit», пишем комментарий о том, что было сделано и нажимаем на кнопку «Push».

Теперь, когда настроена связь с Github, перейдем непосредственно к разработке программы.

В начале файла пропишем все пакеты и библиотеки, которые мы установили ранее (Листинг 5.2.1).

Листинг 5.2.1. Подключение установленных пакетов и библиотек

from flask import Flask

from flask\_restful import Api, Resource, reqparse

import docx2txt

from pathlib import Path

import mysql.connector

import os

import re

Далее создадим функцию, которая будет принимать в себя список документов, которые в данный момент находятся в рабочей папке, а также пропишем список правил по которому будет происходить фильтрация текста (Листинг 5.2.2).

Листинг 5.2.2. Функция files\_enum() для анализа документов в папке

rules = ['срок сдачи до', 'выполнить до', 'подготовить ответ к', 'выполнить до', 'сообщить до', 'согласовать до']

def files\_enum(folder):

    files = os.listdir(folder)

    output = {'files': []}

    id = 1

    for document in files:

        date = read\_docs(str(folder) + '/' + document)

        obj = {

            'name': document,

            'date': date,

            'visible': True,

            'id': id

        }

        output['files'].append(obj)

        id = id + 1

    return output['files']

if \_\_name\_\_ == '\_\_main\_\_':

    files\_enum(Path(\_\_file\_\_).parent/'university-documents')

Для определения даты сдачи документа была создана функция «read\_docs», которая принимает в качестве аргумента путь к документу (Листинг 5.2.3).

Листинг 5.2.3. Функция для получения даты сдачи

def read\_docs(path):

filename, file\_extension = os.path.splitext(path)

date = '';

if file\_extension == '.docx':

date = parse\_docx(path)

if file\_extension == '.txt':

date = parse\_txt(path)

return date

Далее необходимо под каждый формат документа написать свою функцию парсинга данных:

1. Для формата TXT (Листинг 5.2.4):

Листинг 5.2.4. Функция для парсинга файла формата TXT

def parse\_txt(path):

    text = open(path).read()

    date = ''

    for rule in rules: # Перебор правил

        if rule in text:

            date = text[text.index(rule):]

            date = str(re.search(r'\d\d.\d\d.\d\d\d\d', date).group())

    return date

1. Для формата DOCX (Листинг 5.2.5):

Листинг 5.2.5. Функция для парсинга файла формата DOCX

def parse\_docx(path):

    text = docx2txt.process(path)

    date = ''

    for rule in rules:

        if re.search(rule, text):

            date = text[text.index(rule):]

            date = str(re.search(r'\d\d.\d\d.\d\d\d\d', date).group())

    return date

## 5.3 Отправка информации об отсканированных документах в базу данных

Для того, чтобы настроить взаимодействие написанной программы с базой данных, был создан объект соединения, используя метод mysql.connector.connect() , который принимает в себя следующие параметры: имя пользователя, пароль, хост (необязательно, по умолчанию: localhost) и базу данных (необязательно). Затем был создан объект курсора, который вызывает метод cursor() для объекта подключения, созданного ранее .

Чтобы выполнять запросы к уже подключенной базе данных, достаточно вызвать метод execute() и написать нужный запрос к таблице (Листинг 5.3.1).

Листинг 5.3.1. Подключение БД к программе сканирования документов

mydb = mysql.connector.connect(

    host="eyvqcfxf5reja3nv.cbetxkdyhwsb.us-east-1.rds.amazonaws.com",

    user="gjnk9aymcgsx16c8",

    password="wctb0rl370s5b602",

    database="e7aariv527gxjb0w"

)

mycursor = mydb.cursor()

mycursor.execute(f"""select name, date from documents""")

## 5.4 Создание API, используя фреймворк Flask

## 

## Объект Flask реализует приложение WSGI и действует как центральный объект. Ему передается имя модуля или пакета приложения. После создания он будет действовать как центральный реестр для функций просмотра, правил URL, конфигурации шаблона и многого другого.

## Flask-RESTful предоставляет базовый класс «Resource» и «Api». Resource может определять маршрутизацию для одного или нескольких методов HTTP для заданного URL-адреса. Класс Api нужен непосредственного для того, чтобы создать API (Листинг 5.4.1).

Листинг 5.4.1. Создание API, используя Flask

app = Flask(\_\_name\_\_)

api = Api(app)

api.add\_resource(Documents, "/document-service", "/document-service/", "/document-service/<int:id>")

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True, use\_reloader=False)

Метод add\_resource() регистрирует маршруты во фреймворке, используя заданную конечную точку. Если конечная точка не указана, то Flask-RESTful генерирует ее за вас из имени класса. В качестве параметров, данный метод может принимать в себя:

* resource (Type[Resource]) – имя класса ресурса
* urls (строка) – один или несколько url-маршрутов для поиска ресурса, применяются стандартные правила маршрутизации flask. Любые переменные url будут переданы методу resource в качестве args.
* endpoint (строка) – имя конечной точки (по умолчанию Resource.\_\_name\_\_.lower()) Может использоваться для ссылки на этот маршрут в полях fields.Url
* resource\_class\_args (кортеж) – аргументы (args), которые будут переданы в конструктор ресурса.
* resource\_class\_kwargs (словарь) – аргументы (kwargs), которые будут переданы конструктору ресурса.

Поскольку первым параметром в методе add\_resource() передается имя класса ресурса, был создан класс Documents, в котором описаны основные методы для HTTP запросов:

* Метод get(). При обращении к API вернет либо весь список документов, либо один (если указанный id совпадет с номером из базы данных), либо вернет ошибку о том, что файл не найден (Листинг 5.4.2):

Листинг 5.4.2. Реализация метода get()

def get(self, id=0):

files\_enum(Path(\_\_file\_\_).parent/'university-documents')

files\_list = get\_database\_data()

if id == 0:

return files\_list, 200

for file in files\_list:

if file['id'] == id:

return file, 200

return 'File not found', 400

* Метод post(). При обращении можно добавить новый файл. При попытке добавить повторяющийся файл API выдаст ошибку (Листинг 5.4.2):

Листинг 5.4.2. Реализация метода post()

def post(self, id):

      files\_enum(Path(\_\_file\_\_).parent/'university-documents')

      current\_files = get\_database\_data()

      parser = reqparse.RequestParser()

      parser.add\_argument("name")

      parser.add\_argument("date")

      parser.add\_argument("visible")

      params = parser.parse\_args()

      for file in current\_files:

          if (id == file["id"]):

              return f"File with id {id} already exists", 400

          file = {

              "id": int(id),

              "name": params["name"],

              "date": params["date"],

              "visible": params["visible"]

          }

          current\_files.append(file)

      return file, 201

* Метод delete(). При обращении удаляет файл по указанному id (Листинг 5.4.3):

 Листинг 5.4.3. Реализация метода delete()

    def delete(self, id):

        files\_enum(Path(\_\_file\_\_).parent/'university-documents')

        current\_files = get\_database\_data()

        current\_files = [file for file in current\_files if file["id"] != id]

        return f"File with id {id} is deleted.", 200

## 5.5 Выгрузка API на облачную платформу Heroku

Для выгрузки созданного API на облачную платформу Heroku в папке проекта были созданы следующие необходимые файлы:

* requirements.txt — файл, который определяет, какие модули используются в проекте (Листинг 5.5.1)

Листинг 5.5.1. Файл requirements.txt

Flask==2.1.2

Flask-RESTful==0.3.9

gunicorn==20.1.0

docx2txt==0.8

pathlib==1.0.1

mysql.connector==2.2.9

* Procfile — определяет команды, выполняемые приложением при запуске.

Для соединения Heroku с проектом необходимо в консоли прописать следующие команды:

1. «heroku login»
2. «heroku create sfedu-document-service --buildpack heroku/python»
3. «git init»
4. «git add .»
5. «git commit -m “Add new app”»
6. «git push heroku master»

Результат выполненных команд можно увидеть непосредственно на сайте платформы Heroku (Рис. 2).

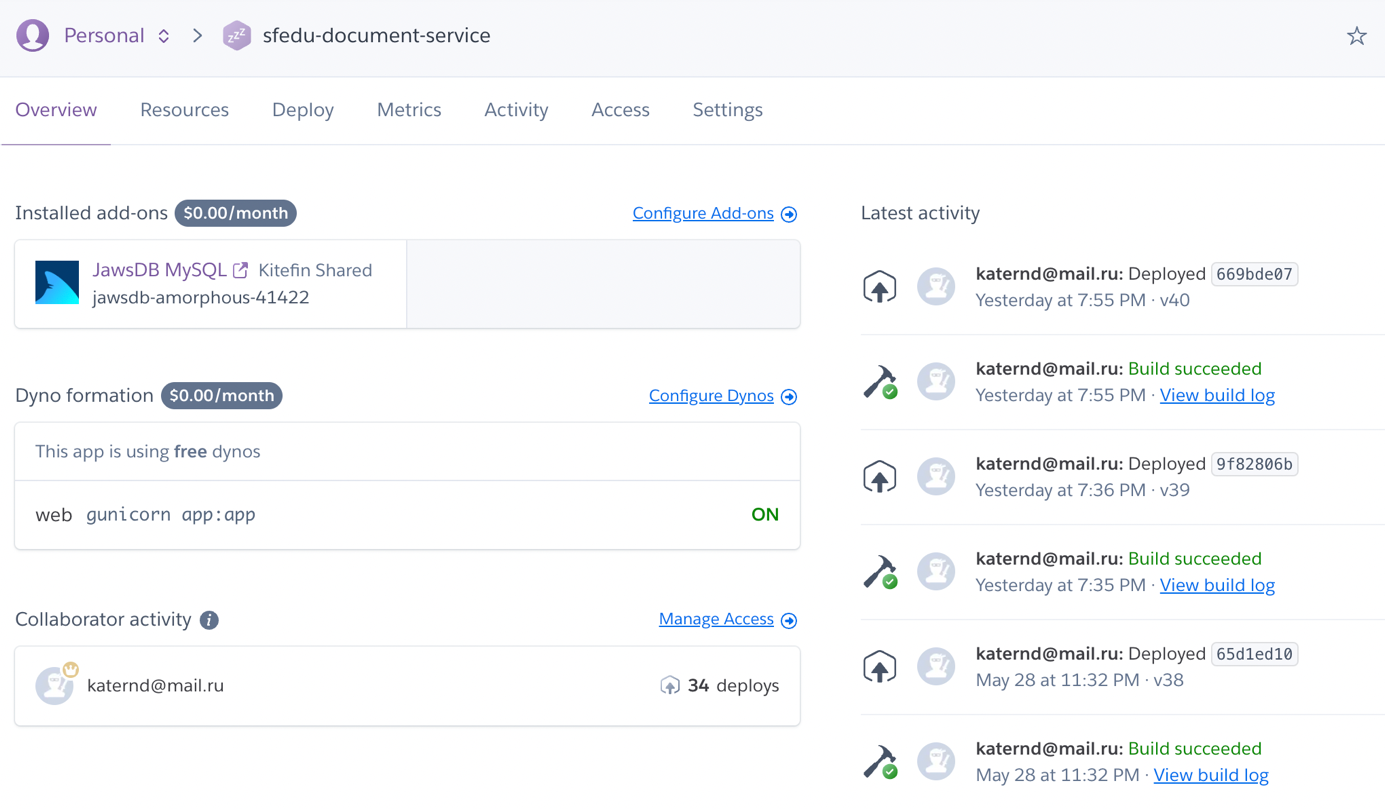


Рис. 2. Результат публикации API в Heroku

# Заключение

# Список литературы