Санкт-Петербургский государственный электротехнический университет им. В.И. Ульянова (Ленина)

Разработка сервиса обработки данных от системы прокторинг

Выполнил: Государкин Я.С., гр. 7382
Руководитель: Кринкин К.В., к.т.н., доцент

Санкт-Петербург, 2021

Цель и задачи

Актуальность: рост спроса в 2020-2021 годах на инструменты прокторинга

Цель: Разработать сервис для формирования статистики нарушений во время экзамена

Задачи:

1. Исследование существующих на момент первой половины 2021 года прокторинг-систем, сравнительный анализ и выявление проблем
2. Формирование требований к разрабатываемому сервису
3. Разработка архитектуры сервиса
4. Программная реализация и демонстрация сервиса
5. Анализ характеристик решения, сравнение с аналогами, апробация

Критерии отбора аналогов

Аналог -- позволяет автоматически оценивать действия и результаты ученика, после чего предоставлять эту оценку преподавателю.

Критерии:

- Спектр инструментов прокторинга, предоставляемых аналогом. Наиболее важный критерий при отборе и сравнении.
- Где происходит формирование оценки аналога – на компьютере студента, преподавателя или удаленном сервере.
- Когда происходит формирование оценки аналога – во время прохождения сессии дистанционной защиты или после нее.

Обзор существующих прокторинг-систем

Название аналога	Фиксирование нарушений на видео с вебкамеры	Фиксирование нарушений на видео с экрана	Фиксирование нарушений на звуковой дорожке с микрофона	Машина формирования оценки и обработки данных	Время формирования оценки сессии
Proctortrack	+	-	+	Компьютер студента	В реальном времени
ProctorExam	+	-	+	Компьютер студента	В реальном времени
Examus	+	+	+	Компьютер студента	В реальном времени
ProctorEdu	+	-	-	Удаленный сервер	После сессии
Proctoru	+	-	+	Компьютер студента	В реальном времени
Proctorio	+	-	-	Компьютер студента	В реальном времени

Проблема аналогов

Из всех рассмотренных аналогов больше всего инструментов реализовано в Examus и ProctorEdu: 7 из 11, остальные аналоги показали меньшее количество.

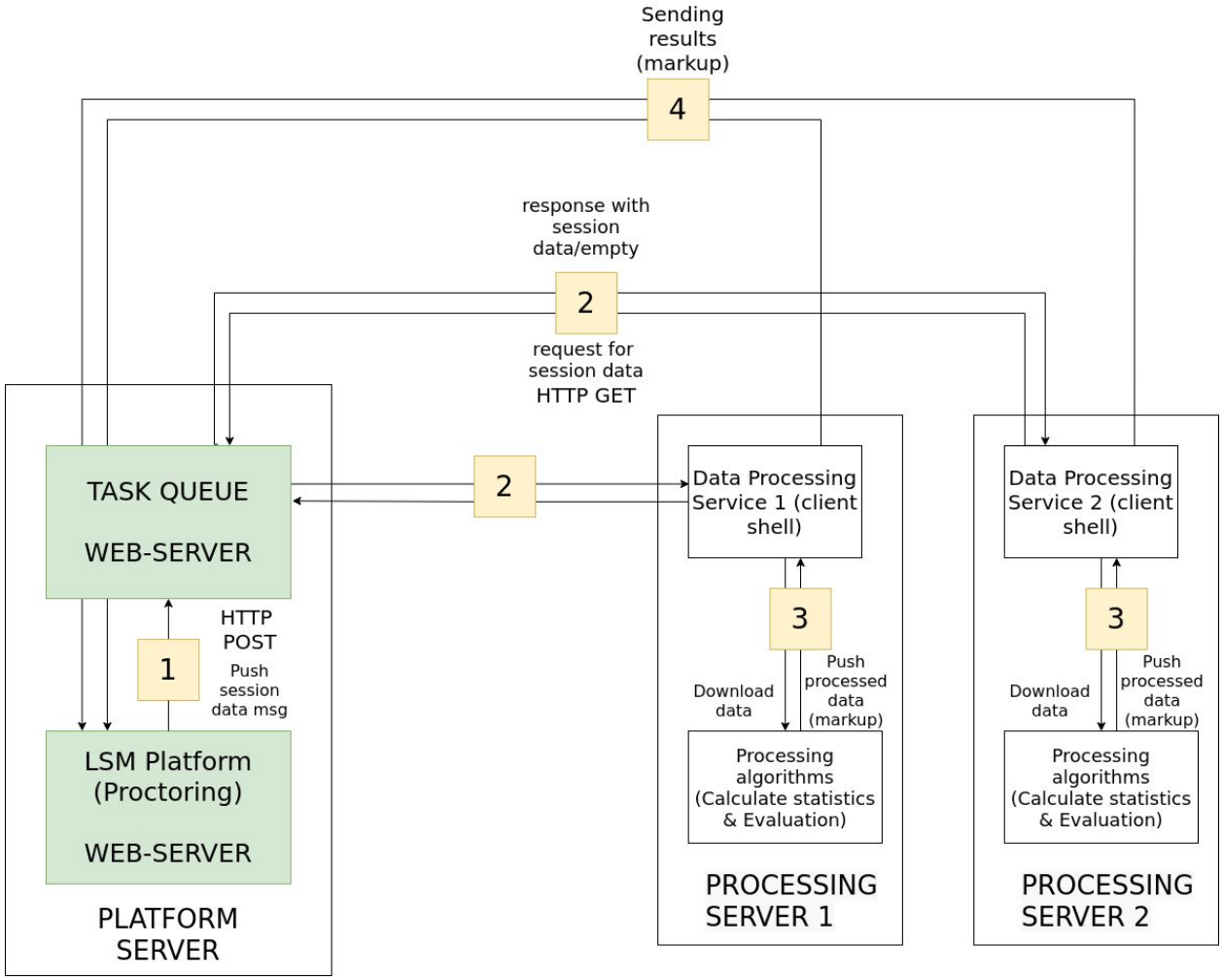
Таким образом основная проблема аналогов – ограниченность инструментария.

Требования к решению

Сервис обработки данных — отдельный от прокторинг платформы процесс

- Реализовать цикл обмена данными (получение данных, отправка статистики нарушений) с прокторинг-системой помощью REST API
- Обеспечить универсальный интерфейс для разработчиков алгоритмов формирования оценок и статистики
- Исключить возможность «подмены» сервиса обработки с помощью предварительной регистрации и авторизации сервиса в прокторинг системе
- Обеспечить масштабирование сервисов с помощью Docker и docker-compose
- Система логгирования статуса всех подключенных к прокторинг-системе сервисов обработки
- Unit-тестирование pull-функционала сервиса

Схема обмена данными



Универсальность сервиса

Формирование статистики происходит в отдельном модуле, который подключается к сервису как динамическая библиотека.

Подключение: на вход сервис получает путь к модулю обработки (конкретная python-функция).

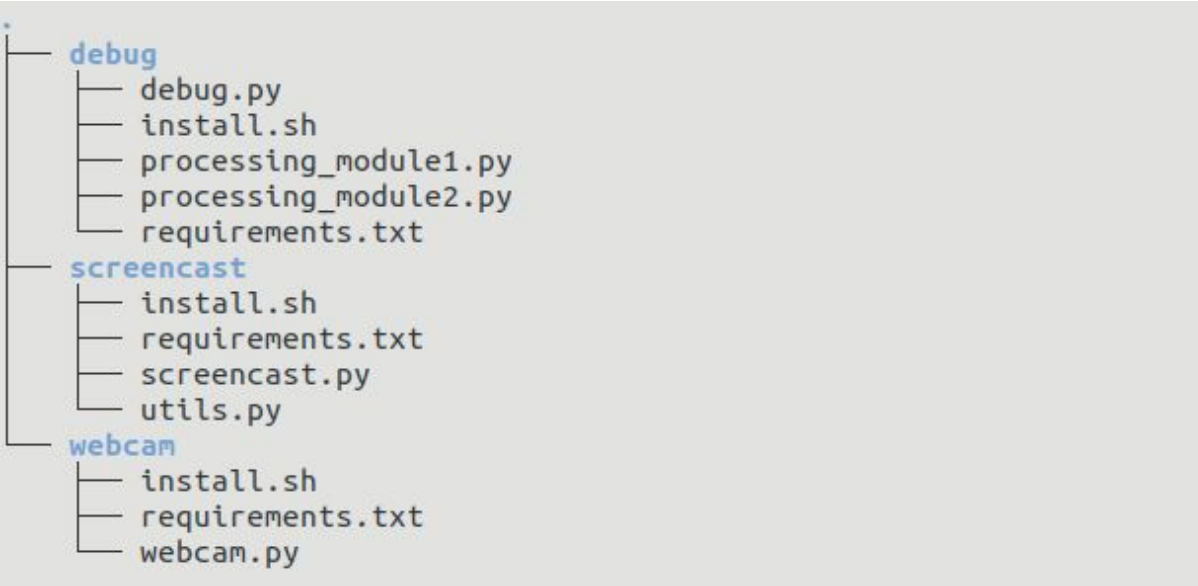
Функция-обработчик должна возвращать:

- result – тип dictionary, результат обработки данных, содержащий статистику нарушений, время обработки, код ошибки.
- msg – тип str, сообщение об ошибке возникшей внутри модуля обработки или успешности обработки.

Модуль обработки, пример конфигурации

Модуль обработки состоит из:

- requirements.txt – для установки пакетов из репозиториев pip
- install.sh – bash-скрипт для более сложной настройки зависимостей
- python-скриптов, реализующих алгоритмы формирования статистики.



Регистрация и авторизация сервиса

Регистрация сервиса производится следующим образом: в базу данных прокторинг платформы предварительно записывают уникальный client_id и client_secret сервиса. Пример:

Client_ID = “123”

Client_Secret = “sovershenno_secrentno”

Эти данные сохраняются в конфигурационном файле сервиса перед развертыванием. Сервис отправляет POST HTTP-запрос на платформу со своими client_id и client_secret, и если платформа находит в базе данных именно такую пару, то сервис считается авторизованным.

Апробация

С октября 2020 года разработанный сервис используется на кафедре МО ЭВМ универстита “ЛЭТИ” в составе кафедральной прокторинг-системы.

- ~ 180 пользователей-студентов (весенний семестр 2021)
- 367 обработанных сессий (на момент 20 мая 2021)
- 5 пользователей-преподавателей и ассистентов кафедры
- Решение развернуто в составе прокторинг-системы на серверах кафедры МО ЭВМ университета «ЛЭТИ»
- Используется 3 типа сервисов обработки – обработка видео с вебкамеры, с экрана, а также, звука с микрофона (на момент 20 мая 2021)

Заключение

Основная проблема аналогов – ограниченность инструментария.

Анализ решения по критериям:

- По критерию универсальности и требований разработанный сервис превосходит все аналоги.
- По критерию требований к host-машине – превосходит 5 из 6 аналогов.
- По критерию срока обработки сессий разработанное решение проигрывает 4 из 6-ти аналогов в usability для преподавателей

Цель работы достигнута: Разработать сервис для формирования статистики нарушений во время экзамена.

Запасные слайды

Системные требования

- ОС, поддерживащая Docker и bash
- Процессор архитектуры X86, X86_64

Docker-compose

```
webcam_service:
  container_name: "ml_webcam_service"
  build:
    context: ./
    args:
      handler_name: webcam
  network_mode: "host"
  image: df/proctoring-ml-webcam:latest
  env_file:
    - ./webcam.env
```

```
2021-05-23 12:30:55,625 ERROR ml_service.client client.py:96 - Failed to get su
bmission from queue=webcam_queue, raw content: Queue 'webcam_queue' is empty, e
rror msg: Expecting value: line 1 column 1 (char 0)
2021-05-23 12:30:57,650 INFO ml_service.client client.py:104 - Got submission f
rom queue=webcam_queue
2021-05-23 12:30:57,651 INFO ml_service.client client.py:150 - Started parsing
submission
2021-05-23 12:30:57,652 INFO ml_service.client client.py:163 - Submission parse
d successfully, token=60aa4abca8955f6f825a09e3
2021-05-23 12:30:57,660 INFO ml_service.client client.py:133 - Sent status chec
k data {'token': 'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2MGFhNDViY2E4
OTU1ZjJhNzA1YTA5Y2MiLCJpYXQiOjE2MjE3NzE3MDh9.CjGlj6v_qKH6adCMvKWOZH0GvV1ns3Yy7k
sVAAFqUUY', 'timestamp': 1621773057.652764, 'event_type': 'session_processing_s
tart', 'data': '60aa4abca8955f6f825a09e3', 'attempt': 0}
2021-05-23 12:30:57,666 INFO ml_service.client client.py:73 - Auth response fro
m http://localhost:8080/ml/auth/token recieved: {'token': 'eyJhbGciOiJIUzI1NiIs
InR5cCI6IkpXVCJ9.eyJfaWQiOiI2MGFhNDViY2E4OTU1ZjJhNzA1YTA5Y2MiLCJpYXQiOjE2MjE3Nz
E3MDh9.CjGlj6v_qKH6adCMvKWOZH0GvV1ns3Yy7ksVAAFqUUY'}
2021-05-23 12:30:57,666 INFO ml_service.client client.py:170 - Started processi
ng submission 60aa4abca8955f6f825a09e3
2021-05-23 12:30:57,911 INFO handlers.webcam.webcam webcam.py:66 - Started proc
essing student webcam
2021-05-23 12:31:24,690 INFO handlers.webcam.webcam webcam.py:68 - Finished pro
cessing student webcam
2021-05-23 12:31:24,690 INFO ml_service.client client.py:174 - Finished process
ing submission 60aa4abca8955f6f825a09e3
2021-05-23 12:31:24,691 INFO ml_service.client client.py:175 - Elapsed time - 2
7.024080402989057 second
2021-05-23 12:31:24,736 INFO ml_service.client client.py:212 - Result for submi
ssion 60aa4abca8955f6f825a09e3 sent successfully
2021-05-23 12:31:24,744 INFO ml_service.client client.py:133 - Sent status chec
k data {'token': 'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2MGFhNDViY2E4
OTU1ZjJhNzA1YTA5Y2MiLCJpYXQiOjE2MjE3NzE3MDh9.CjGlj6v_qKH6adCMvKWOZH0GvV1ns3Yy7k
sVAAFqUUY', 'timestamp': 1621773084.736958, 'event_type': 'session_processing_e
nd', 'data': 'ok', 'attempt': 0}
2021-05-23 12:31:24,750 INFO ml_service.client client.py:73 - Auth response fro
m http://localhost:8080/ml/auth/token recieved: {'token': 'eyJhbGciOiJIUzI1NiIs
InR5cCI6IkpXVCJ9.eyJfaWQiOiI2MGFhNDViY2E4OTU1ZjJhNzA1YTA5Y2MiLCJpYXQiOjE2MjE3Nz
E3MDh9.CjGlj6v_qKH6adCMvKWOZH0GvV1ns3Yy7ksVAAFqUUY'}
2021-05-23 12:31:24,765 ERROR ml_service.client client.py:96 - Failed to get su
bmission from queue=webcam_queue, raw content: Queue 'webcam_queue' is empty, e
rror msg: Expecting value: line 1 column 1 (char 0)
```

Регистрация и авторизация сервиса

```
owlengineer@owlengineer-ThinkPad-E490:~/git$ docker logs 2d86cf3845e0
============================= test session starts ==============================
platform linux -- Python 3.6.12, pytest-6.2.4, py-1.10.0, pluggy-0.13.1
rootdir: /app/service
collected 2 items

tests/test_ext.py ..                                                     [100%]

============================== 2 passed in 0.39s ===============================
2021-05-23 10:24:25,494 INFO ml_service.client client.py:53 - Credentials succes
sfully acquired from http://localhost:8080/ml/auth/token
2021-05-23 10:24:25,565 INFO ml_service.client client.py:73 - Auth response from
 http://localhost:8080/ml/auth/token recieved: {'token': 'eyJhbGciOiJIUzI1NiIsIn
R5cCI6IkpXVCJ9.eyJfaWQiOiI2MGFhMmQ1OWE4OTU1ZjBiMGE1YTA5YjAiLCJpYXQiOjE2MjE3NjU0N
jV9.u4FnbYMdr_nSGdszHACNNo7rOFm3nJKbDfKg3nmwcbM'}
2021-05-23 10:24:26,396 INFO ml_service.client client.py:73 - Auth response from
 http://localhost:18040/xqueue/login/ recieved: {'content': 'Logged in', 'return
_code': 0}
2021-05-23 10:24:26,396 INFO ml_service.client client.py:241 - Auth successful,
got platform response={'token': 'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOi
I2MGFhMmQ1OWE4OTU1ZjBiMGE1YTA5YjAiLCJpYXQiOjE2MjE3NjU0NjV9.u4FnbYMdr_nSGdszHACNN
o7rOFm3nJKbDfKg3nmwcbM'}, xqueue response={'content': 'Logged in', 'return_code'
: 0}
```

Пример .env для конфигурации сервиса

```
XQUEUE_URL=http://localhost:18040
PLATFORM_URL=http://localhost:8000
ML_XQUEUE_AUTH=ml_service:ml_password
ML_XQUEUE_NAME=screencast
ML_ANALYZER=handlers.debug.debug.process
```

Модуль клиента

Структура python-модуля ml-service, который реализует pull-клиент функционал сервиса выглядит следующим образом:

client.py – скрипт, реализующий логику работы сервиса: запросы к внешним приложениям, центральный цикл

__main__.py – скрипт запуска сервиса

config.py – скрипт, содержащий основные настройки сервиса

utils.py – скрипт с вспомогательными функциями

Стэк основных технологий

- Python3 – язык программирования для функционала сервиса и модулей обработки.
- Xqueue – open-source решение, реализующее очередь задач на обработку.
- Pip – система управления python-пакетами. Необходим для установки требуемых библиотек как для общего функционала, так и для модулей обработки.
- HttPretty – open-source библиотека для реализации mock-тестирования функционала сервиса.
- PyTest – утилита для автоматизации тестирования.
- Docker – система автоматизации развертывания приложений в виртуальной машине.
- Docker-compose – инструмент для развертывания мульти-контейнерных систем.

Анализ аналогов, ч1

Название аналога	Идентификация студента по фото	Проверка наличия посторонних лиц	Отсутствие лица студента на изображении с камеры	Распознавание эмоций, мимики лица	Идентификация голоса студента	Проверка посторонних голосов
Proctortrack	+	+	+	-	-	-
ProctorExam	+	-	+	-	-	-
Examus	+	+	+	+	+	-
ProctorEdu	+	+	+	-	+	+
Proctoru	+	-	+	-	-	-
Proctorio	+	+	+	-	-	-

Анализ аналогов, ч2

Название аналога	Определение программного контекста на экране	Фиксация посторонних шумов	Распознавание направления взгляда	Клавиатурный почерк	Распознавание подсказок в разговорах
Proctortrack	+	-	-	-	-
ProctorExam	-	-	-	-	-
Examus	+	-	+	-	-
ProctorEdu	-	-	+	-	+
Proctoru	-	-	-	-	-
Proctorio	-	-	+	-	-

Схема обмена данными с разбиением очередей