

Рабочий лист №1

Дата "1" февраля 2025 г.  
(заполняется оргкомитетом)

Шифр ПИ-30  
(заполняется оргкомитетом)

Оценка работы

(таблица заполняется по итогам проверки работы членами жюри олимпиады)

№ задания	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Итого (итоговый балл, подпись председателя жюри)
Балл																90
№ задания	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	
Балл																ПЗ

Магистры Ум

(название олимпиады, заполняется участником)

Прикладная информатика

(профиль олимпиады, заполняется участником)

Задача 1. Проектирование архитектуры.

Система будет предназначена для автоматического выявления мошеннических телефонных разговоров на основе анализа аудиозаписей, текстовых расшифровок и т.д.

Предлагаю использовать микросервисную архитектуру, т.к. она обеспечивает гибкость, масштабируемость и простоту обслуживания.

Основные компоненты:

1) Модуль сбора и обработки информации (Data Investigation Service)

- Отвечает за получение данных из различных источников, их преобразование и отправку в систему обработки.

Источники данных: телекоммуникационная система (Call Center); база данных (данные о клиентах, истории звонков, контакты в личном списке); API сторонних сервисов (репутация звонков, гео-локация звонков); Загрузка файлов (запись телефонных разговоров).

Технологии: Kafka (очередь сообщений для асинхронной обработки данных), Python (реализация логики сбора и преобразования данных).

2) Модуль расшифровки речи (Speech-to-Text Service)

- Отвечает за преобразование аудиозаписей в текстовые расшифровки.

Сдано: 4 листа

Технологии: Google Cloud Speech-to-Text (распознавание речи); DeepSpeech (open-source библиотека для распознавания речи).

Конвертирует аудиозаписи из очереди сообщений; преобразует аудио в текст; отправляет обратно в очередь.

### 3) Модуль обработки текста (Text Processing Service)

- Отвечает за анализ текстовых расшифровок.

Функции: 1. Предварительная обработка текста (удаление стоп-слов, приведение к нижнему регистру).  
2. Анализ н-грамм слов (поиск слов и фраз, характерных для мошенничества).  
3. Анализ тональности (определение эмоций во время разговора).  
4. Извлечение сущностей (распознавание персональных данных, дат, мест и т.д.).

Технологии: Transformers (библиотека для обработки языков); Python (реализация логики); Pytorch (библиотека для обучения машинного обучения).

### 4) Модуль аналитики и выявления мошенничества (Fraud Detection Service).

- Отвечает за анализ результатов обработки текста и метаданных звонков для выявления потенциальных мошеннических разговоров.

Функции: 1. Фоновая система (наблюдение баллов за различные признаки мошенничества).  
2. Машинное обучение (классификация разговоров).

Технологии: Scikit-learn (библиотека для машинного обучения); Python (реализация алгоритмов), MySQL, SQLite (для хранения полученных данных).

Конвертирует результат анализа, анализирует данные, определяет вероятность мошенничества, результаты записываются в базу данных.

### 4) База данных (Database Service)

- Отвечает за хранение данных о звонках, результата анализа и модели машинного обучения.

Технологии: MySQL, SQLite.

### 5) Модуль визуализации и отчетности (Visualization Service)

- Отвечает за отображение результатов анализа и предоставление отчетности.

Дополнительный рабочий лист  
(без рабочего листа №1 недействителен)

Дата "1" февраля 2025 г.  
(заполняется участником)

Шифр ПИ-30  
(заполняется участником)

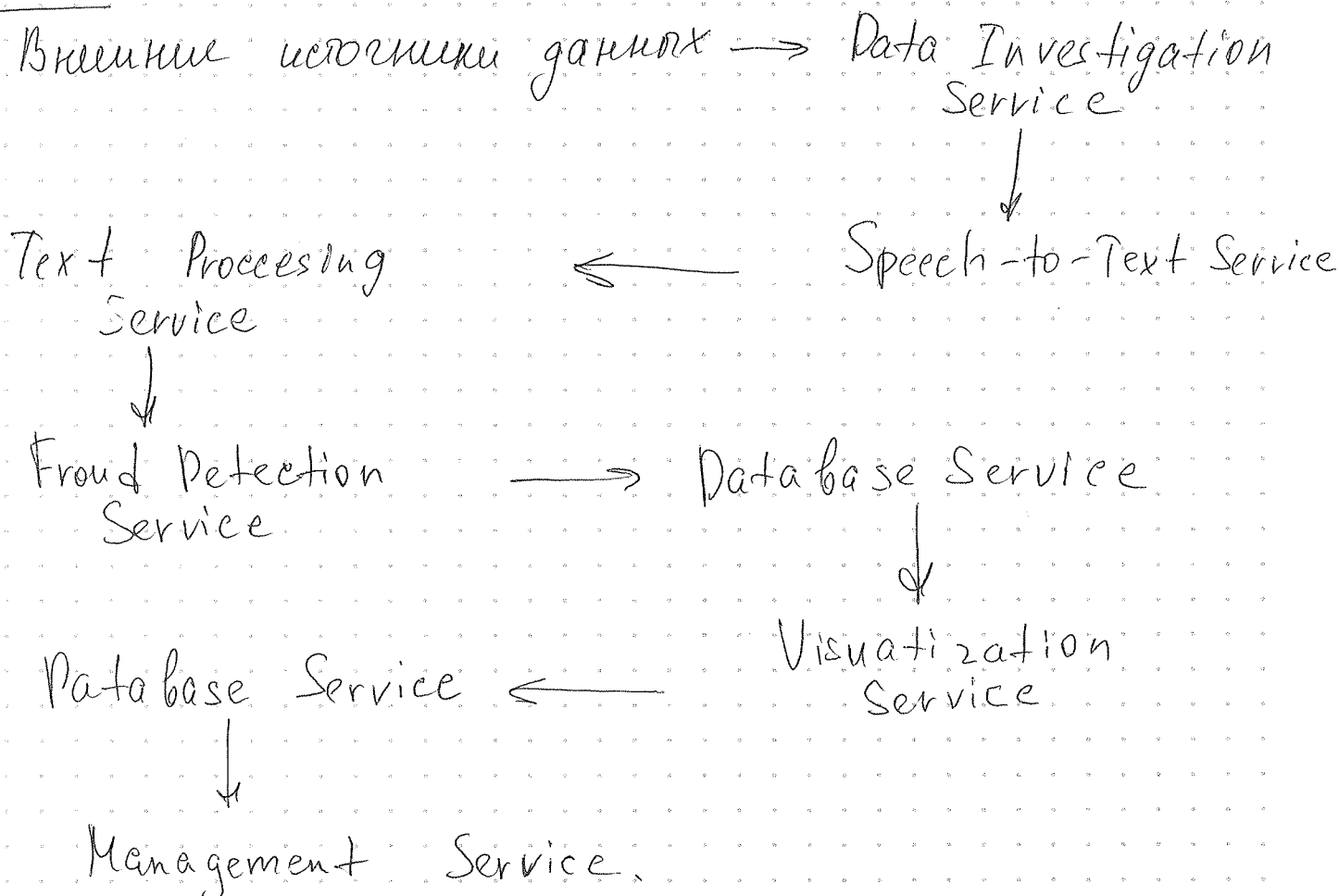
Функции: 1. Дашборд: Обращение ключевых показателей (кол-во звонков, процент мошенничества).  
2. Отчеты: отчеты по результатам анализа.  
3. Уведомления: оповещения о подозрительном разговоре.

Технологии: Grafana (визуализация); Python (REST API).  
6) Модуль управления (Management Service).

• Интерфейс для управления всей системой.

Технологии: Python (создание REST API и Web-интерфейса).

Схема:



## Задача 2. Алгоритм выявления применения мошенничества.

### Основные шаги:

#### 1) Предварительная обработка текста:

1. Разделение текста на предложения
2. Удаление стоп-слов ("и", "а", "но", "в")
3. Приведение к нижнему регистру
4. Приведение слов к базовой форме (купила, купишь, куплю → купи).

#### 2) Анализ лексики и тональности (проверка на наличие в тексте слов характерных для мошенничества)

Также создание словаря по категориям слов:

1. Финансовые: "деньги", "перевод", "карта", "код", "счёт", "пароль", "выигрыш",
2. Спешка: "срочно", "немедленно", "сейчас", "шато"
3. Угрозы / давление: "заблокируют", "откажут", "должно"
4. Обещание: "большой доход", "легкие деньги"
5. Специфические: "ФСБ", "сотрудник банка"

#### 3) Анализ структуры разговора

1. Соблюдение ролей
2. Запрос личной информации
3. Давление / срочность
4. Переход от одной темы к другой

#### 4) Фактная система

1. Наименее банков на основе анализа
2. Определение пороговых значений
3. Результат.

Дополнительный рабочий лист  
(без рабочего листа №1 недействителен)

Дата "1" февраля 20 25г.  
(заполняется участником)

Шифр ФИ-30  
(заполняется участником)

5) Обучные модели:

1. Сбор данных (данные разговоров с мошенниками)

2. Обучные модели

Псевдокод! Функция для предварительной обработки текста

```
def preprocess-text(text):
```

```
    text = text.lower()
```

```
    text = re.sub(r'[^\w\s]', '', text)
```

// нижний регистр  
// удалит все символы  
кроме букв и  
пробелов.

```
    tokens = word_tokenize(text)
```

// разбиваем на  
отдельные слова

```
    stop-words = set(stopwords.words('ru'))
```

// список  
stop-слов

```
    tokens = [token for token in tokens  
                if token not in stop-words]
```

// Удаляем  
stop-слова

```
    base-words = (здесь библиотека для приведения  
                  слова к нормальной форме)
```

```
    return ' '.join(base-words)
```

```
def analyze-lexicon(text):
```

Функция для анализа  
лексикона.

```
    process-text = preprocess-text(text)
```

// предварительная  
обработка

```
    keywords-count = Counter()
```

// подсчитываем  
вхождения  
ключевых слов.

```
    for category, keywords in FRAND_KEYWORDS.items():
```

```
        // считаем число вхождений для текущей  
        категории слов из словаря.
```

```
    return keywords-count
```

// возврат кол-ва  
вхождений

Функция для анализа тональности

```
def analyze-sentiment(text):
```

// анализирует тональность текста и  
помещает оценку по параметрам

return scores // возврат составной оценки

Функция для оценки подозрительности

```
def score-call(text)
```

keywords-count = analyze-lexicon(text)

sentiment = analyze-sentiment(text).

// на основе полученных данных вычисляет  
оценку подозрительности

Функция для предсказания мошенничества

```
def predict-fraud(text, model, vectorizer):
```

// Эта функция на основе всех полученных  
данных и при помощи машинного обучения  
делает предсказание яв-ся ли звонок мошен-  
ническим (0 или 1).

```
def load-model-and-vectorizer:
```

// Загружает обученную модель из файла  
и загружает векторизатор.

# словарь ключевых слов:

FRAUD-KEYWORDS = { "financial": ..., "urgency": ...,  
"threat": ..., "authority": ..., "vague-promise": ... }

Дополнительный рабочий лист  
(без рабочего листа №1 недействителен)

Дата "1" февраля 2025 г.  
(заполняется участником)

Шифр ПУ-30  
(заполняется участником)

```
Задача 3. import re
def extract_personal_data(text):
    "passport": [],
    "snils": [],
    "card-numbers": [],
    "cvv": [],
    "phone-numbers": [],
    "email": [],
    "name": []
    passport = re.compile(r'(\d{2} серию | паспорт | паспорт )
                        \s*(\d{4})\s*(\d{6})')
    re.compile(r'(\d{16}\d{22}\s*\d{22}\s*\d{6}\d{6})')
    re.compile(r'(\d{16}\d{4}\s*\d{6}\d{6})')
    snils = [ ... ]
    card-number = [ ... ]
    cvv = [ ... ]
    phone-numbers = [ ... ]
    email = [ ... ]
    name = [ ... ]
```

← аналогично

data["cvv"].extend(matches)

аналогично для групп типов данных

```
if __name__ == "__main__":
    text = " "
    Здравствуйе, меня зовут Иван. Мой паспорт:
    1234 567890. СНИЛС 111-222-333. карта:
    1234567890123456 и CVV 345'
```

номер. +7927 279 48 78 . Пота text@gmail.com. <sup>11111</sup>

```
extracted_data = extract_personal_data(text)
```

```
for key, value in extracted_data.items():
```

```
    if value:
```

```
        print(f"{key}: {value}").
```

Работа кода:

- 1) Инициализация слова data для хранения результатов
- 2) Для каждого типа данных определяется рекурсивное выражение
- 3) Поиск совпадений
- 3) Сохранение результатов (найденные совпадения добавляются в словарь data)
- 1) Создание текстовой строки
- 2) Вызов ф-ии extracted\_data.items()
- 3) Вывод результата в консоль.