# 

# **Кейс №1. Ассистент по документации компании**

**Цель**

Создать веб-сервис/телеграм бота, который позволяет пользователям находить точные ответы на свои вопросы, используя внутренние документы (PDF, DOCX, Markdown и т.д.), с помощью Retrieval-Augmented Generation (RAG).

**Проблема**

Внутренняя документация часто разрознена: инструкции, политики, техописания, договоры – хранятся в разных форматах и местах. Люди тратят много времени на ручной поиск информации или задают повторяющиеся вопросы другим сотрудникам.

**Решение**

Сервис, который позволяет:

* загружать и индексировать документы (PDF, DOCX, Markdown, TXT);
* искать по ним с помощью естественного языка;
* получать точные ответы с указанием источника (параграфа);
* работать через простой веб-интерфейс (или чат в телеграме).

**Примерная архитектура пайплайна**

Индексация:

* документы обрабатываются (парсинг);
* текст разбивается на чанки;
* чанки кодируются в векторное пространство с эмбеддинг-моделью (например, sentence-transformers);
* вектора сохраняются в БД.

Поиск и генерация:

* пользователь задаёт вопрос;
* система ищет релевантные чанки (top-k);
* LLM получает вопрос + найденные фрагменты, и генерирует ответ;
* возвращается ответ + ссылки на документы/чанки.

**Дополнительные идеи (если будет время)**

* авторизация и персонализация;
* автоматическое обновление базы при изменении файлов;
* поддержка мобильной версия (если веб-сервис);
* режим обучения: создание карточек/тестов на основе документов.

**Критерии успешности реализации решения:**

* модель правильно отвечает на 70% вопросов по документам;
* время ответа ≤ 5 секунд;
* возможность загрузить минимум 10 документов;
* источники указаны для каждого ответа.

# **Примерный стек**

Backend:

* Python (FastAPI или Flask);
* опционально: python-telegram-bot - для работы с Telegram;
* опционально: LangChain или LlamaIndex или что-то своё (для пайплайна RAG);
* Faiss / ChromaDB / pgvector (векторное хранилище)
* Open-source LLM

Frontend:

* Опционально: React / Next.js / Streamlit / Vue

# **Доступные бесплатные модели:**

API Yandex Foundation Models (<https://yandex.cloud/ru/services/foundation-models>) – выдаётся 3000 рублей на использование моделей Яндекса при первой привязке карты.

GigaChat (<https://developers.sber.ru/portal/products/gigachat-api>) – доступно 1кк токенов, потребуется аккаунт Сбера.

Together AI (<https://www.together.ai>) – выдаётся 1 доллар при регистрации, хватает на 1кк токенов для входа и выхода + доступны бесплатные модели.

**Сервисы с GPU:**

Google Colab – потребуется учетная запись в гугле.

Kaggle – доступно 30 часов работы GPU, потребуется привязка телефона.

# 

# **Кейс №2. Персональный помощник по обучению языкам**

**Цель**

Создать веб-сервис\телеграм бота для изучения иностранного языка, позволяющего:

* общаться с ИИ на выбранном языке;
* задавать вопросы по своим учебным материалам;
* получать голосовые ответы и задавать голосом вопросы (дополнительно);
* получать ежедневные напоминания об обучении.

**Проблема**

Большинство языковых ботов основаны на статических правилах или скриптах. Они не гибкие и не могут адаптироваться под конкретные материалы, которыми пользуется обучаемый (например, учебник, курс, PDF, заметки и т.д.).

**Решение**

Сервис, который позволяет:

* общаться с пользователем на изучаемом языке в текстовом или голосовом виде (дополнительно);
* отправлять напоминания, чтобы формировать у обучаемого привычку к регулярному обучению;
* давать ответы и объяснения на основе загруженных материалов.

**Примерная архитектура пайплайна**

Интерфейс:

* принимает текстовые и голосовые (дополнительно) сообщения от пользователя;
* поддерживает команды: /upload, /ask, /remind, /practice, /help.

Обработка пользовательских файлов:

* при загрузке PDF, DOCX или TXT документа бот извлекает текст;
* текст разбивается на логические фрагменты (чанки);
* для каждого чанка создаются векторные представления с помощью embedding-модели;
* векторы сохраняются во внутреннюю базу.

RAG (Retrieval-Augmented Generation):

* когда пользователь задаёт вопрос, сервис ищет релевантные фрагменты текста из загруженных материалов;
* выбранные фрагменты и сам вопрос передаются в языковую модель (LLM), которая формирует ответ.

Языковая модель (LLM):

* используется open-source модель;
* модель работает в режиме генерации ответа по запросу и найденному контексту;
* ответ формируется на изучаемом языке (например, английском или испанском).

Напоминания:

* пользователь может установить расписание напоминаний (например, “/remind 10:00”).
* планировщик (APScheduler или аналог) отслеживает заданное время и отправляет пользователю нотификацию с напоминанием;
* сервис может автоматически предлагать короткое задание или повторение слов в это время.

**Дополнительные идеи (если будет время):**

* ведение личного словаря и трекинг прогресса;
* Text-to-Speech (TTS) - пользователь получает не только текст, но и озвучку ответа - это развивает восприятие речи на слух и произношение;
* Speech-to-Text (STT) - когда пользователь отправляет голосовое сообщение, бот распознаёт его содержимое.

**Критерии успешности реализации решения:**

* рабочий RAG - сервис отвечает на вопросы на основе загруженного PDF/текста, а не "из головы";
* время ответа ≤ 10 секунд;
* упражнения - сервис умеет генерировать хотя бы один тип задания (тест, перевод и т.п.);
* напоминания - пользователь может настроить ежедневное напоминание об обучении;
* поддержка речи - сервис принимает голосовые сообщения и отвечает голосом (дополнительно).

# **Примерный стек**

Backend:

* Python (FastAPI или Flask);
* опционально: python-telegram-bot - для работы с Telegram;
* опционально: LangChain или LlamaIndex или что-то своё (для пайплайна RAG);
* Faiss / ChromaDB / pgvector (векторное хранилище)
* Open-source LLM

Frontend:

* Опционально: React / Next.js / Streamlit / Vue

# **Доступные бесплатные модели:**

API Yandex Foundation Models (<https://yandex.cloud/ru/services/foundation-models>) – выдаётся 3000 рублей на использование моделей Яндекса при первой привязке карты.

GigaChat (<https://developers.sber.ru/portal/products/gigachat-api>) – доступно 1кк токенов, потребуется аккаунт Сбера.

Together AI (<https://www.together.ai>) – выдаётся 1 доллар при регистрации, хватает на 1кк токенов для входа и выхода + доступны бесплатные модели.

**Сервисы с GPU:**

Google Colab – потребуется учетная запись в гугле.

Kaggle – доступно 30 часов работы GPU, потребуется привязка телефона.

# 

# **Кейс №3. Персональный помощник по питанию**

**Цель**

Создать веб-сервис\телеграм бота, который помогает пользователю:

* планировать питание в соответствии с его целью (похудение, набор массы, ЗОЖ);
* объяснять нутриенты, БЖУ, пищевую ценность продуктов;
* отвечать на вопросы по диетам, режиму питания и рекомендациям;
* использовать в своей работе достоверные источники (через RAG и LLM).

**Проблема**

* пользователи часто теряются в диетах и не понимают, что им подходит;
* большинство советов в интернете - обобщённые, без учёта контекста. Потребуется сервис, который адаптирует информацию под конкретного человека и объясняет просто.

**Решение**

Сервис, который позволяет:

* отвечать на вопросы по питанию, продуктам, диетам, дефициту/профициту калорий;
* строить примерное меню или рацион на день/неделю;
* основываться на RAG по достоверным источникам (Harvard Health, NHS, ВОЗ, PubMed и др.).
* учитывать персональные параметры: пол, возраст, активность, цель.

**Примерная архитектура пайплайна**

Интерфейс:

* принимает текстовые и голосовые (дополнительно) сообщения от пользователя.

RAG-механизм:

* база знаний: статьи по питанию, диетологии, FAQ;
* Embedding + индексация (ChromaDB, Faiss);
* поиск фрагментов + генерация ответа через LLM (Mistral, LLaMA и др.)

Генерация рациона:

* LLM составляет меню на день/неделю по цели;
* можно задать ограничения: без лактозы, без глютена, вегетарианское и т.д.;
* используется формула Харриса-Бенедикта;
* используются параметры: возраст, пол, вес, рост, цель

**Критерии успешности реализации решения:**

* RAG отвечает по достоверной базе знаний;
* время ответа ≤ 5 секунд;
* бот учитывает параметры пользователя (возраст, пол, цель и т.д.);
* умеет генерировать рацион на день или неделю;
* отвечает простым и понятным языком.

# **Примерный стек**

Backend:

* Python (FastAPI или Flask);
* опционально: python-telegram-bot - для работы с Telegram;
* опционально: LangChain или LlamaIndex или что-то своё (для пайплайна RAG);
* Faiss / ChromaDB / pgvector (векторное хранилище)
* Open-source LLM

Frontend:

* Опционально: React / Next.js / Streamlit / Vue

# **Доступные бесплатные модели:**

API Yandex Foundation Models (<https://yandex.cloud/ru/services/foundation-models>) – выдаётся 3000 рублей на использование моделей Яндекса при первой привязке карты.

GigaChat (<https://developers.sber.ru/portal/products/gigachat-api>) – доступно 1кк токенов, потребуется аккаунт Сбера.

Together AI (<https://www.together.ai>) – выдаётся 1 доллар при регистрации, хватает на 1кк токенов для входа и выхода + доступны бесплатные модели.

**Сервисы с GPU:**

Google Colab – потребуется учетная запись в гугле.

Kaggle – доступно 30 часов работы GPU, потребуется привязка телефона.



**Кейс №4. Бот для VK Мессенджера как дополнительный канал поддержки и взаимодействия**

**Контекст**

Компания внедрила Клиентский сервис (КС Деск) для обработки обращений пользователей через email,сайте и телефон. Однако современные пользователи все чаще предпочитают мессенджеры как основной канал коммуникации.

**Проблема**

Отсутствует интеграция с популярным мессенджером VK.  
Пользователи вынуждены переключаться на другие каналы, что снижает удобство и скорость обслуживания. Ручная обработка обращений в мессенджерах неэффективна.

**Решение**

Бот для VK Мессенджера, который станет дополнительным каналом для взаимодействия с КС Деск. Упростит пользователям процесс обращения в поддержку. В перспективе сможет автоматизировать часть рутинных запросов (FAQ, статус заявки и др.).

**Функциональные требования к боту**

1. Авторизация и верификация пользователя. Привязка аккаунта VK к учетной записи в системе компании (по email/номеру телефона).
2. Основные сценарии взаимодействия:

* Создание обращения через бота (ввод темы, описания, отправка в КС Деск).
* Проверка статуса заявки по номеру или через привязанный аккаунт.
* Автоматические ответы на частые вопросы.

1. Интеграция с КС Деск:

* Корректная синхронизация заявок между ботом и системой поддержки (формат JSON/XML).
* Поддержка смены статуса заявки.

1. Пользовательский опыт (UX):

* Понятные подсказки и кнопки для навигации.
* Минимальное количество шагов для создания заявки.
* Учет истории обращений для авторизованных пользователей.

1. Аналитика и данные:

* Сохранение полной истории диалогов с метками времени.

**Технические требования**

Backend: Python (FastAPI или Django).

База данных: PostgreSQL.

Библиотека для работы с VK API: vkbotlе.

Деплой: возможность разворачивания на облачных платформах (Docker).

**Формат представления данных**

Демонстрация работы бота со ссылкой на рабочего бота в VK.

Примеры диалогов: успешное создание заявки, обработка FAQ, передача оператору.

**Дополнительные рекомендации**

Уделите внимание безопасности: данные пользователей (email, номера заявок) должны быть защищены.

Продумайте сценарии для тестирования: например, как бот реагирует на спам или нестандартные запросы.



**Кейс №5. Мобильное приложение для распознавания текста и интеграции с Электронным архивом**

**Контекст**

Организации активно переводят документы в цифровой формат для долгосрочного хранения и быстрого поиска. Однако ручной ввод метаданных (реквизитов) документов замедляет процесс, повышает риск ошибок и усложняет систематизацию. Современные решения требуют автоматизации: извлечения ключевых данных из документов, их категоризации и передачи в архивные системы.

**Проблема**

1. Ручной ввод реквизитов документов (название, дата, номер, автор и т.д.) занимает много времени.
2. Ошибки в метаданных приводят к проблемам при поиске документов в архиве.
3. Отсутствие автоматической генерации хештегов/тегов усложняет фильтрацию документов по категориям.

**Решение**

Мобильное приложение, которое:

1. Автоматически распознает текст с фотографий документов.
2. Извлекает ключевые реквизиты для формирования метаданных и хештегов.
3. Интегрируется с системой Электронного архива через API.
4. Упрощает процесс передачи документов, минимизируя ручные действия пользователя.

**Функциональные требования**

1. Сценарий использования:

Пользователь фотографирует документ (например, приказ, договор, справку).

Приложение обрабатывает изображение, распознает текст.

Извлекает реквизиты:

* тип документа (например, "Приказ", "Договор");
* дата создания;
* номер документа;
* автор/организация;
* краткое содержание (опционально).

Генерирует хештеги на основе реквизитов (например, #Приказ\_2024\_05\_15).

Позволяет пользователю редактировать извлеченные данные.

Отправляет данные в Электронный архив через API.

1. Дополнительные функции

Возможность перефотографировать документ при низком качестве распознавания.

История загруженных документов в приложении.

Уведомления об ошибках (например, если не удалось распознать дату).

Поиск по документам в локальной истории.

**Технические требования**

1. Распознавание текста (OCR):

Использование библиотек/API:

* Tesseract, Google ML Kit, Azure Cognitive Services или аналоги.
* Поддержка обработки изображений с коррекцией перспективы и шумов.

1. Интеграция с архивом:

* REST API для передачи данных (формат JSON).

1. Мобильная разработка:

* Платформы: Android и iOS (предпочтительно кроссплатформенное решение: Flutter/React Native).
* Локальное хранение данных: SQLite или аналоги.

1. Безопасность:

* Шифрование передаваемых данных (HTTPS).
* Аутентификация пользователя через токен.

**Формат данных**

1. Входные данные:

* Изображение документа (JPEG/PNG).
* Поддерживаемые разрешения: минимум 1920x1080 пикселей.

1. Выходные данные:

* JSON-объект с реквизитами.

**Пользовательский опыт (UX).**

Понятные подсказки и кнопки для навигации.



**Кейс №6. Веб-приложение для геймификации продуктов компании Кейсистемс**

**Цель**

Создать веб-приложение с элементами геймификации, интегрируемое на сайт компании «Кейсистемс», которое повысит вовлеченность посетителей и будет стимулировать их интерес к продуктам, способствуя привлечению новых клиентов.

**Задача**

Создать прототип интерактивного веб-приложения с игровыми механиками, органично вписывающийся в пользовательский опыт сайта компании «Кейсистемс» и мотивирующий посетителей к активному взаимодействию с продуктами компании.

**Требования**

1. Использование веб-технологий: HTML, CSS, JavaScript и современных фреймворков (React, Vue, Angular и др.) для создания интерактивного, адаптивного и удобного интерфейса, готового к размещению на сайте компании «Кейсистемс»: https://www.keysystems.ru.
2. Игровые механики: внедрить элементы геймификации. Примеры:

* система очков и уровней за выполнение действий (просмотр продуктов, регистрация, участие в опросах);
* прогресс-бары, показывающие продвижение пользователя;
* коллекционирование виртуальных наград и бейджей;
* интерактивные квесты или викторины, связанные с продуктами компании;
* лидерборды для создания соревновательного элемента.

1. Мотивация пользователей: предусмотреть систему наград, которая может включать виртуальные бонусы и реальные стимулы (скидки, пробные версии, консультации).
2. Интеграция с сайтом: решение должно легко встраиваться в существующую структуру сайта компании «Кейсистемс», быть легковесным и не ухудшать производительность.
3. UX/UI: интерфейс должен быть интуитивно понятным, привлекательным и не перегруженным, чтобы геймификация усиливала, а не отвлекала от основного контента.
4. Аналитика: предложить способы сбора данных о вовлеченности пользователей и эффективности геймификации (например, через Google Analytics, Яндекс Метрика или собственные метрики).

**Ожидаемый результат**

1. Рабочий прототип веб-приложения.
2. Описание сценария взаимодействия пользователя с геймифицированным элементом, включая мотивационные стимулы и систему наград.
3. Набор метрик для оценки эффективности геймификации, включающий показатели вовлеченности (время на сайте, глубина просмотра), конверсии (регистрации, заявки).

**Типовой сценарий использования**

1. Первичное вовлечение: пользователь (потенциальный клиент) заходит на сайт компании «Кейсистемс» и видит на главной странице интерактивный элемент – например, квест или викторину, посвящённую продуктам компании. Например, это может быть «Путешествие по продуктам Кейсистемс» с виртуальной картой, где нужно собрать коллекцию знаний или выполнить задания, связанные с функционалом продуктов.
2. Выполнение игровых заданий: пользователь выполняет серию небольших заданий – отвечает на вопросы, решает задания мини-игры, открывает «пасхалки» на страницах сайта, набирает очки и получает виртуальные награды (бейджи, уровни).
3. Мотивация и награды: за активное участие пользователь получает бонусы – скидки на услуги, пробные периоды, консультации или возможность участвовать в розыгрыше призов. Это стимулирует его глубже изучать продукты и повышает вероятность обращения в компанию.
4. Социальное вовлечение: пользователь может делиться своими достижениями в соцсетях или приглашать коллег для совместного участия, что расширяет охват и привлекает новых клиентов.
5. Аналитика и обратная связь: система собирает данные о вовлеченности, активности и конверсии пользователей, позволяя компании оптимизировать геймификацию и маркетинговые стратегии.

**Критерии оценки**

1. Оригинальность и соответствие бизнес-целям компании «Кейсистемс».
2. Качество и удобство веб-интерфейса.
3. Глубина и продуманность игровых механик.
4. Техническая реализуемость и масштабируемость решения.
5. Потенциал повышения вовлеченности и привлечения новых клиентов.
6. Наличие предложений по аналитике и оценке результатов.



**Кейс №7. Веб-приложения для загрузки, трансформации данных с их сохранением в хранилище данных**

**Цель**

Создание или настройка веб-приложения с графическим интерфейсом, позволяющим решать вопросы загрузки, трансформации данных с последующим их сохранением в хранилище данных.

**Задача**

Разработать или использовать любой из существующих ETL инструментов с открытым исходным кодом. Настроить загрузку данных по https из публичного ресурса, провести трансформацию и загрузку информации в хранилище.

**Требования**

1. Использование веб-технологий: HTML, CSS, JavaScript и современных фреймворков (React, Vue, Angular и др.);
2. Хранилище данных - СУБД PostgreSql.
3. Процесс загрузки, преобразования и сохранения данных должен логгироваться.
4. Провести локализацию продукта – решение должно быть с интерфейсом на русском языке.
5. Важен дружелюбный интерфейс.
6. Предусмотреть возможность планирования задач. Задачи создаются и
7. запускаются с определенной периодичностью. Возможность просматривать текущие активные задачи и их статус.
8. Желательно, если проект с открытым исходным кодом, то собрать самостоятельно дистрибутив продукта.
9. Трансформация (преобразование) данных допускается как настраиваемся через интерфейс, там и программная – со ставками управляющего кода.

**Ожидаемый результат**

Рабочий прототип веб-приложения.

**Пример исходных данных**

Исходные данные - публичный ресурс <https://www.rosstat.gov.ru/>.

**Ссылки на скачивание данных**

<https://rosstat.gov.ru/storage/mediabank/02-24-12.xlsx>

<https://rosstat.gov.ru/storage/mediabank/04-23-12.xlsx> <https://rosstat.gov.ru/storage/mediabank/04-22-12.xlsx>

<https://rosstat.gov.ru/storage/mediabank/04-21-12.xlsx>

Данные по «Средняя заработная плата отдельных категорий работников социальной сферы и науки в организациях» за 4 года. Как пример, загрузить исходные данные выделив только «Приволжский федеральный округ» и сохранить в хранилище данные. При трансформации данных можно посчитать среднюю заработную плату по округу и по РФ.

Настроить задачу по загрузке данных. Дополнительно, можно взять любой сервис (либо его создать), который возвращает большой объем данных в виде json или xml файла и обработать его. Желательно, чтобы графически был виден статус задачи и результат.

**Критерии оценки**

1. Оригинальность и соответствие бизнес-целям компании «Кейсистемс».
2. Качество и удобство веб-интерфейса.
3. Сложность трансформации и создания скриптов обработки данных.



**Кейс №8. Голосовой ассистент бухгалтера**

**Контекст**

В организации внедрены:

1. Система бухгалтерского учета 1С: БГУ 2.0.
2. Система бюджетной отчетности Свод-СМАРТ («Кейсистемс»)

**Задача**

Создать ассистента, который по запросу голосом от пользователя смог бы сформировать API вызов в адрес данных систем и ответить на него голосом.

**Требования**

Информацию по используемым справочникам можно взять из: Бюджетная классификация на 2025 (https://www.keysystems.ru/products/npi/npi-2025.php) по ссылке Скачать файл, Приложении 1, 2)

**По блоку «Планирование и бюджет» (справочник видов расходов)**:

Примеры фраз запросов:

* «Ассистент, скажи остатки по КВР 111»
* «Ассистент, проверь лимиты на закупки по 244»

**По блоку «Учет и оплата» (справочник КОСГУ):**

Примеры фраз запросов:

* «Проверь статус платежа по документу 789»
* «Какой код КОСГУ используется на канцелярию?»

**По блоку «Отчетность» (справочник форм отчетности см. Приложение – в основном в речи используются последние 3 цифры и буквы после):**

* «Какой срок сдачи отчетности по форме 737?»
* «Какой статус у отчета по форме 725?»

Примеры работы по блоку «Планирование и бюджет» (скачайте справочник видов расходов):

Интент: **get\_budget\_balance**

Описание: Получение остатков по КВР

Примеры фраз:

* "Какие у нас остатки по КВР 111?"

Примеры API вызовов в адрес 1С:БГУ:

POST \_endpoint1

{

"command": "get\_budget\_balance",

"parameters": {

"kvr": "111"

}

}

Результат API (Озвучивание):

На 111 КВР на сегодня остатки составляют 1 миллион рублей.

Интент: **get\_check\_limit\_by\_kvr**

Описание: Уточнение доведенных лимитов по КВР

Примеры фраз:

«Посмотри лимиты по КВР 340»

POST \_endpoint1

{

"command": "get\_check\_limit\_by\_kvr",

"parameters": {

"kvr": "340"

}

}

Результат API (Озвучивание):

Доведенные лимиты по 340 КВР составляют 10 миллионов рублей.

Примеры работы по блоку «Учет и оплата» (см.Приложение 1):

Примеры фраз:

«Проверь статус платежа по документу 789»

«Какой код КОСГУ используется на канцелярию?»

Интент: **get\_check\_document\_status**

Описание: Проверка статуса документа

Примеры фраз:

"Какой статус документа 789?"

Примеры API вызовов в адрес 1С:БГУ:

POST \_endpoint1

{

"command": "get\_check\_document\_status",

"parameters": {

"document\_id": "789"

}

Озвучивание:

Документ находится на этапе «Согласование главным бухгалтером»

Интент: **get\_suggest\_kosgu**

Описание: Подбор КОСГУ по описанию

Примеры фраз:

"Какой код КОСГУ для канцелярских товаров?"

"Подбери КОСГУ на ремонт принтера"

POST \_endpoint1

{

"command": "get\_suggest\_kosgu",

"parameters": {

"description": "канцелярские товары"

}

}

Результат API (Озвучивание):

КОСГУ на канцелярские товары нужно отражать по Виду расходов подстатье КОСГУ 346 или 310, если срок использования превышает 1 год.

Примеры работы по блоку «Отчетность» (см.Приложение 2 «Справочник форм отчетности» – в основном в речи используются последние 3 цифры и буквы после)

Примеры API вызовов в адрес «Свод-СМАРТ»:

Интент: **get\_check\_report\_status**

Описание: Проверка факта сдачи отчёта

Пример фраз:«Какой статус у отчета по форме 725?»

POST \_endpoint2

{

"command": "get\_last\_report\_status",

"parameters": {

"document\_id": "725"

}

}

Результат API (Озвучивание):

Отчет 725 находится в статусе «Утвержден»

Интент: **get\_reporting\_deadline**

Описание:Получение сроков сдачи отчёта

Пример фраз:«До какого числа нужно сдать форму 737?»

POST \_endpoint2

{

"command": "get\_last\_report\_deadline",

"parameters": {

"report\_type": "737"

}

}

Результат API (Озвучивание):

Отчет 737 необходимо предоставить до 15 числа текущего месяца.

Интент: **get\_reporting\_instructions**

Описание:Получение инструкции по заполнению отчетности

Пример фраз:«Как заполнить форму 737?»

POST \_endpoint2

{

"command": "get\_reporting\_instructions",

"parameters": {

"report\_type": "737"

}

}

Результат API (Озвучивание):

Инструкция по заполнению отчетности расположена в разделе Навигатор\Контрольные соотношения и форматы загрузки.

Приложение 1 (КОСГУ фрагмент)

300|Поступление нефинансовых активов

310|Увеличение стоимости основных средств

320|Увеличение стоимости нематериальных активов

330|Увеличение стоимости непроизведенных активов

340|Увеличение стоимости материальных запасов

341|Увеличение стоимости лекарственных препаратов и материалов, применяемых в медицинских целях

342|Увеличение стоимости продуктов питания

343|Увеличение стоимости горюче-смазочных материалов

344|Увеличение стоимости строительных материалов

345|Увеличение стоимости мягкого инвентаря

346|Увеличение стоимости прочих материальных запасов

347|Увеличение стоимости материальных запасов для целей капитальных вложений

349|Увеличение стоимости прочих материальных запасов однократного применения

350|Увеличение стоимости права пользования

351|Увеличение стоимости права пользования активом

352|Увеличение стоимости неисключительных прав на результаты интеллектуальной деятельности с определенным сроком полезного использования

353|Увеличение стоимости неисключительных прав на результаты интеллектуальной деятельности с неопределенным сроком полезного использования

360|Увеличение стоимости биологических активов

Приложение 2 (Справочник форм отчетности):

0503730|Баланс государственного (муниципального) учреждения (ОКУД 0503730)

0503725|Справка по консолидируемым расчетам учреждения (ОКУД 0503725)

0503710|Справка по заключению учреждением счетов бухгалтерского учета отчетного финансового года (ОКУД 0503710)

0503737|Отчет об исполнении учреждением плана его финансово-хозяйственной деятельности (ОКУД 0503737)

0503738|Отчет об обязательствах учреждения (ОКУД 0503738)

0503738-НП|Отчет об обязательствах учреждения по национальным проектам (ф. 0503738-НП)

0503721|Отчет о финансовых результатах деятельности учреждения (ОКУД 0503721)

0503723|Отчет о движении денежных средств учреждения (ОКУД 0503723)

0503760|Пояснительная записка к Балансу учреждения (ОКУД 0503760)

0503790|Сведения о вложениях в объекты недвижимого имущества, об объектах незавершенного строительства бюджетного (автономного) учреждения (ОКУД 0503790)

0503830|Разделительный (ликвидационный) баланс государственного (муниципального) учреждения (ОКУД 0503830)



**Кейс №9. Создание внешней обработки или расширения с функцией выгрузки в ручном режиме данных по документам бухгалтерии и представление их в виде отчета**

**Цель**

Создание внешней обработки или расширения с функцией выгрузки в ручном режиме данных по документам бухгалтерии и представление их в виде отчета.

**Задача**

Разработать информационную систему с подсистемами (документы, справочники, администрирование). Подсистема документы должна содержать в себе следующие документы: «Приходная накладная», «Расходная накладная».

**Требования**

1. Обеспечить данные для выгрузки: создать несколько документов каждого типа, наполнить данными и выполнить операции проведения.
2. Интерфейс обработки должен содержать в себе таблицу документов с возможностями выбора нескольких документов для выгрузки, просмотром статуса выгрузки, других характеристик документов.
3. Таблица документов должна состоять из следующих реквизитов: тип документа, наименование документа, номер документа, дата документа, сумма документа.
4. Настройки выгрузки: должны предоставлять возможность задать:

* путь к папке (куда необходимо выгрузить данные);
* формат выгрузки данных (XML);
* период документов для выгрузки.

1. В выгрузку должны попадать только проведенные и не помеченные на удаление документы.
2. Выгрузка должна формировать файлы в разрезе суток.



**Кейс №10. Разработка мобильного приложения для учета посетителей оффлайн-события**

**Контекст**

Маркетинговая служба компании является постоянным организатором крупномасштабных мероприятий для клиентов и партнеров и сталкивается с проблемами:

* неэффективного учета гостей: бумажные списки, очереди на входе и т.п.;
* потери данных: нет аналитики по отраслям деятельности, географии участников.
* фродом: «лишние» люди в зале

**Цель**

Мобильное приложение для учета посетителей оффлайн-события, обеспечивающего учет входа и выхода, проверку регистрации посетителей на событие и ведение аналитики

**Функционал**

1. Для пользователей приложения:

* регистрация в приложении;
* просмотр списка событий;
* регистрация на событие;
* сканирование qr-кода события (открывается форма регистрации на событие в приложении);
* профиль в приложении с qr-кодом пользователя.

1. Для организаторов событий:

* создание, редактирование, удаление события (поля: наименование, место проведения, даты начала и окончания, описание события);
* просмотр участников события, одобрение/отклонение заявок на участие;
* сканирование qr-кодов пользователей приложения в меню события с отображением пользователя, код которого сканируется (ФИО, фото);
* вывод информации о регистрации пользователя на событие.

1. Аналитика регистраций:

* количество регистраций по дням.

1. Аналитика посещений:

* количество входов;
* количество выходов;
* среднее время посещения;
* текущее количество посетителей в виде онлайн-графика с отображением за последние три часа;
* журнал посещений, отображение списка сканированных пользователей на событие, с фильтрацией по входу/выходу;
* по выбранному участнику события, сортировки по дате;
* экспорт аналитики и списка участников csv.

**Требования**

1. Реализация на pwa, либо на любом кроссплатформенном фреймворке (react, flutter, xamarin)
2. Одно приложение для организатора и для пользователя, организатор может регистрироваться как участник в событиях других пользователей.
3. Уведомления: почта, опционально пуши.
4. Обязательные требования: учет посетителей, аналитика
5. REST API
6. OAUTH (vk, google)