МЕДСИ

Описание функциональных характеристик
системы контроля качества **«КОМПАС»**

СОДЕРЖАНИЕ

1. **Общие сведения**
   - **Полное название ПО**: Компас.
   - **Класс ПО:** 04.13 Программное обеспечение для решения специфических задач медицинских организаций.
2. **Назначение программы**

Программа предназначена для повышения эффективности и качества предоставляемых медицинских услуг за счет автоматизации процессов анализа и оценки медицинской документации (на основании приказа N 785н от 31 июля 2020 г Министерства Здравоохранения Российской Федерации). Программой могут пользоваться врачи, заведующие отделениями, главные врачи, эксперты по качеству.

3. **Функциональные возможности**

3.1. **Модуль обработки входящих документов**

Обеспечивает интеграцию с внешними медицинскими информационными системами (МИС) для получения структурированных данных (протоколов приемов, заключений диагностических исследований).

3.2. **Модуль автоматизированного контроля качества полученной документации**

Модуль осуществляет автоматизированную валидацию полученных документов на соответствие предустановленным медико-технологическим критериям и алгоритмам, с последующей визуализацией результатов проверки в интерфейсе пользователя. Модуль обеспечивает процесс проверки, который включает:

3.2.1. Определение типа входящего документа и расчет его параметров по настроенным правилам;

3.2.2. Планирование проверки по настроенным для типа документа правилам (с учетом выполнения/не выполнения условий запуска);

3.2.3. Запуск и проведение запланированной проверки по описанию правила;

3.2.4. Формирование результатов и вывод их на экран пользователей;

3.2.5. Переход на запись в МИС.

3.3. **Модуль перезапуска проверок**

Реализует логику ручного перезапуска проверок. Реализованы возможности перезапуска одной проверки, или ряда проверок по случаю.

3.4. **Модуль агрегации клинических случаев и управления картами контроля**

Реализует логику объединения разрозненных медицинских записей в единый «Случай медицинского обслуживания» по завершении этапа лечения. Модуль обеспечивает:

3.4.1. Автоматическую генерацию первичных карт контроля (0-го уровня);

3.4.2. Формирование печатных форм карт контроля установленного образца с возможностью их последующей печати;

3.4.3. Поддержку многоуровневой системы экспертизы: создание, расчет оценки, утверждение и иерархическое копирование оценок между экспертными уровнями с возможностью заполнения комментариев.

3.5. **Модуль управления правилами и алгоритмами (Rule Engine)**

Представляет собой среду настройки бизнес-логики обработки медицинских документов. Позволяет описывать типы входящих документов, определять наборы параметров и формировать сложные цепочки правил для автоматической проверки качества оказания медицинской помощи. Модуль обеспечивает возможности

3.5.1. Создания и редактирования параметров разных типов (простые, табличные, структурированные(json), параметры-массивы, рассчитываемые на лету параметры). Механизм расчета параметров предполагает работу с текстом входящего протокола по логике регулярных выражений, библиотеки yargy.

3.5.2. Создания и редактирования правил, которые могут быть

3.5.2.1. Правилами наличия документа

3.5.2.2. Правилами качества документа

3.5.2.3. Правила могут учитывать всю цепочку документов пациента из его карты за указанный период, правила могут настраиваться на любой описанный параметр документа из карты пациента с условиями сравнения с любыми ручными значениями или другими параметрами с операторами больше/меньше/равно/содержит/не содержит и тд. У правил могут быть условия на их запуск.

3.5.2.4. Правила могут быть «внешними» - то есть забирать информацию по результатам проверок из стороннего сервиса (разрешенного на уровне доступа из сети). Например, для настройки обращений к внутреннему сервису ИИ.

3.6. **Модуль организационно-штатной структуры**

Реализует получение и хранение информации об учреждениях, отделениях, сотрудниках и их штатных единиц из системы МИС.

3.7. **Модуль аналитической отчетности**

Реализует механизмы статистической обработки данных и генерации регламентированной управленческой отчетности. Обеспечивает формирование ежемесячных и квартальных отчетов по качеству документации и результативности экспертиз для руководства медицинской организации в

различных разрезах (по подразделениям, сотрудникам, типам ошибок). Реализован на базе интеграции с аналитической платформой с открытым исходным кодом (Apache Superset).

3.8. **Модуль контроля назначений и проведения консилиумов**

Автоматизирует процесс проверки необходимости проведения врачебных консилиумов. Модуль осуществляет мониторинг триггерных событий, требующих коллегиального решения, и фиксирует результаты проведенных проверок в виде структурированных отчетов.

3.9. **Модуль администрирования и информационной безопасности**

Реализует ролевую модель доступа к данным, разграничивая права между категориями пользователей (врач-эксперт, заведующий отделением, главный врач), а также ролевую модель доступа к функциональным блокам (администратор, системный настройщик). Модуль обеспечивает полнотекстовое логирование (аудит) действий пользователей для обеспечения прослеживаемости процессов и защиты целостности данных.

4. Архитектура и стек технологий

4.1. **Тип архитектуры:** Клиент-серверная, сервисно-ориентированная архитектура с событийной обработкой данных (*рисунок 1. Архитектурная схема и модель взаимодействия компонентов программного комплекса «Компас»).*

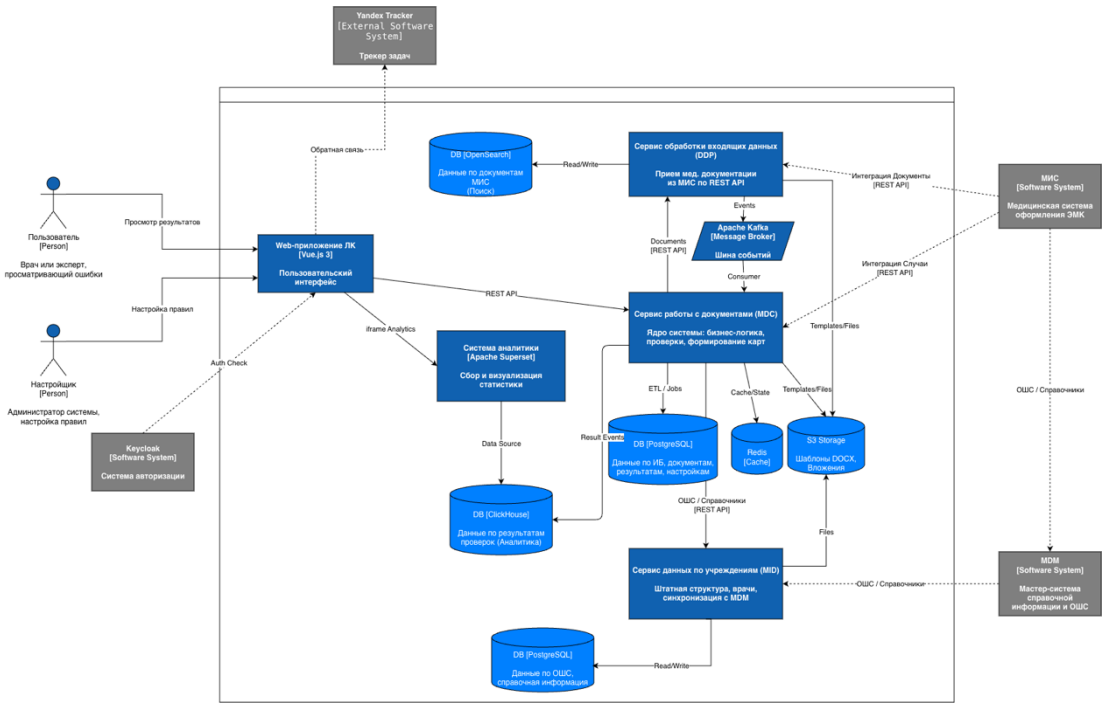

Рисунок 1. Архитектурная схема и модель взаимодействия компонентов программного комплекса «Компас»

4.2. **Клиентский уровень (Frontend):**

Единый веб-интерфейс, обеспечивающий визуализацию данных и управление процессами контроля качества. Взаимодействует с бэкенд-сервисами по протоколу HTTP/HTTPS через защищенный API.

4.2.1. **Сервисный уровень (Backend Services):**

Система разделена на три функционально независимых компонента (сервиса), каждый из которых имеет собственную зону ответственности и выделенные интеграционные интерфейсы:

4.2.1.1. Сервис приема и первичной обработки (DDP — Document Data Processing)

4.2.1.1.1. **Назначение:** обеспечение входного шлюза для потоковых данных из внешних систем.

4.2.1.1.2. **Логика работы:** сервис предоставляет REST-интерфейс для приема медицинских документов (протоколы приемов, результаты исследований).

4.2.1.1.3. **Интеграция:** взаимодействует с Медицинской информационной системой (МИС). Со стороны МИС настроено регламентное задание (Job), которое каждые 5 минут инициирует передачу новых документов в сервис DDP.

4.2.1.1.4. **Результат:** валидация входящих структур данных и постановка их в очередь для обработки в контуре контроля качества.

4.2.1.2. **Сервис контроля и формирования экспертиз (MDC — Medicine Documents Computing)**

4.2.1.2.1. **Назначение**: ядро системы, реализующее основную бизнес-логику проверок и управления случаями лечения.

4.2.1.2.2. **Логика работы:** осуществляет сборку отдельных документов в единые клинические случаи, выполняет автоматизированные проверки по заложенным правилам и формирует итоговые результаты экспертиз (карты контроля).

4.2.1.2.3. **Интеграция:** В составе общего программного комплекса MDC взаимодействует с другими сервисами по HTTP/REST API и через брокер сообщений Apache Kafka. Взаимодействует с МИС в части синхронизации состояний случаев лечения. Процесс инициируется на стороне МИС (активная сторона) с периодичностью один раз в

час, обеспечивая пакетную обработку обновлений (до 1000 случаев за одну итерацию).

4.2.1.2.4. **Результат:** актуальная база результатов контроля качества и безопасности медицинской деятельности.

4.2.1.3. **Сервис управления штатной структурой (MID — Medicine Institution Data)**

4.2.1.3.1. **Назначение:** управление нормативно-справочной информацией о структуре организации и персонале.

4.2.1.3.2. **Логика работы:** хранит и предоставляет другим сервисам эталонные данные о врачах, отделениях и должностях, используется для разграничения прав доступа к данным.

4.2.1.3.3. **Интеграция:** интегрирован с централизованной системой управления мастер-данными (MDM). Синхронизация обеспечивает единство идентификаторов сотрудников и подразделений между системой контроля качества и общебольничной (или региональной) инфраструктурой.

4.2.1.3.4. **Результат:** гарантия корректной привязки результатов экспертиз к конкретным исполнителям и структурным единицам.

4.3. **Технологический стек:**

4.3.1. **Язык программирования**: Python 3.12+ (Backend), JavaScript/TypeScript (Frontend).

4.3.2. **СУБД и хранение данных**: Система использует каскадную модель хранения данных: транзакционный слой на базе PostgreSQL 16, аналитический слой на базе ClickHouse для высокопроизводительной обработки больших объемов медицинских метрик, Redis для кэширования сессий и OpenSearch для обеспечения мгновенного полнотекстового поиска в электронных медицинских документах. Хранение неструктурированных данных (сканы, вложения) осуществляется в S3-совместимых хранилищах.

4.3.3. **Фреймворки и библиотеки**: Высокопроизводительный асинхронный фреймворк FastAPI, реактивная библиотека Vue.js 3, ORM-библиотека SQLAlchemy. Взаимодействие между сервисами реализовано через шину данных на базе Apache Kafka и протокол FastStream.

4.3.4. **Искусственный интеллект и NLP**: Применяется среда исполнения ONNX Runtime для локального (on-premise) запуска нейросетевых моделей анализа

текстов, что гарантирует сохранение врачебной тайны внутри контура организации без обращения к облачным сервисам.

4.3.5. **Веб-сервер:** При развёртывании вне Kubernetes внешним веб-сервером и обратным прокси служит Nginx; в Kubernetes маршруты публикуются через Nginx Ingress Controller. TLS-соединения завершаются на соответствующем компоненте. Приложение обслуживается ASGI-сервером Uvicorn. В состав поставки программного комплекса «Компас» входят три серверных сервиса: Medicine Documents Computing (MDC), Document Data Processing Service (DDP) и Medicine Institution Data (MID). Адреса сервисов и ключи доступа задаются в конфигурации среды развёртывания.

5. Состав программного обеспечения

5.1. Серверная часть MDC

Medicine Documents Computing (MDC) предоставляет интерфейс для описания структуры медицинских документов и правил их проверки, обрабатывает тексты входящих документов в соответствии с заданными структурами, запускает проверки и формирует расписания их выполнения.

- Backend и REST API для выполнения прикладных операций и интеграции с внешними системами;
- модуль обработки электронных медицинских документов;
- модуль ведения и обработки медицинской истории;
- модуль проверки полноты и качества медицинской документации;
- модуль формирования контрольных карт и результатов проверок;
- модуль формирования клинически значимых событий;
- обработчики асинхронных сообщений на базе FastStream и Apache Kafka;
- планировщик регламентных и фоновых заданий;
- модуль интеграции с МИС и смежными медицинскими сервисами по HTTP/REST API с использованием формата JSON;
- модуль интеграции с MID по HTTP/REST API для получения и кэширования справочных данных об организациях, подразделениях, сотрудниках и типах медицинских записей, а также для обогащения результатов проверок и данных пользователя;
- модуль интеграции с DDP по HTTP/REST API и Apache Kafka для получения уведомлений о сохранённых документах, загрузки текста и метаданных электронных медицинских документов по внутреннему или внешнему идентификатору, а также повторной загрузки при перезапуске обработки;

- слой доступа к PostgreSQL на базе SQLAlchemy и asyncpg;
- подсистема аналитического хранения на базе ClickHouse;
- подсистема кэширования и оперативного in-memory хранения данных на базе Redis, в том числе для прав доступа, регистров документов, дедупликации сообщений и заданий планировщика;
- модуль работы с S3-совместимым объектным хранилищем;
- опциональный модуль нейросетевого анализа медицинских текстов на базе ONNX Runtime;
- модуль формирования документов по шаблонам DOCX;
- модуль аутентификации и авторизации с интеграцией с Keycloak;
- средства журналирования, мониторинга и трассировки на базе Sentry и OpenTelemetry.

5.2. **Серверная часть DDP**

Document Data Processing Service (DDP) выполняет приём, обработку и хранение входных данных медицинских документов, поступающих из МИС. API сервиса определяет формат входных данных и предоставляет методы для сторонних клиентов и внутренних сервисов, работающих с полученными документами.

- API, определяющий формат входных данных медицинских документов;
- методы API для сторонних клиентов и внутренние методы для сервисов программного комплекса;
- модуль обработки и хранения документов и связанных с ними данных;
- подсистема хранения данных: PostgreSQL для метаданных; Elasticsearch или OpenSearch для индексирования и полнотекстового поиска; Redis для кэширования, состояний задач, дедупликации, расписаний и оперативного in-memory хранения; S3-совместимое объектное хранилище для файлов документов;
- модуль публикации уведомлений о сохранённых документах в Apache Kafka для последующей обработки в MDC.

5.3. **Серверная часть MID**

Medicine Institution Data (MID) служит точкой синхронизации данных медицинских учреждений для их последующего отображения в системе «Медконтроль» и настройки прав доступа пользователей.

- API синхронизации и выдачи данных о медицинских учреждениях, их подразделениях и сотрудниках;
- модуль подготовки данных медицинских учреждений для отображения в системе «Медконтроль»;

- модуль настройки и контроля прав доступа пользователей;
- подсистема хранения данных: PostgreSQL для данных медицинских учреждений; Redis для заданий планировщика, краткоживущих состояний, кэширования и оперативного in-memory хранения; S3-совместимое объектное хранилище для файлов синхронизации;
- средства журналирования, мониторинга и контроля доступности API.

5.4. **Пользовательский веб-интерфейс**

Пользовательский веб-интерфейс программного комплекса «Компас» разработан на Vue.js 3 и TypeScript. Для сборки используется Vite, для маршрутизации — Vue Router, для управления состоянием — Pinia; интерфейс построен с применением разработанной внутри компании компонентной библиотеки и стилей SCSS. Vue.js, TypeScript, Vite, Vue Router и Pinia являются свободным программным обеспечением с открытым исходным кодом (Open Source).

6. Системные требования

6.1. **Серверная часть**

Минимальная конфигурация для пилотной или тестовой установки при размещении PostgreSQL, Kafka, ClickHouse и объектного хранилища на отдельных серверах:

- процессор архитектуры x86-64: не менее 4 виртуальных ядер;
- оперативная память: не менее 8 ГБ;
- свободное место на диске: не менее 50 ГБ на SSD;
- сетевое подключение к инфраструктурным сервисам;
- контейнерная среда Docker, Podman или Kubernetes.

Минимальная конфигурация для промышленной установки при размещении PostgreSQL, Kafka, ClickHouse и объектного хранилища на отдельных серверах:

- процессор архитектуры x86-64: не менее 8 виртуальных ядер;
- оперативная память: не менее 16 ГБ;
- при использовании модуля нейросетевого анализа: не менее 24 ГБ оперативной памяти;
- свободное место на диске: не менее 100 ГБ на SSD;
- сетевое подключение к инфраструктурным сервисам со скоростью не менее 1 Гбит/с;
- операционная система семейства Linux;
- контейнерная среда Docker, Podman или Kubernetes, в том числе их российские дистрибутивы, такие как Deckhouse, Боцман, Nodus, и.т.д.

Указанная конфигурация рассчитана на запуск одного экземпляра API и ограниченного количества обработчиков фоновых заданий при небольшой или средней интенсивности обработки медицинских документов.

Для обеспечения отказоустойчивости рекомендуется использование не менее двух узлов приложения, каждый из которых соответствует указанным минимальным требованиям. Точное количество обработчиков и экземпляров приложения определяется по результатам нагрузочного тестирования.

Точные требования к промышленной среде определяются количеством обрабатываемых медицинских документов, числом одновременно работающих пользователей, сроками хранения данных и результатами нагрузочного тестирования.

6.2. **Клиентское рабочее место**

Установка специализированного программного обеспечения MDC на АРМ врача не требуется. Доступ к пользовательскому интерфейсу осуществляется через веб-браузер отдельного клиентского компонента.

Рекомендуемые требования:

- процессор x86-64;
- не менее 4 ГБ оперативной памяти;
- разрешение экрана не менее 1366 x 768;
- Яндекс Браузер, chromium-gost, или аналогичный веб браузер с поддержкой JavaScript, cookies и TLS 1.2 или выше

Для клиентской части поддерживаются любые операционные системы, поддерживающие совместимый веб-браузер.

6.3. **Совместимость с операционными системами**

Серверная часть поставляется в виде Linux-контейнеров. Развёртывание допускается на любых дистрибутивах, при наличии совместимой контейнерной платформы, в том числе дистрибутивах отечественной разработки, таких как:

- Astra Linux;
- РЕД ОС;
- ALT Linux.

Базовый контейнерный образ программного обеспечения может быть пересобран на основе Debian-совместимого дистрибутива Linux российского происхождения, например Astra Linux. Перед промышленной эксплуатацией такого образа проводится проверка сборки Python 3.14, нативных библиотек и функциональное тестирование программного обеспечения.

Серверная совместимость обеспечивается контейнерным способом поставки; конкретные версии ОС и контейнерной платформы фиксируются в эксплуатационной документации после проведения испытаний.

7. Обеспечение информационной безопасности

7.1. **Идентификация и аутентификация**. Доступ пользователей осуществляется с применением централизованной системы аутентификации Keycloak и токенов доступа. Для взаимодействия внутренних сервисов предусмотрена отдельная аутентификация по сервисному токену.

7.2. **Разграничение прав доступа**. Реализована ролевая модель управления доступом. Предусмотрены роли администратора, эксперта и специалиста по настройке. Дополнительно применяется ограничение области доступных данных по медицинской организации, подразделению и врачу.

7.3. **Защита передаваемых данных**. Внешние подключения к системе защищаются протоколом HTTPS с использованием SSL/TLS. Терминация TLS-соединений выполняется на уровне Nginx, а при развёртывании в Kubernetes — на уровне Nginx Ingress Controller. Используется TLS версии 1.2 или выше.

Внутренние соединения между PostgreSQL, Kafka, ClickHouse, Redis, МИС и компонентами системы «Компас» размещаются в изолированном сетевом сегменте. Для этих соединений дополнительно настраивается TLS.

7.4. **Защита конфиденциальных параметров**. Пароли, токены, ключи доступа и иные секреты не включаются в исходный код или контейнерные образы. Их передача выполняется через защищённые переменные окружения и специализированное хранилище секретов. При промышленном развёртывании поддерживается использование Yandex Lockbox или аналогичного средства.

7.5. **Журналирование и аудит**. Система регистрирует действия пользователей, обращения к API и технические события. Для мониторинга и трассировки используются Sentry и OpenTelemetry. Доступ к журналам предоставляется только уполномоченным администраторам.

7.6. **Защита среды исполнения**. Финальные рабочие контейнеры системы запускаются от имени системного пользователя nonroot. При промышленном развёртывании компоненты исполняются в контейнерах Kubernetes и распределяются по отдельным модулям развёртывания (Deployment/Pod). Сервис приложения имеет внутренний тип ClusterIP и непосредственно во внешнюю сеть не публикуется. Внешний доступ к API предоставляется через контролируемый сетевой шлюз: Nginx при развёртывании вне Kubernetes либо Nginx Ingress

Controller при развёртывании в Kubernetes, с использованием TLS. Прямая публикация порта допускается только в локальной тестовой конфигурации Docker Compose.

7.7. **Отсутствие недекларированных возможностей**. Архитектурой программного обеспечения не предусматриваются скрытые механизмы удалённого управления, недекларированные учётные записи, обход штатной аутентификации либо иные скрытые функции. Отсутствие недекларированных возможностей подтверждается проверкой исходного кода, статическим анализом безопасности, контролем состава зависимостей и приёмочными испытаниями соответствующей версии программного обеспечения.