Описание функциональных характеристик программного обеспечения СЕРВИС "РЕЕСТР-КОНТРОЛЬ"

1. Общее описание ПО

СЕРВИС "РЕЕСТР-КОНТРОЛЬ" — это веб-приложение, разработанное для автоматизированной проверки корректности медицинских реестров, формируемых медицинскими организациями для сдачи в Территориальные фонды обязательного медицинского страхования (ТФОМС).

Система состоит из двух основных компонентов:

- веб-интерфейс на PHP/Laravel для взаимодействия с пользователями;
- ядро обработки данных на Python для выполнения бизнес-логики валидации.

2. Основные функциональные модули

2.1 Модуль валидации ошибок (errors.py)

Назначение: основной модуль для проверки корректности заполнения медицинских реестров.

Типы реестров, обрабатываемые модулем:

- HM* — Госпитализация (основной);
- CM* — Стоматология (основная);
- DPM, DVM, DOM, DFM, DAM, DBM, DUM* — Дневники (поликлиника, стоматология и др.);
- DD, DE — Дневники (диспансеризация, экстренная помощь).

Валидации, выполняемые модулем:

- проверка существования страховой организации (СМО);
- проверка актуальности СМО (дата закрытия);
- проверка уникальности полиса пациента (NPOLIS);
- проверка типа документа ОМС (VPOLIS);
- проверка корректности даты рождения;
- проверка заполненности документа, удостоверяющего личность (паспорт);
- проверка диагнозов по МКБ-10 (существование, подстрочники);
- проверка диагнозов Z00.1, Z00.2, Z00.3 в зависимости от возраста пациента;
- проверка соответствия диагноза профилю лечения (вне ТП);
- проверка паспортных данных (длина номера паспорта);
- проверка корректности дат документа;
- проверка дублирования ранее оплаченных ДВН/ПВН;
- проверка пересечения дат госпитализации с амбулаторными посещениями;
- проверка корректности заполнения диагноза онкологического заболевания;

- проверка пересечения дат посещений специалистов одного профиля в один день;
- проверка заполнения памяток о предоставлении медицинской помощи;
- проверка корректности даты направления;
- и многие другие (более 40 типов валидаций).

Результат: файл отчёта в формате .xlsx, содержащий:

- всю строку реестра с ошибкой;
- информацию о пациенте (фамилия, имя, отчество, пол, дата рождения, полис);
- информацию о враче (специальность, ФИО);
- информацию об услуге (код, дата);
- описание ошибки.

2.2 Модуль проверки подсчётов (count.py)

Назначение: автоматический подсчёт показателей по реестрам для формирования справочной информации и контроля корректности данных.

Типы реестров, обрабатываемые модулем:

- HT, CT, TT* — Стационар, дневник, скорая помощь;
- DPT, DVT, DOT, DFT, DAT, DBT, DUT, DST — Дневники (различные профили);
- DD, DE — Дневники (диспансеризация, экстренная помощь).

Функциональность:

- автоматический подсчёт количества записей по различным параметрам;
- проверка корректности вычисленных показателей;
- формирование отчётов по подсчётам для последующего анализа.

Результат: файл .xlsx с подсчитанными показателями и контрольными суммами.

2.3 Модуль проверки B-M файлов (vm.py)

Назначение: валидация и исправление B-M файлов (выписных данных), включая перекодировку диагнозов и услуг согласно современным классификаторам.

Типы реестров, обрабатываемые модулем:

- HM, CM — Основные реестры (стационар, стоматология);
- VM, VPM, VVM, VOM, VFM, VAM, VBM, VUM, VSM, VDM, VEM* — Выписные данные.

Функциональность:

- проверка корректности данных в B-M файлах;
- перекодировка по стандарту V021 (классификатор медицинских специальностей);
- перекодировка по стандарту V025 (классификатор целей посещения);
- применение правил перекодировки diagnoses и procedure;
- формирование исправленного архива с правильными кодами.

Результат:

- архив с исправленными файлами (No_FLK*.zip);
- отчёт об ошибках в формате .xlsx.

2.4 Модуль обработки санкций МЭК (sank_mek.py)

Назначение: проверка санкционных файлов МЭК на соответствие нормативам и корректность данных.

Типы реестров, обрабатываемые модулем:

- HT, CT, TT, DPT, DVT, DOT, DFT, DAT, DBT, DUT, DST, DD, DE*.

Функциональность:

- проверка корректности санкционных записей;
- проверка соответствия данных в санкционных файлах;
- формирование отчётов по выявленным несоответствиям.

Результат: файл .xlsx с информацией о санкциях и выявленных ошибках.

2.5 Модуль проверки структуры реестров (structure.py)

Назначение: проверка структуры реестров и формирование списков для отправки в различные инстанции.

Типы реестров, обрабатываемые модулем:

- HT, CT, TT, DPT, DVT, DOT, DFT, DAT, DBT, DUT, DST, DD, DE*.

Функциональность:

- проверка корректности структуры файлов;
- формирование списков пациентов с различными параметрами (отказы, госпитализации, профилактика);
- разделение данных по инстанциям (РСПО, ФОМС, ОМС).

Результат:

- архив с выходными файлами и списками;
- отчёт в формате .xlsx.

2.6 Модуль формирования списков ДН (dn_list.py)

Назначение: формирование списков пациентов, находящихся на диспансерном наблюдении.

Типы реестров, обрабатываемые модулем: D* — Дневники (все профили).

Функциональность:

- выделение пациентов на диспансерном наблюдении;
- формирование отдельных списков по различным параметрам;
- проверка корректности перечня услуг.

Результат:

- ZIP-архив с списками ДН (DN*.ZIP);
- отчёт в формате .xlsx.

2.7 Модуль формирования списков ДН Loose (dn_list_loose.py)

Назначение: формирование упрощённых списков ДН для специальных случаев.

Типы реестров, обрабатываемые модулем: HT, CT — Стационар.

Функциональность:

- формирование упрощённых списков ДН;
- проверка соответствия указанным критериям.

Результат: ZIP-архив с упрощёнными списками.

2.8 Модуль отчётов по врачам (doctors_report_module.py)

Назначение: формирование отчётов по активности врачей для анализа работоспособности и загрузки.

Типы реестров, обрабатываемые модулем: DPT, DOT, DDT, DAT — Дневники (поликлиника, стоматология, др.).

Функциональность:

- сбор данных об услугах, оказанных врачами;
- формирование отчётов по врачу (ФИО, специальность, код услуги, количество услуг);
- формирование сводных отчётов по учреждению.

Результат: файл .xlsx с отчётами по врачам.

2.9 Модуль отчётов по врачам для школ и ДН (doctors_report_dn_school.py)

Назначение: формирование отчётов по врачам для учреждений школ и ДН.

Типы реестров, обрабатываемые модулем: HT, CT — Стационар.

Функциональность:

- выделение услуг, оказанных в учреждениях школ и ДН;
- формирование специализированных отчётов.

Результат: файл .xlsx с отчётами.

2.10 Модуль экспорта госпитализаций (export_komtek_hosp.py)

Назначение: экспорт данных о госпитализации для интеграции с системой Комтек.

Типы данных: CSV-файлы (источник).

Функциональность:

- сбор данных о госпитализации из внешних источников;
- обогащение данными из реестров ТФОМС;
- формирование выходных CSV и Excel файлов.

Результат:

- CSV файлы (tmp*.csv);
- Excel файл с отчётом (.xlsx).

2.11 Модуль сбора истории ДВН/ПВН (fill_d_history.py)

Назначение: сбор и хранение информации о ранее оказанной медицинской помощи для проверки дублирования.

Функциональность:

- сбор данных о визитах пациентов из различных реестров;
- хранение в базе данных для последующей проверки;
- поддержка задания периода сбора данных (начальная и конечная даты).

Результат: информация сохраняется в базу данных, выводится отчёт о количестве записей.

3. Модули работы с данными

3.1 Модуль обновления справочников (update_dir.py)

Назначение: автоматическое обновление справочных данных НСИ (нормативно-справочная информация).

Поддерживаемые справочники:

- F014 — специальности врачей;
- F002 — СМО (страховые медицинские организации);
- F032 — ЛПУ (лечебно-профилактические учреждения).

Функциональность:

- выгрузка справочников из внешних источников (API, файлы);
- конвертация в сжатые датафреймы для быстрого доступа;
- обновление базы данных.

3.2 Модуль конвертации XML (make_df.py)

Назначение: конвертация XML-файлов справочников в сжатые бинарные форматы.

Функциональность:

- парсинг XML справочников;
- создание оптимизированных датафреймов;
- сохранение в сжатом формате (.gzip, .pickle).

3.3 Модуль обновления БД госпитализации (fill_hosp_db.py)

Назначение: обновление базы данных о госпитализациях.

Функциональность:

- сбор данных из различных источников;
- интеграция с существующими записями;
- обновление историй госпитализаций.

4. Модули вспомогательного назначения

4.1 error_utils.py

Содержит вспомогательные функции для работы с ошибками:

- объединение ошибок в строку;
- извлечение информации о врачах;
- проверка пересечения дат госпитализации;
- работа со справочниками МКБ-10;
- извлечение информации о закрытии ЛПУ;
- фильтрация пациентов по наличию ЭЦП.

4.2 controls.py

Модуль валидационных функций:

- check_corrected_primary_reestr_entries() — проверка исправленных записей;
- check_usl_exists_only_once() — проверка уникальности услуг;
- check_already_paid_dvn_pvn() — проверка дублирования ДВН/ПВН;
- check_passport_num_len() — проверка паспортных данных;
- check_usl_payment_method() — проверка способа оплаты.

4.3 config.py

Конфигурация системы:

- настройки проверок (включение/выключение отдельных модулей);
- префиксы допустимых имён файлов для каждого модуля;
- пути к файлам и справочникам;
- уровень логирования.

4.4 refs.py

Работа со справочниками:

- загрузка и хранение справочных данных;
- доступ к справочникам по кодам;
- кэширование справочников.

4.5 d_history.py

Работа с историей ДВН/ПВН:

- хранение и поиск историй визитов;
- методы поиска дублирующихся записей;
- поддержка различных типов реестров.

4.6 hosp.py

Работа с госпитализацией:

- хранение и поиск случаев госпитализации;
- проверка пересечений дат;
- интеграция с КОМТЕК.

4.7 pers.py

Работа с персоналом:

- хранение данных о врачах;
- получение ФИО врача по коду;
- интеграция с внешними системами.
-

5. Веб-интерфейс (Laravel)

5.1 Маршруты загрузки файлов

- /upload/vm — загрузка В-М файлов;
- /upload/count — загрузка для подсчётов;
- /upload/sank-mek — загрузка санкционных файлов;
- /upload/errors — загрузка для валидации ошибок;

- /upload/structure — загрузка для проверки структуры;
- /upload/doctors-report — загрузка для отчётов по врачам;
- /upload/doctors-report-dn-school — загрузка для отчётов по ДН/школам;
- /upload/dn_list — формирование списков ДН;
- /upload/dn_list_loose — формирование упрощённых списков ДН;
- /upload/komtek_hosp — экспорт госпитализаций (для администраторов);
- /fill/d_history — сбор истории ДВН/ПВН (для администраторов).

5.2 Модели данных (Laravel Eloquent)

- User — пользователи системы (администраторы и обычные пользователи);
- Task — задачи обработки реестров;
- News — новости системы;
- NewsRead — прочитанные пользователем новости;
- ClinicBlacklist — список заблокированных медицинских организаций;
- UserArch — архив пользователей (удалённые аккаунты).

5.3 Контроллеры

- LoginController — аутентификация пользователей;
- UploadController — обработка загрузки файлов и запуск Python-скриптов;
- TaskController — загрузка списка задач с фильтрацией;
- DownloadController — скачивание результатов обработки;
- UsersController — управление пользователями (для администраторов);
- NewsController — управление новостями;
- InstructionController — управление инструкциями;
- ReportsController — редактирование отчётов.

5.4 Виды задач (status)

- NEW — новая задача, в очереди на выполнение;
- RUN — задача выполняется Python-скриптом;
- SUCCESS — задача успешно завершена;
- ERROR — задача завершилась с ошибкой.

6. Интеграция с внешними системами

6.1 Базы данных

- MySQL — основная база данных Laravel (пользователи, задачи, справочники);
- PostgreSQL — база данных для справочной информации и госпитализаций;
- Redis — кэширование (blacklist, справочники).

6.2 Внешние API

- API АРМа — получение данных о клиниках и параметрах;
- remd API — проверка наличия ЭЦП у пациентов;

- API НСИ — получение актуальных справочников (F002, F014, F032).

7. Форматы обрабатываемых файлов

7.1 Входные форматы

- ZIP-архивы — архивы с файлами реестров;
- CSV-файлы — для модуля экспорта госпитализаций.

7.2 Выходные форматы

- .xlsx — основной формат отчётов (Excel);
- .zip — архивы с исправленными файлами;
- .csv — для экспорта госпитализаций.

7.3 Форматы файлов внутри архивов

- HM, CM, D* — основные файлы реестров (txt/csv);
- HT, CT* — дневники и стационар;
- DM, WM, RM, UM* — списки для ДН.

8. Системные требования

8.1 Серверная часть (Python)

Python 3.8+

Библиотеки:

- pandas 1.5.2;
- openpyxl 3.0.10;
- psycopg2-binary 2.9.6;
- mysql-connector-python 8.0.33;
- redis 4.5.5;
- и др. (см. requirements.txt).

8.2 Веб-интерфейс (PHP/Laravel)

- PHP 8.0+;
- Laravel 9+;
- MySQL 5.7+ или PostgreSQL 12+;
- Redis 6.0+;
- веб-сервер: Nginx или Apache.

8.3 Дополнительные требования

- Docker и docker-compose (для контейнерной установки);
- Git для управления версиями.

9. Безопасность

9.1 Аутентификация и авторизация

- система использует токены для аутентификации;
- разделение пользователей на администраторов и обычных пользователей;
- обычные пользователи видят только свои задачи;

- администраторы имеют доступ ко всем задачам и функциям системы.

9.2 Проверка файлов

- проверка соответствия имени файла шаблону;
- проверка принадлежности файла к заблокированной МО (через blacklist);
- проверка типов файлов.

9.3 Логирование

- логирование всех операций в файлы;
- логирование ошибок выполнения Python-скриптов;
- логирование веб-запросов через Laravel.

10. Источники данных и справочники

10.1 Основные справочники

Для валидации:

- диагнозы вне ТП в поликлинике (unpaid_oms_ds_pol.csv);
- диагнозы вне ТП в стационаре (unpaid_oms_ds_stac.csv);
- соответствие профиля медпомощи и специальности врача (profprvs.csv);
- кратность услуг (usl_frequency.csv);
- услуги первичного и повторного приёма (pair_of_usl_in_one_day.csv);
- услуги в основном и диспансерном реестрах (path_to_pair_d_h_usl_codes.csv);
- список кодов МО без доступа (black_list.csv).

Для перекодировок:

- классификатор медицинских специальностей V021 (v021.json);
- номенклатура медицинских услуг (medical_services.json);
- V025 классификатор целей посещения (v025.json).

10.2 Обновляемые справочники (через update_dir.py)

- F014 — классификатор медицинских специальностей;
- F002 — страховые медицинские организации;
- F032 — коды ЛПУ и даты их закрытия.

10.3 Данные о госпитализации

- база данных PostgreSQL с историями госпитализаций;
- интеграция с системой Комтек для дополнительных данных.

11. Коды возврата Python-скриптов

- 0 — успешное завершение;
- 64 — не найден датафрейм F002;
- 65 — отсутствует файл проф/prvs;
- 66 — не найден датафрейм МКБ-10;
- 67 — не найден словарь F032;
- 68 — нарушена структура файлов реестра;

- 69 — не найден список диагнозов вне ТП;
- 70 — справочник не найден;
- 113 — неожиданная ошибка (исключение).

12. Архитектура обмена между модулями

Веб-интерфейс (Laravel, PHP 8.0+, Nginx/Apache) принимает файлы реестра через веб-форму. Обработчик загрузки (UploadController) проверяет файл, создаёт задачу в базе данных и копирует файлы в storage/tasks/{task_id}/. Далее вызывается Python-скрипт (exec('python3 module.py')), который читает данные из ZIP-архива, загружает справочники (pandas DataFrames), выполняет валидацию и обработку данных, формирует отчёт (.xlsx). После этого статус задачи в базе данных обновляется (SUCCESS/ERROR), и пользователю становятся доступны результаты для скачивания.

13. Инструкции по использованию

13.1 Основной процесс работы

- Пользователь авторизуется в системе;
- переходит в раздел загрузки;
- выбирает тип реестра;
- загружает ZIP-архив с файлами;
- система автоматически запускает обработку;
- пользователь получает уведомление о завершении;
- пользователь скачивает результаты обработки.

13.2 Для администраторов

- управление пользователями (добавление, удаление, архивация);
- настройка справочников;
- обновление баз данных;
- просмотр всех задач системы;
- генерация отчётов по всем пользователям.

14. Ограничения и особенности

- длина периода запроса задач не более 1 месяца;
- максимальный размер загружаемого файла ограничен настройками PHP/Nginx;
- система не поддерживает параллельную обработку одного файла несколькими пользователями;
- история ДВН/ПВН собирается за период, заданный пользователем (по умолчанию — отчётный месяц).

15. Модульность и расширяемость

Система разработана с использованием модульного принципа:

- каждый тип реестра обрабатывается своим модулем;
- легко добавлять новые типы валидаций;
- легко добавлять новые модули обработки;

- все модули используют общие библиотеки и вспомогательные функции.