/МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Вятский государственный университет»**

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Допущено к защите

Руководитель проекта

\_\_\_\_\_\_\_\_/Исупов К.С./

(подпись) (Ф.И.О)

«\_\_»\_\_\_\_\_\_\_\_\_\_202\_г.

Реализация алгоритмов миграции данных и экспериментальная апробация модуля

Пояснительная записка курсового проекта по дисциплине

«Комплекс знаний бакалавра в области программного и аппаратного обеспечения вычислительной техники»

ТПЖА.09.03.01.723 ПЗ

Разработал студент группы ИВТ-41 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Кудяшев Я.Ю./

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Исупов К.С./

Консультант \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Долженкова М.Л./

Работа защищена с оценкой «\_\_\_\_\_\_\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(оценка) (дата)

Члены комиссии \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ \_\_\_\_\_\_\_\_\_\_\_/

(подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /\_\_\_\_\_\_\_\_\_\_\_\_/

(подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /\_\_\_\_\_\_\_\_\_\_\_\_/

(подпись)

Киров 2023

Реферат

Кудяшев Я.Ю. Реализация алгоритмов миграции данных и экспериментальная апробация модуля.

. ТПЖА.090301.021 ПЗ: Курс. проект / ВятГУ, каф. ЭВМ; рук. Исупов К.С. - Киров, 2023. – ПЗ 26с, 14 рис..

Объект курсового проекта – платформы для автоматизации

Предмет курсового проекта – экспорт данных.

Цель курсового проекта – облегчить реализацию решений для заказчиков и уменьшить затрачиваемое на перенос данных время.

Результатом выполнения курсового проекта является модуль для экспорта данных с платформ, имеющих разную архитектуру.

Содержание

[ВВЕДЕНИЕ 6](#_Toc132301474)

[1 Обзор предметной области 7](#_Toc132301475)

[1.1 Общие сведения 7](#_Toc132301476)

[1.1.1 Полное наименование системы и её условное обозначение 7](#_Toc132301477)

[1.1.2 Наименование разработчика системы и реквизиты заказчика 8](#_Toc132301478)

[1.1.3 Основания для создания системы 8](#_Toc132301479)

[1.1.4 Плановые сроки начала и окончания работы по созданию системы. 8](#_Toc132301480)

[1.1.5 Источник финансирования работ по созданию системы 8](#_Toc132301481)

[1.2 Назначение и цели создания системы 8](#_Toc132301482)

[1.2.1 Назначение системы 8](#_Toc132301483)

[1.2.2 Цели создания системы 8](#_Toc132301484)

[1.3 Характеристики объекта автоматизации 8](#_Toc132301485)

[1.3.1 Краткие сведения об объекте автоматизации 8](#_Toc132301486)

[1.4 Требования к системе 9](#_Toc132301487)

[1.4.1 Перспективы развития, модернизации системы 9](#_Toc132301488)

[1.4.2 Требования к надежности структуры 9](#_Toc132301489)

[1.4.3 Требования к эксплуатации 9](#_Toc132301490)

[1.4.4 Требования к функциям, выполняемым системой 9](#_Toc132301491)

[1.5 Выводы 9](#_Toc132301492)

[2 Алгоритмы функционирования 10](#_Toc132301493)

[2.1 Алгоритм работы модуля 10](#_Toc132301494)

[2.1.1 Получениие токена пользователя 10](#_Toc132301495)

[2.1.2 Получение элементов из ELMA3 12](#_Toc132301496)

[2.1.3 Формирование элементов в ELMA365 16](#_Toc132301497)

[2.2 Алгоритм формирования объектов 19](#_Toc132301498)

[2.2.1 Простые объекты 19](#_Toc132301499)

[2.2.2 Составные объекты 21](#_Toc132301500)

[2.3 Выводы 22](#_Toc132301501)

[3 Апробация модуля 23](#_Toc132301502)

[3.1 Тестирование методом «белого ящика» 23](#_Toc132301503)

[3.2 Выводы 24](#_Toc132301504)

[ЗАКЛЮЧЕНИЕ 25](#_Toc132301505)

[СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 26](#_Toc132301506)

ВВЕДЕНИЕ

В последние годы всё больше теряет спрос ручной труд и растёт спрос на автоматизацию. Больше всего данная тенденция затронула бизнес: люди стремятся свести ручной труд к нулю и автоматизировать процессы как внутри компании, так и снаружи. Это, в первую очередь, необходимо для экономии времени и, как следствие, получения большей прибыли.

На сегодняшний день существуют компании по автоматизации бизнеса и бизнес-процессов, которые помогают компаниям сэкономить время, облегчить коммуникацию внутри компании и получить большую прибыль. Лидеров в данной отрасли среди стран СНГ является компания ELMA. Она занимается разработкой систем для управления бизнес-процессами. На данный момент компания имеет в распоряжении несколько платформ: ELMA 3, ELMA4 и ELMA365. Последняя является наиболее ходовым продуктом компании, которая обгоняет своих предшественников в функционале и гибкости настройки. В связи с этим, у многих компаний появилась потребность в переходе с платформ ELMA3/4 на ELMA365. Основная проблема при переходе с одной платформы на другую – различные архитектуры. Помимо этого, компаниям необходимо перенести свои старые данные на новую платформу, где возникают такие же проблемы.

Решением данной проблемы может послужить модуль для внедрения в систему, который будет переносить данные с одной площадки на другую вне зависимости от структуры решения, т.е. выполнять автоматизированный экспорт данных. Перед реализацией необходимо подробно изучить архитектуры обоих площадок, найти их различия и придумать пути решения проблемы по переносу данных, чему и посвящён данный курсовой проект.

1. Анализ предметной области

На данном этапе работы необходимо рассмотреть функции модуля миграции данных для BPM платформ с разными архитектурами, провести обзор аналогов, существующих на сегодняшний день, их возможностей по экспорту данных, выяснить их недостатки и обосновать актуальность разработки текущего модуля.

* 1. Модуль миграции данных

Функциями модуля миграции данных для BPM платформ с разными архитектурами являются:

* авторизация на платформе ELMA3 и получение токена пользователя с правами администратора;
* получение структуры элементов, хранящихся на платформе ELMA3, и их значений;
* получение и просмотр структуры элементов платформы ELMA365;
* возможность вручную соотносить свойства экспортируемых элементов с платформы ELMA3 со свойствами структуры в ELMA365;
* возможность автоматически соотнести свойства экспортируемых элементов платформы ELMA3 со свойствами структуры в ELMA365 на основе типов данных с дальнейшим их ручным изменением;
* создание новых элементов в ELMA365 с соответствующими данными и структурой элементов из ELMA3;
* присвоение признака экспортированного элемента новым на платформе ELMA365.
  1. Обзор аналогов

На данном этапе необходимо рассмотреть текущие возможности и инструменты для миграции данных между платформами, проанализировать их функционал, составить список основных недостатков и ограниченностей моделей и сделать выводы по данной части.

1. Стандартный импорт данных

На сегодняшний день на любую площадку платформы ELMA365 можно без особого труда стандартными средствами перенести данные в любое приложение раздела путем подготовки, структурированной заранее таблицы Excel или файла формата .csv. На рисунке 1 приведен пример формы для приема импортируемого файла в ELMA365.

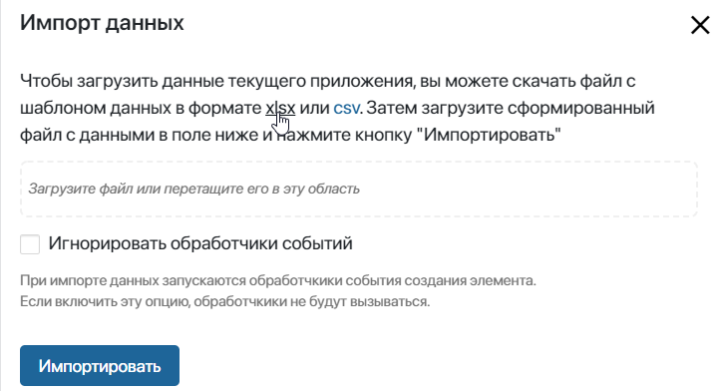


Рисунок 1 - Форма импорта данных в ELMA365

Общий процесс импорта данных в систему таким путём состоит из следующих шагов:

* Создание приложения, в котором будут храниться данные. Данный этап подразумеваем подготовку структуры, которая должна удовлетворять потребностям как заказчика, так и импортируемых объектов;
* Генерация шаблона документа. На этом этапе система предложит пользователя ознакомиться с шаблоном, которому должен удовлетворять файл с импортируемыми данными в формате .xlsx или .csv;
* Приведение импортируемого файла в соответствие с шаблоном. Этап, на выполнение которого требуется больше всего времени, т.к. данные должны быть получены и приведены в строго соответствующий шаблону формат;
* Загрузка подготовленного файла с данными в систему. На последнем этапе происходит перенос данных в систему и занесение их в таблицы в базе данных PostgreSQL.

Необходимо обратить внимание, что индексы создаются в БД для каждого свойства приложения, что значительно увеличивает время импорта данных. Наличие любого несоответствия типов данных мгновенно прерываем процесс импорта. На рисунке 2 приведен пример неудачного импорта данных, с наличием несоответствия типов данных.

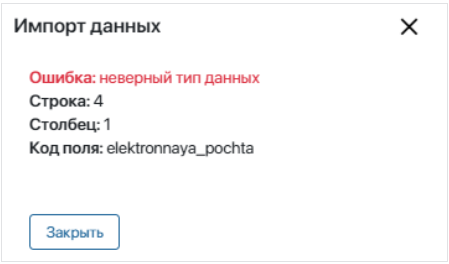


Рисунок 2 - Ошибка при импорте данных

Главными особенностями такого импорта является отсутствие возможности редактирования структуры файла прямо на платформе, что увеличиваем время для обрабокти данных, а также необходимость указания ссылок на приложения в виде идентификаторов уже существующих в системе элементов, что делает данный спопоб практически нелегетимным ля импорта данных из ELMA3, т.к. элементов, на которые должна ссылаться система, ещё нет.

1.2.2 Интеграционное приложение с ELMA365

Относительно надавно в ELMA365 Store появилось решение, позволяющее осуществлять миграцию данных с платформы ELMA3/4 на платформу ELMA365. Данное решение представляет из себя модуль для платформы ELMA3/4.

К ключевым особенностям и ограничениям данного модуля относится:

* Привязка к платформе ELMA3/4. Для взаимодействия и контроля импорта данных;
* Платная основа. Модуль является платным решением и находится в магазине ELMA Store;
* Отсутствие интерфейса. Нет как такого интерфейса для взаимодействия с пользователем. Модуль работает в виде консольного приложения;
* Миграция данных для системных разделов. Импорт данных возможен только для системных разделов, которые идут в стандартной поставке ELMA365.

После анализа функционала и особенностей можно сделать вывод, что данный модуль подойдёт для компаний, которые пользуются стандартными решениями компании ELMA и не имеют в своём распоряжении решений, подготовленных специально для их компании: со своей структурой, свойствами и особенностями.

1.2.3 Недостатки аналогов

Рассмотрев аналоги и обозрев их функционал можно точно сказать, что они имеют большое количество недостатков и ограниченного функционала. К основным относятся:

* отсутствие интерфейса для управления процессом экспорта данных. Для экспорта данных необходимо вручную приводить экспортируемые сущности к виду, соответствующему структуре на платформе ELMA365;
* отсутствие массового импорта данных на уровне всех разделов и приложений. Одновременно можно импортировать данные лишь в одно приложение раздела, что заставляет тратить на данный процесс огромное количество времени;
* отсутствие явных признаков импортированных объектов. Нет чётких индикаторов в системе, говорящих о принадлежности объектов другой платформе, что иногда затрудняет работу с ними;
* неявный формат ссылок на элементы при импорте данных. Как было сказано выше, для импорта связей элементов с другими элементами системы, необходимо прописывать идентификаторы этих объектов. При этом, нельзя сослаться на элемент, который ещё отсутствует в системе;
* дублирование сущностей. При импорте сущностей в систему нет проверки на дубли, что значительно осложняет работу в систему в виду одинаковых элементов;
* ненадёжность инструментов импорта. При возникновении технических проблем во время импорта сущностей, нет возможности продолжить импорт с того элемента, на котором он был прерван;
  1. Актуальность разработки

Модуль, позволяющий осуществить плавный перенос данных со старой платформы на новую на сегодняшний день имеет максимальный приоритет в разработке. Это, в первую очередь, связано с потребностью клиентов в совершенствовании и плавным переходом со старого продукта компании ELMA3, архитектура которого представляет из себя монолит, на новую платформу ELMA365, являющуюся микро сервисным решением. На данный момент все больше клиентов уже проектируют решения на новой платформе и после завершения данной процедуры необходим плавный перенос данных компаний со старой платформы на новую с учётом новой архитектуры и свойств системы, для чего и предназначен разрабатываемый модуль.

Выводы по разделу

Проведя анализ предметной области можно сделать следующие выводы:

* текущие инструменты и модули для миграции данных имеют ряд недостатков, которые мешают каждой компании мигрировать данные на новую платформу;
* данная тематика и разработка модуля актуальна на текущий момент, т.к. большая часть клиентов стремится к переходу на новую платформу;
* наиболее продуктивной разработка видится как модуль к платформе ELMA365, а не ELMA3, в связи с её устаревшей архитектурой.

1. Проектирование подходов решения

В данном разделе представлен процесс проектирования и анализа архитектур обоих платформ для понимания и поиска наиболее оптимальных путей решения. Анализ платформ, приведённый в данном разделе, необходим для проектирования дальнейшего решения, описанного в разделе программной реализации.

2.1 Исследование архитектуры ELMA3

Работа системы ELMA 3/4 предполагает наличие необходимой и достаточной серверной инфраструктуры на стороне заказчика.Развертывание системы возможно в двух вариантах: Single Instance и Web Farm.

Single Instance предполагает, что для работы системы достаточно одного сервера приложений. Он является безальтернативным вариантом для конфигураций ниже Enterprise. Данную конфигурацию используют небольшие компании, для работы которых хватает мощности одного сервера. На рисунке 1 представлена схема Single Instance.

Web Farm предназначен для более крупных компаний, которым необходима мощность более 1 сервера и наличие резервного сервера для обеспечения надежности. Данный вид компоновки предполагает, что для работы системы недостаточно одного сервера приложений и требуется кластеризация серверов для повышения отказоустойчивости при работе системы. Доступен только в конфигурации Enterprise, наряду с Single Instance. Схема Web Farm представлена на рисунке 2.

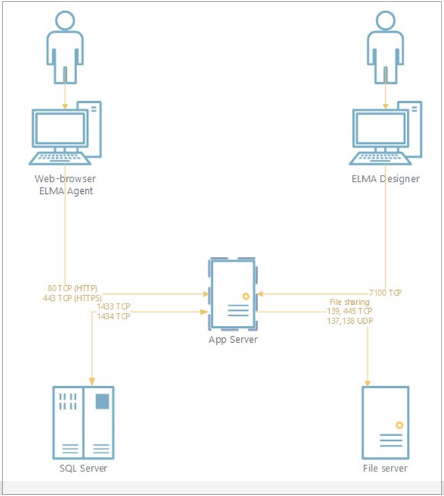


Рисунок 1 - Схема Single Instance

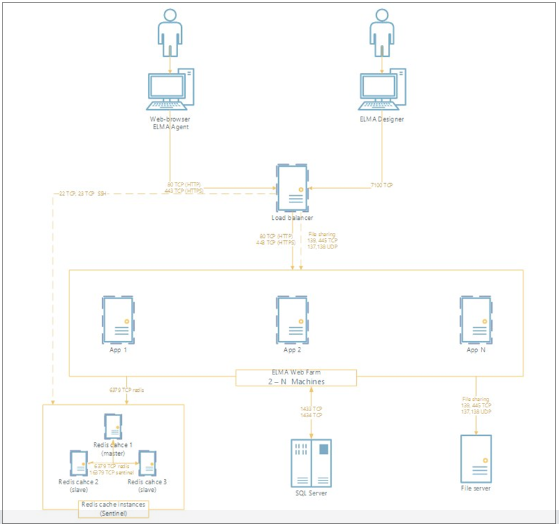


Рисунок 2 – Схема Web Farm

Безусловно, основными компонентами системы являются сервер приложений и сервер базы данных, однако есть множество дополнительных компонентов системы, которые влияют на её работу. На рисунке 3 представлена схема взаимодействия компонентов системы.

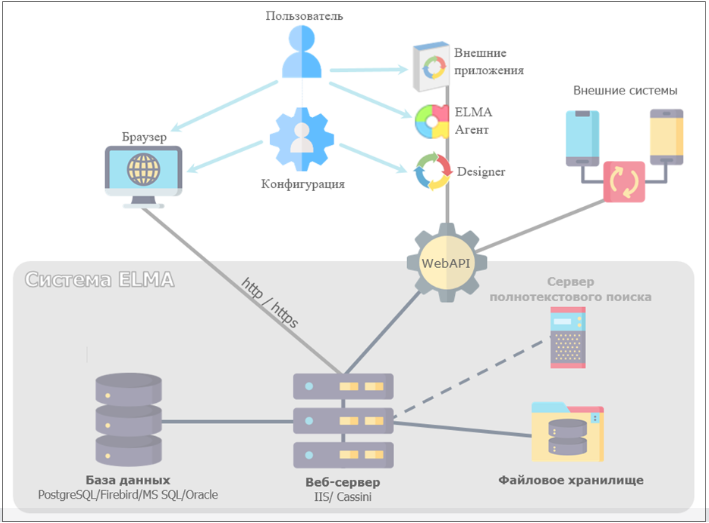


Рисунок 3 – Схема взаимодействия компонентов системы

По умолчанию, компонент в ELMA3/4 состоит из двух модулей:

Серверная часть;

Веб часть.

Сервеная часть представляет собой подключаемую к приложению библиотеку и содержит бизнес-модель и связанную с ней логику.

Веб часть представляет собой полноценное ASP .NET MVC 4.0 приложение, которое полностью реализует паттерн MVC в своей реализации архитектуры.

Структура проекта подобного приложения состоит из следующих рекомендованных директорий:

App\_Data – Место хранения данных для приложения. Это могут быть как простейшие базы данных, так и XML файлы и прочие ресурсы;

Content – Рекомендуемое место для хранения статичного содержимого приложения (стили, изображения и т.п.);

Scripts – Рекомендуемое место для хранения скриптов;

Controllers – место для хранения классов контроллеров приложения. Каждый контроллер должен заканчиваться словом «Controller» так требует Фреймворк MVC;

Models – Место для хранения классов моделей с которыми будет работать приложение и взаимодействовать контроллеры и представление;

Views – Место расположения представлений. Для Razor разметки используются файлы с расширением «.cshtml».

Файлы представлений являются шаблоном HTML страницы. Для преобразования шаблона страницы в готовую для передачи HTML разметку ASP.NET MVC использует движки представлений, такие как WebForms или Razor. Так же можно подключить другие движки. В ELMA используется Razor.

Объектная модель системы – это набор классов, сущностей и перечислений, которые хранят данные конфигурации и используется в бизнес-логике. Под сущностью понимается объект способный хранить данные в базе данных.

Для описание объектной модели в системе ELMA BPM используются метаданные. Метаданные – это специализированные данные, предназначенные для описания данных объектной модели. Для их создания используется Дизайнер ELMA или плагин к Visual Studio, позволяющий визуально описывать объектную модель системы.

В файловой системе все метаданные хранятся в виде файлов с расширением *.md*, которые в действительности представляют собой xml-файлы. На основе метаданных с использованием механизма кодогенерации генерируется код C#, который описывает объект, хранящийся в базе данных. На рисунке 5 представлена оболочка редактора сущностей, в которой видны свойства, задаваемые при создании какой-либо сущности.

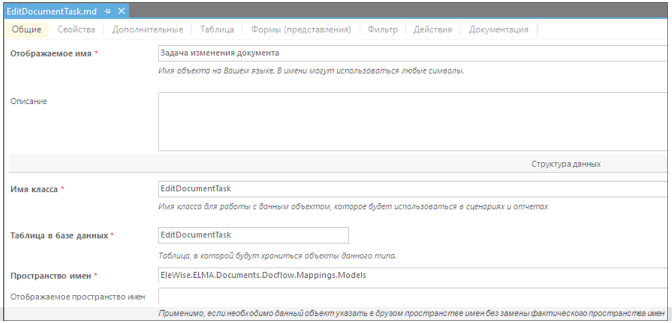


Рисунок 5 – Информация при создании сущности

На данной вкладке вводится общая информация о создаваемой сущности. Приведены следующие поля:

Отображаемое имя – используется для отображения названия типа в списке объектов. Данное поле является локализуемым;

Описание – описание типа создаваемой сущности. Поле является локализуемым;

Имя класса – название класса, описывающего тип создаваемой сущности;

Таблица в базе данных – название таблицы в базе данных, в которую будут сохранятся экземпляры создаваемой сущности;

Пространство имен – название пространства имен, в котором будет расположен создаваемый тип сущности;

Отображаемое пространство имен – используется для указания пространства имен.

В базе данных объектная модель представляет собой множество взаимосвязанных таблиц, соответствующих классам сущностей системы.

Важно учитывать определённую особенность архитектуры данных, которая называется «Наследование». Некоторые объекты системы имеют общего «родителя», который реализует общую для «потомков» составляющую: атрибуты, методы, логику работы.

Помимо этой особенности системы есть ещё одна: промежуточные таблицы связей. Данные таблицы, как правило, состоят из двух столбцов: Parent и Child и служат для организации хранения данных со связью N-N. Parent хранит Id основного объекта, а Child – Id связанных объектов.

Для наглядности будет рассмотрен пример связей сущностей при помощи MS SQL Server Manager Studio. На рисунке 7 приведен пример связи между двумя таблицами. Его трактовка с точки зрения БД будет пониматься как «Таблица Absence связана с таблицей GrounsForAbsence через атрибут Grounds, в котором хранится Id записи таблицы GrounsForAbsence.()хранение реляционное а обработка объектаян.

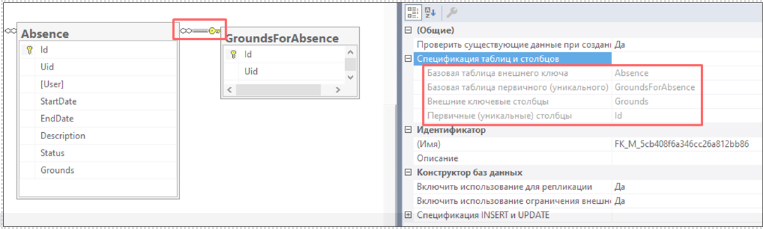


Рисунок 7 – Описание связи между таблицами

С точки зрения ELMA 3/4 эта информация обозначает, что объект с классом Absence (Отсутствие пользователя) содержит атрибут Grounds (Основание для отсутствия на рабочем месте) с типом атрибута GroundsForAbsence (Основание для отсутствия на рабочем месте) и одиночной связью.

2.2 Исследование архитектуры ELMA365

1. Техническое задание

В данном разделе представлено техническое задание на разработку модуля миграции данных для BPM платформ с разными архитектурами.

3.1 Краткая характеристика области применения

Модуль предназначен для клиентов компании ELMA, которые ранее использовали для работы монолитную платформу ELMA3 и решили перейти на новую платформу с микросервисной архитектурой ELMA365.

3.2 Назначение разработки

Функциональным назначением модуля является предоставление возможности клиентам, пользовавшимся платформой ELMA3, перенести данные с площадки своей компании на новую площадку, развернутую на основе микросервисной и более современной платформы ELMA365.

Модуль должен эксплуатироваться на ПК клиентов, вне зависимости от наличия решения на платформе ELMA3. Модуль доступен для загрузки как на облачную платформу ELMA365, так и на персональную, развернутую на сервере клиента. Особые требования к конечному пользователю не предъявляются.

3.3 Требования к модулю

Модуль должен обладать следующими функциональными возможностями:

1. отображение структуры сущностей платформы ELMA3;
2. отображение структуры сущностей платформы ELMA365;
3. возможность вручную соотносить свойства сущностей двух платформ с соответствующей валидацией;
4. атоматическое соотношение свойств сущностей на основе регулярных выражений;
5. сохранение текущей сессии экспорта данных в БД платформы;
6. подсчет поотраченного на экспорт времени;
7. шкала прогресса процесса импорта данных;
8. подсчет количества ошибок, возникших в ходе импорта данных;

3.4 Требования к надёжности

Надёжное (устойчивое) выполнение программы должно быть обеспечено выполнением пользователем совокупности организационно-технических мероприятий, перечень которых приведён ниже:

1. организацией бесперебойного питания технических средств;
2. использованием лицензионного программного обеспечения.

Отказы модуля возможны вследствие некорректных действий пользователя при взаимодействии с бразуером или вмешательством в код страницы другими пользователями. Во избежание возникновения отказов программы по указанным выше причинам следует обеспечить работу конечного пользователя без предоставления ему возможности вмешиваться в код страницы.

3.5 Требования к составу и параметрам технических средств

В состав технических средств должен входить IBM-совместимый персональный компьютер, включающий в себя:

1. дисплей с разрешением не меньше, чем 1024x768;
2. не менее 4-х гигабайт оперативной памяти;
3. клавиатура, мышь;

Системные программные средства, используемые программой, должны быть представлены следующими операционными системами:

* 64-разрядная OC Windows 7 или более поздней версии;
* Mac OS X 10.10 или более поздняя версия;
* Linux ARM 64 bit.

3.6 Требования к интерфейсу

Модуль должен обеспечивать взаимодействие с пользователем посредством графического пользовательского интерфейса и предоставлять возможность осуществлять ввод данных путем использования физической клавиатуры.

Предполагается использовать организацию интерфейса, выполненную в стилистике платформы ELMA365, в характерных для платформы цветах. Это необходимо для предотвращения ощущения вырванности из контекста модуля.

Также необходимо обеспечить поддержку экранов с высоким разрешением.

3.7 Требования к программной документации

Состав программной документации долженн включать в себя:

1. техническое задание;
2. руководство пользователя;
3. техническую документацию;
4. исходны код.
5. Разработка структуры модуля

В данном разделе в соответствии с требованиями, поставленными в техническом задание, описана структура модуля, алгоритмы функционирования, а также определен механизм валидации атрибутов сущностей для обеих платформ для избегания конфликтных ситуаций.

4.1 Разработка этапов миграции данных

3 этапа тут привести. Тут схемы алгоритмов надо и макеты меню

4.2 Алгоритмы функционирования для миграции данных

Тут надо алгоритм ля подтягивания сущностей. И их фомирования

4.3 Регулярные выражения для маппинга сущностей

Тут реглярки надо привести и т.д

5 Программная реализация

В данном разделе описаны выбранные инструменты для реализации модуля (язык программирования, библиотеки), разработан графический интерфейс пользователя, а также предоставлены детали реализации и визуализации реализации применённых ранее алгоритмов.

5.1 Выбор инструментов разработки

Т.к. было принято решение реализовать миграцию данных для BPM платформ с разными архитектурами путем создания модуля, то и языки программирования необходимо было выбирать из числа тех, которые поддерживаюттся платформой. Помимо вышесказанного, необходимо реализовать пользовательский интерфейс, который можно, при возможности, наращивать и добавлять новый функционал без особого изменения остальной структуры кода.

Среди доступных языков реализации логики имеются:

* JavaScript. JavaScript является широко используемым языком программирования, который поддерживается практически всеми современными браузерами. Он идеально подходит для разработки пользовательского интерфейса (UI) и взаимодействия с веб-страницами. JavaScript обеспечивает динамическое изменение содержимого страницы, обработку событий и взаимодействие с сервером.
* TypeScript. TypeScript является надмножеством JavaScript, которое добавляет статическую типизацию и другие возможности, улучшающие разработку и поддержку кода. Он компилируется в JavaScript и обладает расширенными возможностями для структурирования и поддержки проекта. TypeScript позволяет обнаруживать ошибки на этапе разработки и упрощает работу с большими кодовыми базами.
* C#. C# является языком программирования, разработанным Microsoft, и часто используется для создания приложений на платформе .NET. Он обладает широким набором инструментов и библиотек для разработки мощных и масштабируемых приложений. C# подходит для реализации бизнес-логики и взаимодействия с базой данных.
* Golang. Golang, также известный как Go, является языком программирования, разработанным Google, с фокусом на производительность, простоту и масштабируемость. Он подходит для создания эффективных и надежных системных приложений. Golang имеет нативную поддержку параллельных вычислений и хорошую производительность.
* C++. C++ - это язык программирования общего назначения, который обеспечивает высокую производительность и низкоуровневый доступ к системным ресурсам. Он широко используется для разработки системного и высокопроизводительного программного обеспечения. C++ предлагает богатые возможности, но может быть более сложным в освоении и требовать более тщательного управления памятью.

Рассмотрев основные языки программирования, приведенные выше, был проведен анализ и учтены требования к функциональности, производительности и доступности инструментов разработки конкретного языка.

Первым требованием к реализации является реализация модуля, а не отдельно взятой программы по ряду причин:

* Разделение функциональности. Модуль позволяет разделить функциональность проекта на отдельные части. Это способствует лучшей организации и структурированию кода, делает его более понятным и легко поддерживаемым. Каждый модуль может быть ответственным за определенную задачу или функцию, что упрощает разработку и обслуживание проекта.
* Модульность и повторное использование. Модульность позволяет создавать независимые модули, которые можно повторно использовать в разных проектах или внедрять в другие модули. Это сокращает время разработки и способствует повышению производительности. Учитывая архитектуру платформы – модуль можно внедрить на площадку любого клиента без особых усилий.
* Гибкость и масштабируемость. Модульный подход обеспечивает гибкость и масштабируемость проекта. Модули могут быть легко добавлены или удалены, а новый функционал может быть внедрен без влияния на другие модули или функции проекта. Это позволяет проекту быть более адаптивным к изменениям требований и обеспечивает более простое добавление нового функционала.
* Тестирование и отладка. Модульный подход облегчает тестирование и отладку проекта. Модули могут быть независимо тестированы, что облегчает обнаружение и исправление ошибок. Также модули могут быть заменены заглушками или имитациями для более эффективного тестирования.
* Расширяемость. Модули могут быть легко расширены для добавления нового функционала или взаимодействия с другими системами или модулями. Это позволяет проекту быть более гибким и способным адаптироваться к новым требованиям.
* Нативный доступ к данным. Благодаря прямому подключению модуля к платформе, данные берутся напрямую, не через запрос к серверу, что значительно ускоряет процесс обработки информации и экономит значительное количество времени.

В целом, модульный подход позволяет создавать более организованные, гибкие и расширяемые проекты.

На следующем этапе необходимо было выбрать родительскую платформу для модуля: ELMA3 или ELMA365. Обе эти платформы имеют свои точки расширения благодаря модульной структуре. Был проведен анализ возможностей платформ и выбран наиболее оптимальный вариант.

В виду наличия значительных преимущест было принято решение реализовывать модуль для платформы ELMA365. В числе преимуществ модуля для ELMA365 над ELMA3 следующее:

* Гибкая настройка модуля. Платформа ELMA365 имеет гибкую настройку модуля: возможность написания методов API, создание виджетов на языке JavaScript, настройка точек расширения, создание BPMn –процессов, для настройки бизнес-логики. Пример интерфейса настроек модуля на платформе ELMA365 представленаа на рисунке 11

5.2 Реализация этапов миграции данных

5.2.1 Получениие доступа к площадке для экспорта

Тут про рест апи и ввод логина и пароля надо. Что токен получаем и тд.

5.2.2 Получение структуры элементов платформы

Тоже самое. Что направляем запросы апи на платформу элма3 и тд. Потом получаем массив объектов и парсим их

5.2.3 Маппинг сущностей платформы

Тут про регулярные выражения, на каком языке они написаны и т.д. про автоматический и ручной режим сказать. Интрефейс на html сделан. Сущности всех разделов и приложений.

5.2.4 Создания элементов в ELMA365

Ну тут про то, что используетяс среда разработки платформенная, на TS и тд. Что использую методы JS, библиотеку math и тд. Динамическое формирование элементов с enum. И т.д. create – метод платформы, запись объекта в БД.

5.3 Тестирование и отладка модуля

Тут про белый ящик и про алгоритм тестирования: что проверяем по id со старой платформы: все ли они есть и т.д.

Заключение