a) Няма нищо, което да се допълни или промени в програмата ☐

b) (Не мога да преценя) ☐

c) (----------) ☐

5. По мое мнение раздадените писмени материали бяха:

a) на много високо ниво и достатъчни ☐

b) добре подготвени и достатъчни ☐

c) незадоволителни ☐

6. Общото ми мнение за организацията на обучението:

a) на много високо ниво ☐

b) задоволително ☐

c) незадоволително ☐

Заличаванията в документите са на основание Чл. 4 от Регламент (ЕС) 2016/679

7. Коя част от обучението беше най-интересна и коя част – най-безинтересна?

Еднакво интересни са и двете части – първата обуславя прилагането на втората на практика
Темата като цяло беше интересна
Интересна – примери
Тестване на системата
Тестване на механизъм за оценяване на рисковия потенциал
Като цяло обучението беше много интересно
Всички

8. Моля да дадете Вашите идеи и препоръки във връзка с организиране на подобни обучения в бъдеще:

Не мога да преценя
Нямам препоръки
Много добра организация. Нямам препоръки
Нямам предложения
Нямам

Бележки:

ОЦЕНКА НА ОБУЧИТЕЛИТЕ

Име на обучителя	Презентация	*Оценка*			
		Незадоволителна	*Добра*	*Много добра*	*Отлична*
	Съдържание				
	Представяне				
	Практическа важност за Вашата работа				

8.2.Подход за реализация на софтуерното решение

Софтуерната система ще отговаря на следните нефункционални изисквания:

- Ще бъде разработена на базата на съвременна SOA архитектура (архитектура, базирана на услуги). Приложението трябва да предоставя функционалностите си като SOAP и/или REST уеб услуги с цел лесна интеграция с външни системи.
- Ще бъде разработена изключително чрез използване на технологии и компоненти, спазващи утвърдени международни стандарти и практики.
- Системна конфигурируемост - системните функционалности ще могат да бъдат конфигурирани от администратор чрез използване на конфигурационни файлове или потребителски интерфейс. Системните параметри не трябва да бъдат закодирани в кода на приложението.
- Приложението ще бъде уеб-базирано и ще работи на всички популярни интернет-браузъри - като минимум ще се поддържат текущи версии на IE (Edge), Firefox и Chrome.
- Потребителският интерфейс на приложението ще бъде разработен според утвърдените стандарти и добри практики за достъпност и ползваемост.
- Всички технически документи като тези описващи системният дизайн и архитектура ще бъдат написани на български език.
- Изпълнителят ще изработи софтуера с технологични средства и подходи и на софтуерен език, които намери за най-подходящи, но които да отговарят на съвременните стандарти в ЕС и Република България, и тенденции в разработването и внедряването на подобни системи в други страни.

Системата ще поддържа следните видове и брой потребители:

- Потребители, които могат да извършват експертен мултидименсионен и релационен анализ, включително да извършват анализ с помощта на MS Excel, да изготвят и публикуват шаблонни отчети и информационни табла (дашборди) - 15 броя;
- Потребители, които създават и администрират функционалността за извличане преобразуване въвеждане на данни в хранилището - 2 броя;
- Потребители, които въвеждат данни и прикачат документи в пикселен формат – не по малко от 500 – общини, изпълнители на договори с общини;
- Потребители, които консумират изготвените отчети и анализи през портала. Неограничен брой.

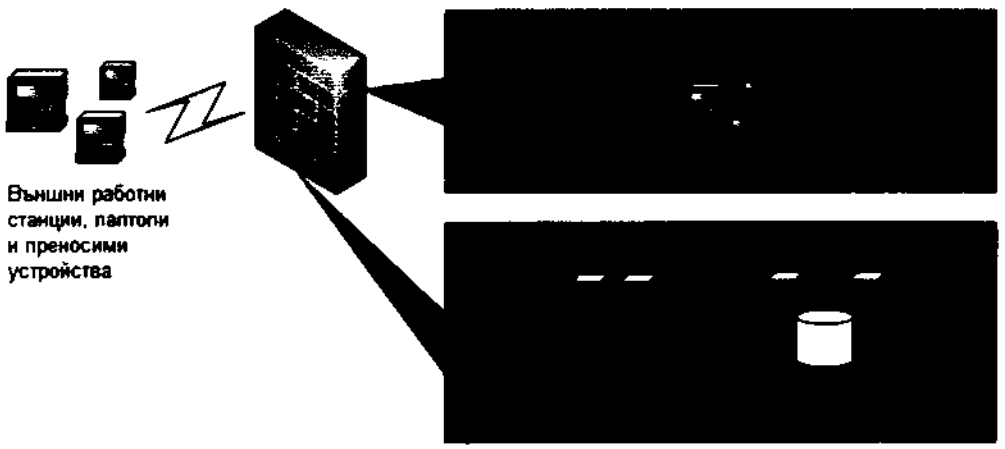
За да изпълни горепосочените изисквания екипът на Изпълнителя предлага да се реализира интегрирана информационна система, която да комбинира в себе си възможностите за обработка на транзакции по въвеждане на информационните обекти свързани с процеса на обработка на отпадъците в онлайн режим (OLTP), с функционалност за агрегиране и аналитична обработка на данните, гарантирани от използването и на хранилище на данните (data warehouse), което черпи информация от релационния модел на оперативната част на системата.
Системата ще се реализира чрез използването на двата модела на работа – от една страна стандартна транзакционна логика с нормализиран релационен модел на данните за оперативната част на системата, осигуряваща безпроблемно въвеждане на данни от различните източници – плоски файлове, потребителски интерфейс и електронен обмен чрез интеграционния модул на системата, а същевременно дименсионален модел на хранилище на данни, оптимизиран с цел предварително изчисляване на агрегирани стойности и максимално бързото им извличане за представяне в аналитични справки. Свързващото звено между тези две области на системата ще бъде ETL процеса, който ще зарежда данните съхранени в нормализирания релационен модел, ще ги обработва, изчиства, агрегира и трансформира, за да зареди дименсионалния модел на хранилището от данни.
Хранилищата на данни, които играят роля на стандартизирани точки на събиране, изчистване и консолидиране на данните, следва да бъдат изграждани с ясното съзнание, че тяхната функция и жизнен цикъл ще бъде тясно свързана с развитието на системите – източници, както и с нуждата от добавяне на нови източници на данни. Така архитектурата на всеки нов проект за изграждане на хранилище за данни следва да дава приоритет на ефикасността и гъвкавостта на реализацията, така че да може лесно да се интегрират идентифицирани източници на данни, да се използват разширяващите за възможности и нови методи за събиране на данни, както и да се отговори на нарастващата нужда от средства за анализ на натрупаните данни.

За целта архитектурата следва да е фокусирана не просто в събирането и изчистване на данни, но и в осигуряване на нужния инструментариум за статистическа обработка на тези данни, както и на възможности за управление на информацията от статистическите индикатори и връзките между тях. Реализацията на подобна архитектура изисква централизиран подход при събирането и обработка на данни, реализация на модел на

данни, който осигурява максимално преизползване на наличните данни, както и поддържане на каталог от мета данни за наличните данни в хранилището, който да позволява ясно, стандартизирано и гъвкаво конфигуриране на начина на използване и управление на хранилището.

8.2.1. Физическа архитектура

Предлаганото решение ще се базира на гъвкава архитектура, която ще може да се разгръща както в реализираната в рамките на проекта физическа среда, така и в реализирана в бъдеща виртуализирана среда, като държавния хибриден частен облак или друго подобно решение.



Фигура 20 - Физическа архитектура

От гледна точка на мрежовата организация в системата ще бъдат обособени следните зони с цел осигуряване на високо ниво на сигурност:

- External Users Zone – външна зона, която включва компютрите и мобилните устройства на външните потребители на приложението. Компютрите/ мобилните приложения имат достъп само и единствено до reverse-proxy сървъра, позиционирани в DMZ зоната за сигурност след преминаване на контрола на външната защитна стена и при спазване на софтуерните изисквания за сигурност на достъпа (HTTPS/SSL комуникация).
- Internal Users Zone – вътрешна зона, в която са позиционирани вътрешните потребители. Въпреки това работните станции имат достъп отново само до reverse-proxy сървъра, позиционирани в DMZ зоната за сигурност също след преминаване на контрола на външната защитна стена и при спазване на същите софтуерните изисквания за сигурност на достъпа (HTTPS/SSL комуникация).

- DMZ зона за сигурност – зона в която е позициониран reverse-proxy сървъра на системата. Целта от отделянето на този сървър в отделна зона е да се маскират вътрешните сървъри на системата, така че ако по някаква причина атакуващ получи достъп зад защитната стена, да не получи достъп до основните сървъри във вътрешната зона за сигурност.
- Secure Zone – вътрешна зона за сигурност, където са позиционирани, защитени от втора вътрешна защитна стена, уеб/приложния сървър и сървъра на базата данни на Системата и останалите сървъри на системите.

8.1.1.7. Предлагани параметри на виртуален сървър за хостинг на реализацията

Изпълнителят ще осигури виртуален сървър с доказана отказоустойчивост на който ще бъдат хоствани компонентите на системата. Той ще има следните параметри:

Операционна система	Оперативна памет	Виртуални процесори	Дисково пространство SSD
Linux CenOS	48GB	12	960GB

8.2.2. Логическа архитектура

Предлаганото решение цели да реализира софтуерно решение състоящо се от няколко основни компонента:

- Интерфейс за получаване на данни – компонентите реализирани с цел извличане на получените данни от отделни източници данни в системата. Реализиран под формата на SOAP базирана електронна услуга, и REST базиран програмен интерфейс (API), позволяващи на системата да се интегрира лесно с всякакви външни системи.
- Система за управление на документи – компонент реализиран под формата на подсистема състояща се от модул за приемане и съхраняване на архив от

електронни документи (под формата на файлове и структурирани данни) на сурови данни зареждани в модела на данни на системата.

- Област за предварителна обработка на данни (staging area) – нормализирана релационна база данни в която се зареждат получените данни за захранване на хранилището, които да бъдат обработени от процеса за трансформация на данните.

- Процес за трансформация на данните и зареждане на хранилището (Extract Transform Load process) – функционален механизъм за валидация, изчистване, трансформация и зареждане на данните в хранилището за данни.

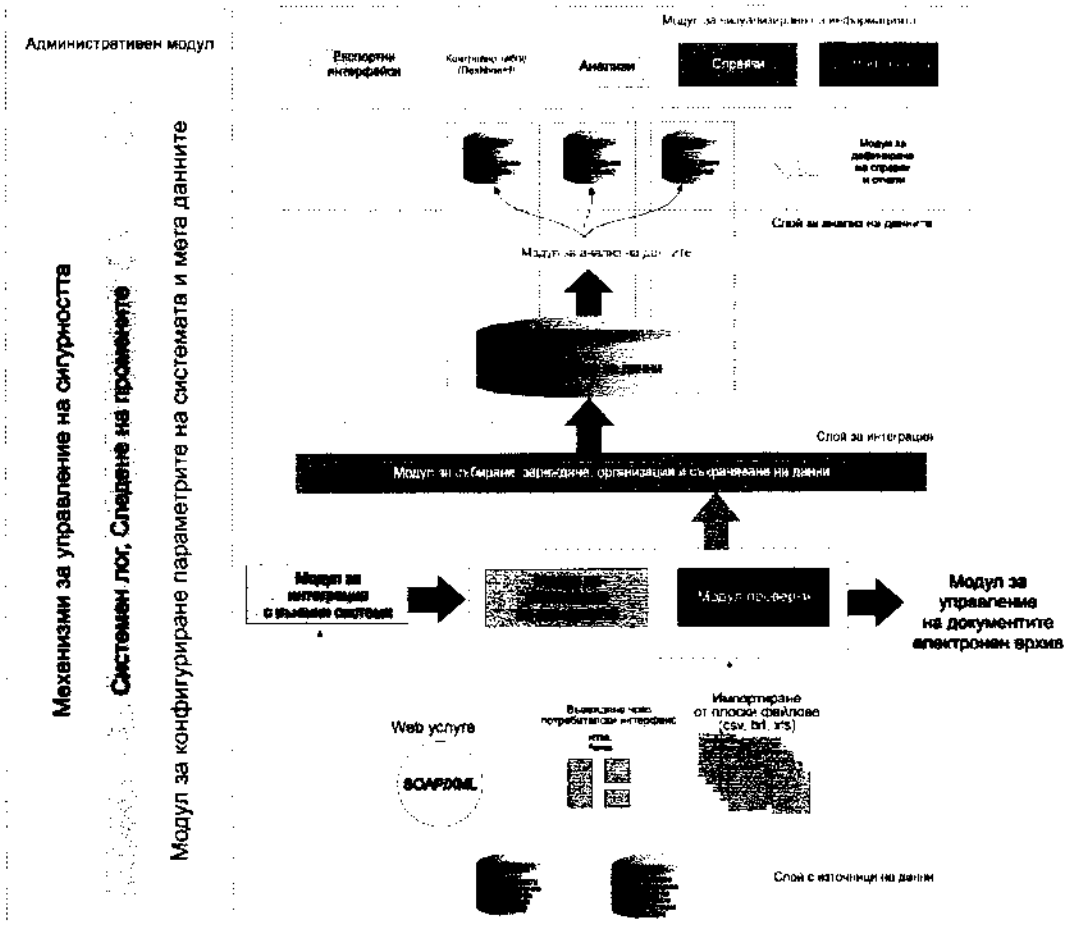
- Хранилище за данни (DWH) – релационна база данни с дименсионален денормализиран модел на данните ориентиран към предоставяне на аналитична информация, а не към обработка на транзакционна логика, реализирана като комбинация от факт таблици, съдържащи детайлната информация заредена от ETL процеса, както и отделни складове за данни (data mart) обслужващи извличането на аналитична информация от даден разрез от модела на хранилището на данни.

- Сървър за поддържане на кубове с аерирана информация (OLAP sube server) – компонент от приложението осигуряващ възможността за поддържане на кубове от предварително изчислени и заредени в паметта многомерни масиви данни, както и интерфейс, под формата на уеб услуга, позволяващ изпълнение на заявки към кубовете от данни и получаване на резултата от тях в структуриран вид (XML).

- Уеб приложение за представяне на обработената информация – уеб приложение, позволяващо на потребителите да изпълняват аналитични справки срещу данните в хранилището, да извличат информация за анализ, да дефинират мета данни за справките и администрират мета данните и параметрите за конфигурация на системата.

Взаимодействието на компонентите на реализацията ще се осъществи врез

използване на архитектура ориентирана към услугите и може да бъде илюстрирано със следната диаграма:



Фигура: Логическа архитектура на системата и компоненти на реализацията

8.2.3. Компоненти на реализацията

На логическо ниво отделните компоненти изграждащи приложението можем да обособим според спецификата им в следните основни категории:

- **Нормализирана (оперативна) и дименсионална (хранилище) бази данни** – Оперативната база данни е източник на натрупване на данни за зареждане на дименсионалната денормализирана база данни на хранилището. Избраният подход е реализация на база на релационна база данни PostgreSQL и за двете бази поради богатите функционални възможности на тази система за управление на база данни както по отношение на

транзакционни системи така и по отношение на хранилища за данни. Оперативната система ще играе роля на т.нар. staging area, където ще се извлича информация от различните фунцкионалности за регистриране на данни в системата.

- **ETL средства** – избраното ETL средство – Kettle, ще обработва събраните в оперативната база данни, ще ги валидира, трансформира и зареди в хранилището за данни

- **Мета данни** – мета данните ще бъдат съхранявани също в базата данни на хранилището, но следва да бъдат разглеждани като самостоятелен компонент на системата, защото концепцията е за максималното им използване във всички механизми. Мета данните са допълнителен набор от данни в базата, но подхода към съхраняването и управлението им е напълно отделен от зареждането на основните данни на хранилището. По тази причина мета данните ще бъдат обособени в отделен компонент – хранилище за мета данни.

- **Приложни средства за достъп до функционалността** – приложенията за представяне на аналитичната функционалност, включващо и административния модул на системата, ще бъдат обособени в отделен компонент, изградени със сходен подход и доколкото е възможно реализирани със сходна архитектура. Тук ще бъде реализира набор от функционалност за анализ на данни и работа с отчети, която да изпълни в пълна степен изискванията на заданието и да позволи пълноценно използване на хранилището.

- **Складове за данни (data marts)** – всеки от дефинираните складове за данни ще бъде реализиран в зависимост от резултатите от спецификацията на изискванията и проектирането на модела на данни на хранилището. След завършване на този етап ще бъде определено дали съответния склад ще бъде практически имплементиран като куб в OLAP сървъра, като отделна база данни с разрез от информацията на хранилището или като част от модела на данни на хранилището, под формата на набор от таблици с предварително

агрегирани данни. При който и да е от подходите ще бъдат следвани принципите на дизайна „star shema", в комбинация с описания подход към моделирането и проектирането и на база на опита на екипа от разработка на сходни проекти, ще бъде избрана най-подходящата форма за конкретна имплементация.

- **Приложни средства за управление, мониторинг и бекъп** – като отделен процес на приложния сървър ще бъде добавен и пълнофункционален модул, който да осигури възможност за управление и наблюдение на работата на компонентите на системата от единна точка, изпълняването на регулярни задачи, включително на архивиране на данните от базата данни, както и нотификация към администраторите на системата и събиране на статистическа информция за бързодействието и натоварването, която да се анализира с цел осигуряване на стабилност и бързодействие при натрупване на данни.

- **Интерфейс за разпространение на информацията** – в съответствие с изискванията за използване на архитектура ориентирана към услугите, интерфейсите за експорт, абониране и представяне на данни ще бъдат реализирани като SOAP и REST базирани електронни услуги, с формално дефиниран контракт от методи за заявяване на експорт на данни и стандартизирана схема за извеждане на данните.

8.2.4. Подход за реализация на ETL процеса

Предлаганото решение в настоящото предложение е да се използва технологичната платформа на Pentaho Data Integration, която включва в себе си пълнофункционален ETL сървър, както и графична среда, в която се моделира процеса на зареждане на данни. За целта ETL реализацията бива декомпозирана на съставни елементи, които биват записани под формата на мета данни. След моделиране на цялостния процес, той е готов за изпълнение от сървъра, като тази реализация може да бъде преконфигурирана и разширяване, чрез редактиране и допълване на въпросните мета данни.

Възможността за промяна в начина на работа на ETL процеса без директно програмиране на реализацията му под формата на писане на програмен код или SQL

заявки е основното предимство и причина за използване на ETL Middleware в настоящият проект. Наличието на визуалната среда Spoon за графично конфигуриране на мета данните за ETL процеса надгражда подходящо инфраструктурата предоставяна от Kettle сървъра, а и ще позволи по-лесен трансфер на знания към екипа на Възложителя, така че да може след изграждане на хранилището да се постигне лесно и удобно надграждане на реализирания процес.

Предлаганото решение в настоящото предложение е да се използва безплатната open source версия на технологичната платформа на Pentaho Data Integration – Kettle Data Integration, която включва в себе си пълнофункционален ETL сървър, както и графична среда, в която се моделира процеса на зареждане на данни. За целта ETL реализацията бива декомпозирана на съставни елементи, които биват записани под формата на мета данни. След моделиране на цялостния процес, той е готов за изпълнение от сървъра, като тази реализация може да бъде преконфигурирана и разширяване, чрез редактиране и допълване на въпросните мета данни.

Възможността за промяна в начина на работа на ETL процеса без директно програмиране на реализацията му под формата на писане на програмен код или SQL

заявки е основното предимство и причина за използване на ETL Middleware в настоящият проект. Наличието на визуалната среда Spoon за графично конфигуриране на мета данните за ETL процеса надгражда подходящо инфраструктурата предоставяна от Kettle сървъра, а и ще позволи по-лесен трансфер на знания към екипа на Възложителя, така че да може след изграждане на хранилището да се постигне лесно и удобно надграждане на реализирания процес.

8.2.4.1. Аргументация на подхода за реализация на ETL процеса

При избора на методиката за реализация на ETL процеса следва да се избере една от многото алтернативни възможности за реализация на процеса: алтернативи по отношение на използваните средства, по отношение на структурата на процеса, по отношение на моделирането на мета данните.

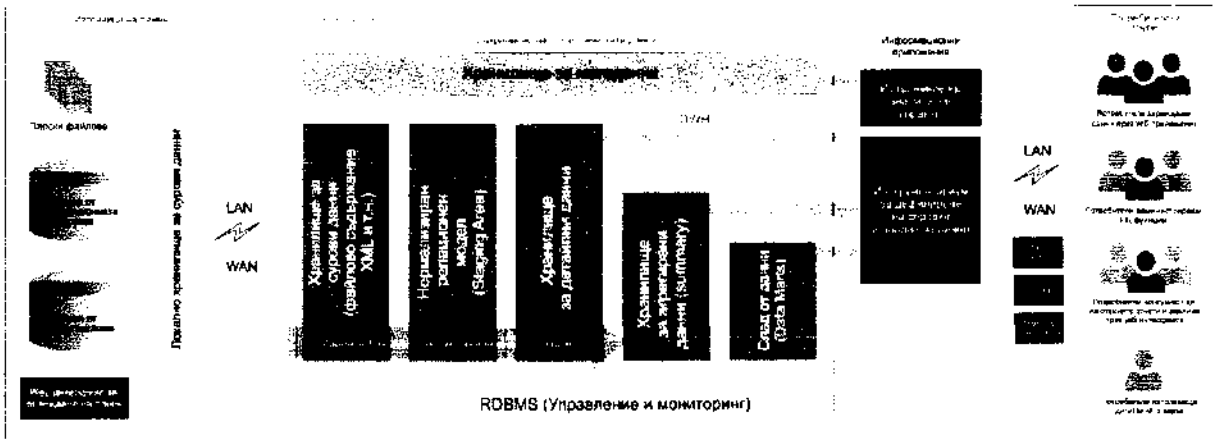
Първия избор при започване на реализацията на ETL процеса се отнася до това дали да се реализира отделно приложение, в което да се програмира логиката за изпълнение

на процеса, да се използва готов генератор с подходяща базова функционалност, да се реализира процеса изцяло със средствата на СУБД, чрез съхранени SQL процедури, или реализация чрез готов продукт за реализация на ETL процеси. Поради изискванията за максимална гъвкавост и възможността за по-ефективен трансфер на знания към екипа на Възложителя, подходящият избор е използване на готов продукт за реализация на ETL – в случая продукта Kettle.

Втория избор при реализацията на ETL процеса е в избора на средство за извършване на трансформацията на данните – при подхода ETL, трансформацията се извършва преди зареждането в базата на хранилището (extract -> transform -> load), при алтернативния подход се използват средствата на самата база данни, така че суровите данни се зареждат в хранилището и се трансформират на място в него (extract -> load -> transform). Използването на ETL сървър, нужно за дефинирането на параметризиран и конфигуриран процес предопределят избора на модел на реализация ETL.

Допълнителни технологични решения включени в предлагания подход са използването на област за съхраняване на сурови (оригинални) данни, както и наличието на staging database. От гледна точка на пълната проследимост на процеса (audit trail), от гледна точка на по-сигурното управление на оригиналните данни (поддържането им в контролирана от системата област осигурява по-добър контрол от ограничаването до съхраняване на пътища до файловата система), както и от гледна точка на по-доброто разграничаване на етапите от процеса (с наличието на staging база се отделя процеса на извличане от процеса на самата трансформация), използването на тези два допълнителни компонента ще доведе до по-стабилна и устойчива архитектура.

8.2.4.2. Предлаган процес на реализация на ETL



Фигура: Архитектура на ETL процеса

Предлагания процес за реализация на ETL в системата включва следните основни стъпки описани на горната диаграма:

1. Извличане на данните от източниците – концепцията на системата приема, че ще се създаде оперативна подсистема, която ще регистрира и управлява въведените данни в нормализирания релационен модел на системата (комуникационна и функционална), като събраните данни ще бъдат представяни на подсистемата на хранилището чрез електронна услуга, за да се спазват принципите на архитектурата ориентирана към услугите и ще се запази максимална гъвкавост и независимост на компонентите.

2. След получаване чрез електронна услуга, данните биват съхранени в отделен компонент – система за управление на документи. Това представлява централизирано хранилище за XML данни, плоски файлове, както и всякакви други формати, които могат да служат за зареждане на хранилището. В случай на междусистемна комуникация в хранилището за сурови данни се съхраняват съответните обменени съобщения, в случай на SOAP или REST комуникация.

3. Съхранените сурови данни биват извлечени от хранилището за сурови данни, в рамките на първоначалната стъпка на ETL процеса – извличането на данни. Оттам се преминава към извличане на данни за трансформацията, което се реализира на базата на мета данни за хранилището, в които се описва откъде в модела на staging базата се зарежда всеки тип данни, който след трансформация ще попадне в модела на данни за хранилището (т.нар.

Logical Data Map).

4. Стъпка на анализиране на данните – в тази стъпка се извършват няколко основни операции свързани с осигуряване на оптимално като бързодействие и консистентност зареждане на данните. В тази стъпка се извършва валидация на данните, спрямо зададените в мета данните към модела на хранилището условия и ограничения, както и анализ на отклоненията в данните, по предварително зададени от администраторите на системата критерии.

5. Трансформиране на данните – при тази стъпка заредените в staging базата данни, съхранени в типичен релационен модел, ще бъдат заредени в модела за данни на хранилището, изграждайки съответните факт таблици и димeнсии. Впоследствие OLAP сървъра ще зареди в паметта нужните кубове от предварително агрегирани данни.

8.2.4.3.Инструменти за реализация на ETL процеса

За реалзиацията на ETL процеса предлагаме Kettle Data Integration, безплатната community версия на платформата Pentaho Data Integration, която съдържа нужния инструментариум за изграждане на сложни ETL процеси, със зареждане на данни от разнообразни източници, конфигурируеми трансформации на данните, както и възможност за визуално моделиране.
Продуктите от платформата Kettle Data Integration включват:

- Report Designer – средство за визуално създаване на справки и отчети за представяне на резултатите от тях
- Aggregation Designer – средство за визуално конфигуриране на трансформации при създаване на агрегирани данни
- Schema Workbench – средство за управление на схемата на хранилища от данни.
- Metadata Editor – редактор за мета данни по отношение на хранилището за данни.

В допълнение ще бъде използван и безплатен сървър за OLAP кубове с отворен код – Mondrian.
Изборът на софтуер с безплатно ползване и отворен код ще позволи лесно скалиране на системата при нужда от разширяване на нейния обхват и обем от данни, както и ще осигури че в дългосрочен план системата няма да доведе до лицензни разходи за Възложителя и след изтичане на 24 месечния гаранционен период на системата.

8.2.4.4. Осигуряване на качеството ЕТЛ процеса

Описания процес на зареждане на данните в хранилището преминава през всички нива на приложението – стъпките от процеса включват извличане на данни от staging базата, валидирането и трансформирането им и зареждането в модела на данни на хранилището.

Моделът на данните на логическо ниво, включително и в склада от данни (datamarts) ще позволява валидиране на данните, които трябва служат като източник за справки и отчети. Моделът на данните ще се описва със съответните мета данни, които ще позволят управление на структурата и валидността на данните от екипа на Възложителя чрез дефиниране на валидационни изисквания и ограничения, които да бъдат прилагани към данните.

ETL процесът ще бъде реализиран чрез средствата на Kettle, т.е. реализацията ще бъде композирана като процедури за извличане и трансформация с отделни стъпки и преходи, които биват конфигурирани също на база мета данните на приложението.

При тестването на ETL процедурите ще бъдат използвани следните видове тестове за осигуряване на качеството на процеса за зареждане на данни:

- Функционално тестване: може би най-комплексната и критична фаза на тестване, поради факта, че тя се отразява директно върху качеството на данните.

- Тест на елементите (Unit test): тестове от типа „white-box“, които всеки разработчик прилага на елементите, които е разработил. Те позволяват да се намали сложността на тестване, и да се създаде възможност за създаване на по-подробни доклади за напредъка на проекта.

- Интеграционен тест: тестове от типа „black-box" , които позволяват да бъде проверена коректността на потока от данни в ETL процедурите.

- Тестове от типа „Принудителна грешка“: разработени, за да поставят принудително ETL процедурите в условия на грешки, насочени към проверка, дали системата може да се справи с дефектни данни, както е планирано по време на анализа на изискванията.

- Тъй като ETL е силно код-базиран, повечето стандартни метрики за генерично тестване на софтуерна система могат да се използват и тук.

8.2.5. Подход към реализация на хранилището от данни

В модерните тенденции за разработката на хранилища на данни все по-често се утвърждава подхода към поддържане на обособени складове на данни като част от хранилищата, за сметка на традиционния подход, при който справките и извличането на данни се изпълняват директно срещу схемата на хранилището. Подхода с използване на складове за данни(data marts) се превръща в широко разпространена алтернатива на хранилищата за данни (data warehouses). С придобиването на популярност на складовете за данни, както и с разработката на все повече проекти при използване на този подход, самата концепция се развива, като се обособяват поне три различни модела или информационни модела за реализация на складове за данни.

Първият подход при изграждане на складовете за данни, се базира на това че те могат да бъдат най-добре определени като „подмножество“ или „разрез“ (често до известна степен или напълно агрегирани данни от факт таблиците на основния модел на хранилището). Въпросният агрегиран набор от данни може да бъде инсталиран на отделна, относително евтина компютърна платформа, след което периодично да бъде обновяван от централното хранилище за данни. Погледнато така складовете за данни могат да бъдат разглеждани, като производни или прилежащи елементи на хранилищата за данни.

Вторият модел за реализация отрича използването на основното хранилище за данни като източник на информация и разглежда складовете за данни като независимо изградени, директно от източниците на информация, които служат за зареждането както на основното хранилище за данни, така и на складовете за данни. При този подход за зареждане на складовете за данни се използват техниките за организация и инструментите използвани в ETL процеса на самото хранилище за данни. Складовете за данни структурно са подобни на хранилищата за данни, като могат да бъдат представени като малки хранилища за данни със специфична бизнес функция. Разглежданите по този начин складове за данни, прерастват съвсем естествено в хранилища за данни.

Третият модел за развитие има за цел да синтезира и премахне несъответствията които естествено се развиват при подхода на независимо изграждане на складовете от модела на хранилището, прилаган в предишните два модела. При третия подход складовете за данни се изграждат успоредно с хранилищата за данни. Така модела на данни е зависим от наличните данни от източниците, но складовете за данни не се налага да „изчакват" интегрирането на въпросните данни в модела на хранилищата за данни. Всички складове за данни се ръководят от основния модел на хранилището за данни. При спазване на този принцип, склада от данни може да бъде завършен преди реализацията на основния модел, след което интегриран след изграждането на корпоративното хранилище за данни.

И при трите посочени модела на развитие на складовете за данни не е застъпен конкретен подход към интегрирането на промени наложени от потребителската обратна връзка в процеса на изграждане. Всеки от подходите разгледани до тук третира връзката между хранилището за данни и склада за данни като относително статична.

Тук следва да се отбележи и връзката с двата алтернативни подхода, които са разглеждат като възможности за реализация на модела на основното хранилище:

- Модела „От горе на долу" (The top down model)

Хранилището за данни се изгражда на базата на предварително проведени подробни анализи на данните, които ще трябва да бъдат обработвани, чрез приложение на ETT процесите. Хранилището за данни интегрира всички данни в общ формат и обща софтуерна среда. Всички ресурсни данни на компанията са обединени в структурата на хранилището за данни. Всичките необходими данни нужни за аналитичната обработка са налични в хранилището за данни. Едно от основните предимства на този модел е че след като веднъж хранилището за данни е имплементирано, не е необходимо допълнително обединяване на данни. Данните трябва да бъдат само предоставени на потребителите в разбираем и удобен за тях формат.

- Модел на изграждане „От долу на горе"(Bottom - up)

При този модел не се извършват предварителни анализи на необходимите обработки на подаваните данни, складовете за данни се съставят на базата на зададените

източници на данни. При този модел складовете за данни са независимо разработени и внедрени, поради което не са свързани едни с други спрямо използвания проект за разработка. При разрастването на изградени по този модел складове за данни често са на лице дублиране или пропуски в информацията. Така изградените складове са слабо или тотално дезинтегрирани, което в дългосрочен план означава, че проблемите с качеството и консистентността на данните се пренасят от данните в информационните системи източници към складовете на хранилището, като по този начин се налага ново ниво на консолидация, вече в хранилището на данни.

- Модел „Паралелна разработка“

Все по-популярен модел на разработка, който ограничава обособеността на складовете с цел запазване на цялостния интегритет на модела. На първо място, изграждането на складовете от данни следва да бъде съобразявано с реализацията на модела на основното хранилище, който отразява общия изглед над всички данни в хранилището. От друга страна се оставя независимост при разработката на складовете, като при откриване на дублиране или недостиг на данни в отделен склад, въпросното несъответствие с общия модел се документира и се планират мерки за отстраняване. При този процес опита от разработката на отделни складове може да се имплементира при разработка на следващите складове и служи за подобряване на цялостния модел.

8.2.5.1.Аргументация на подхода към реализация на основния модел на хранилището на данни

Използването на моделите на изграждане отдолу нагоре, както и на паралелния модел е по-подходящо при проекти, в които има ниска степен на дефинираност и липса на цялостен поглед върху наличните данни. Често такава ситуация се среща в големи организации със силно хетерогенна среда от системи, сложна организационна структура и разнообразни бизнес процеси. При тези организации е възможно никой да няма точна представа за качеството на данните и възможността за консолидация до момента да разработка на складовете. В такава ситуация следва да се приеме някой от двата посочени подхода, за да се върви напред с малки стъпки и да се консолидира постепенно натрупания опит на по-късен етап във все по-обхватен модел.

От друга страна настоящият проект има ясен обхват, предхождащите го дейности

дават ценна и относително пълна информация за вида и източниците на данни, които ще се използват. Дефинираните цели пред проекта са предварително определени. При наличната основа от информация, налична за анализ, както и при кратките срокове за реализация, смятаме, че подхода на изграждане „отгоре - надолу" с обратна връзка (top – down with feedback), ще позволи да се създаде бързо основен модел на база на документираните изисквания, след което да се получи обратна връзка в рамките на пилотното използване на системата, така че да се прецизира и разшири системата преди окончателното внедряване в продуктивна среда.

8.2.5.2.Интегриране на обратната връзка в предлагания подход за реализация.

Макар от гледна точка на предварителния анализ да има висока степен на определеност на целите и обхвата от данни, който трябва да бъде включен в хранилището, очакването при пилотното използване на проекта е да се открият нови данни и дори нови източници на данни. В този първи етап на интегриране на обратна връзка от потребителите следва да се анализира дали новото изискване надхвърля обхвата на модела на данните на хранилището. Втория етап е интегрирането на промените в модела на хранилището и на съответния склад нужната нова информация. Така се създава един процес на постоянно надграждане и подобряване на основния модел на хранилището, с цел удовлетворяване на нуждите на потребителите. Разбира се при този подход имаме събиране на обратна връзка от периферията (отделните складове), към центъра (основната схема на хранилището), което означава, че няма как всички изисквания да бъдат удовлетворени и то бързо, но все пак се постига постоянно подобрение.

8.2.6. Основни функционални модули на реализацията

8.2.6.1.Модул „Управление на отпадъци"

Модулът ще предлага функционалност, където всеки отпадък, от видовете изброени по-долу, трябва да може да бъде зачислен към определен център и да се проследява неговото движение до окончателното му третиране.

Този модул трябва ще бъде обектно ориентиран и в него да се обособят 3 обекта,

всеки от които се образува от предходните обекти, като съдържа техните данни, както и данни за действията със самия обект.

Обектите, които трябва да обхваща системата са следните:

- Отпадъците намиращи се в даден общински център по видове и количества описани по-долу, във всеки момент;
- Отпадъците, по видове и количества описани по-долу, натоварени на превозно средство с приемо-предавателен протокол, за транспортиране извън всеки общински център;
- Отпадъци по видове и количества описани по-долу, които са предадени в съоръжение за третиране /оползотворяване или обезвреждане/, както и приложимата дейност по третиране за всеки вид отпадък (R1 до R13 или D1 до D15;

- Отпадъци по видове и количества описани по-долу, които са окончателно третирани /оползотворени или обезвредени/, приложимата дейност по третиране за всеки вид отпадък (от R1 до R11 и/или от D1 до D-15).

В модулът ще могат да бъдат въведени към съответните обекти и пренесени в следващите обекти, следните данни:

- запис на всеки предаден битов отпадък в системата с уникален код на записа, дата, данни за лицето, което предава отпадък – имена, чрез мобилен център ли е събран отпадъкът или пряко лицето го е предало, наименование и код на предадения отпадък съгл. Наредба №2 от 23.07.2014г. за класификация нса отпадъците, по-конкретно:

Вид на отпадъка	Код и наименование по Наредба № 2 за класификация на отпадъците	
Лаково бояджийски материали и покрития:		
1. Бои	20 01 27*	бои, мастила, лепила/адхезиви и смоли, съдържащи опасни вещества
2. Лакове	20 01 27*	бои, мастила, лепила/адхезиви и смоли, съдържащи опасни вещества
3. Разтворители	20 01 13*	Разтворители
4. Грундове	20 01 27*	бои, мастила, лепила/адхезиви и смоли, съдържащи опасни вещества
5. Лепила	20 01 27*	бои, мастила, лепила/адхезиви и смоли, съдържащи опасни вещества
6. Смоли	20 01 27*	бои, мастила, лепила/адхезиви и смоли, съдържащи опасни вещества
7. Мастила	20 01 27*	бои, мастила, лепила/адхезиви и смоли, съдържащи опасни вещества
Домакински препарати и химикали:		

8. Перилни и почистващи препарати (препарати за почистване на стъкла, фурни, белина, препарати отстраняващи петна и ръжда, почистващи повърхности, дезинфектанти)	20 01 29*	перилни и почистващи смеси, съдържащи опасни вещества
9. Киселини	20 01 14*	Киселини
10. Основи	20 01 15*	Основи
11. Препарати за растителна защита и борба с вредителите (препарати за поддържане на тревни площи, цветя, овощни дървета, зеленчукови растения - пестициди, хербициди)	20 01 19*	Пестициди
12. Фотографски материали	20 01 17*	фотографски химични вещества и смеси
13. Спирачни течности	16 01 13*	спирачни течности
14. Антифризни течности;	16 01 14*	антифризни течности, съдържащи опасни вещества
Фармацевтични продукти:		
15. Лекарства с изтекъл срок на годност;	20 01 31*	цитотоксични и цитостатични лекарствени продукти
16. Продукти, свързани с грижи за домашни животни	20 01 31*	цитотоксични и цитостатични лекарствени продукти
Живак и живаксъдържащи отпадъци:		
17. Живак, живачни термометри, живачни прекъсвачи, живачни ампули от бойлери и др.	20 01 21*	луминесцентни тръби и други отпадъци, съдържащи живак
Кърпи за изтриване и предпазни средства, замърсени с опасни вещества:		
18. Кърпи, парцали за избърсване, замърсени с опасни препарати	15 02 02*	абсорбенти, филтърни материали (включително маслени филтри, неупоменати другаде), кърпи за изтриване, предпазни облекла, замърсени с опасни вещества
19. Предпазни средства – ръкавици, маски, филтри и др., използвани при боядисване, нанасянето на	15 02 02*	абсорбенти, филтърни материали (включително маслени филтри, неупоменати другаде), кърпи за изтриване, предпазни облекла, замърсени с опасни вещества

покрития и почистване.		
Замърсени дървесни материали		
20. Замърсени дървесни материали	20 01 37*	дървесина, съдържаща опасни вещества
Масово-разпространени битови отпадъци		
21. Негодни за употреба батерии и акумулатори	20 01 33*	Батерии и акумулатори, включени в 16 06 01, 16 06 02 или 16 06 03, както и несортирани батерии и акумулатори, съдържащи такива батерии
22. Излязло от употреба електрическо и електронно оборудване	20 01 35*	Излязло от употреба електрическо и електронно оборудване, различно от упоменатото в 20 01 21 и 20 01 23, съдържащо опасни компоненти
23. Хладилна и климатична техника с охлаждащ агент съдържащ хлорофлуоро-въглеводороди	20 01 23*	Излязло от употреба оборудване, съдържащо хлорофлуоровъгле-водороди
24. Технически масла (не хранителни масла и мазнини)	20 01 26*	Масла и мазнини, различни от упоменатите в 20 01 25

- *за всеки вид отпадък, ще се предвиди основно и допълнително поле за статистическия код на отпадъка, както и поле дали се прилага към дадения отпадък принципът "Разширена отговорност на производителя" (т.е. лицата, пускащи на пазара продукти, след употребата на които се образуват масово разпространени отпадъци, отговарят за разделното им събиране и третиране).*
- наименование и идентификационен, трицифрен номер на общинския пилотен център, свързан с уникалия код на всеки запис за всеки предаден отпадък, GPS координати на центъра, както и трицифрен код за местонахождение на центъра по класификатора на териториалните единици за статистически цели в България;
- дата на всяко натоварване за транспортиране извън общинския пилотен център, свързана с определено количество и вид отпадък, от вид изброен по-горе;

- регистрационен номер на превозното средство, с което се транспортира определено количество и вид отпадък, от вид изброен по-горе, поле за наименование и Поле за ЕИК на транспортната фирма;

- опаковка, общо нетно тегло, на транспортирания вид отпадък, от вид изброен по-горе, свързан с уникалния запис на всеки входиран битов отпадък за периода от последното транспортиране на такъв вид отпадък извън центъра – при всяко транспортиране извън общинския център;

- фирма на оператора/преработвателя, при когото ще бъде транспортиран отпадъкът, Поле за ЕИК, полета за отбелязване на разрешените на преработвателя, Поле за въвеждане на дейности по оползотворяване ("R1 до R13") или обезвреждане ("D1 до D15"), които преработвателят може да извършва и максимални позволени количества, които той може да третира, дата до която са разрешени.

- действителен адрес на площадката или съоръжението на преработвателя вкл. държава, дали е площадка за временно съхранение или за третиране /обезвреждане ишли оползотворяване/ до която се транспортира отпадъкът от даденият вид, код, количество, опаковки, тегло;
- дата на примо-предаване на преработвателя, при когото ще бъде транспортиран отпадък, от вид изброен по-горе, свързан с уникалния запис на всеки приет в общинския център битов отпадък, за периода от последното транспортиране на дадения вид отпадък извън центъра;

- датата на Декларацията/Удостоверението от преработвателя, че даден отпадък е бил третиран – ако отпадъкът е третиран в България, или дата на обезвреждане или оползотворяване, съответстваща на тази в полетата на Приложение 1Б от Регламент (ЕС) N1013/2006 на Европейския Парламент и на Съвета от 14 юни 2006г. относно превозването на отпадъци, или датите в полетата на еквивалентният документ за движение по Базелската конвенция, когато третирането е извършено в държава извън ЕС – ако отпадъкът е третиран извън страната;
- наименование на всеки вид третиран отпадък, неговият код, общо нетно тегло, код на дейността по третиране (от R1 до R11 и/или от D1 до D-15).

На съответените етапи, изброени по-горе, ще се осигури възможност, за въвеждане на данни освен тези в приложенията към *Наредба № 1 от 04 юни 2014 г. за реда и образците, по които се предоставя информация за дейностите по отпадъците, както и реда за водене на публични регистри,* свързани с:

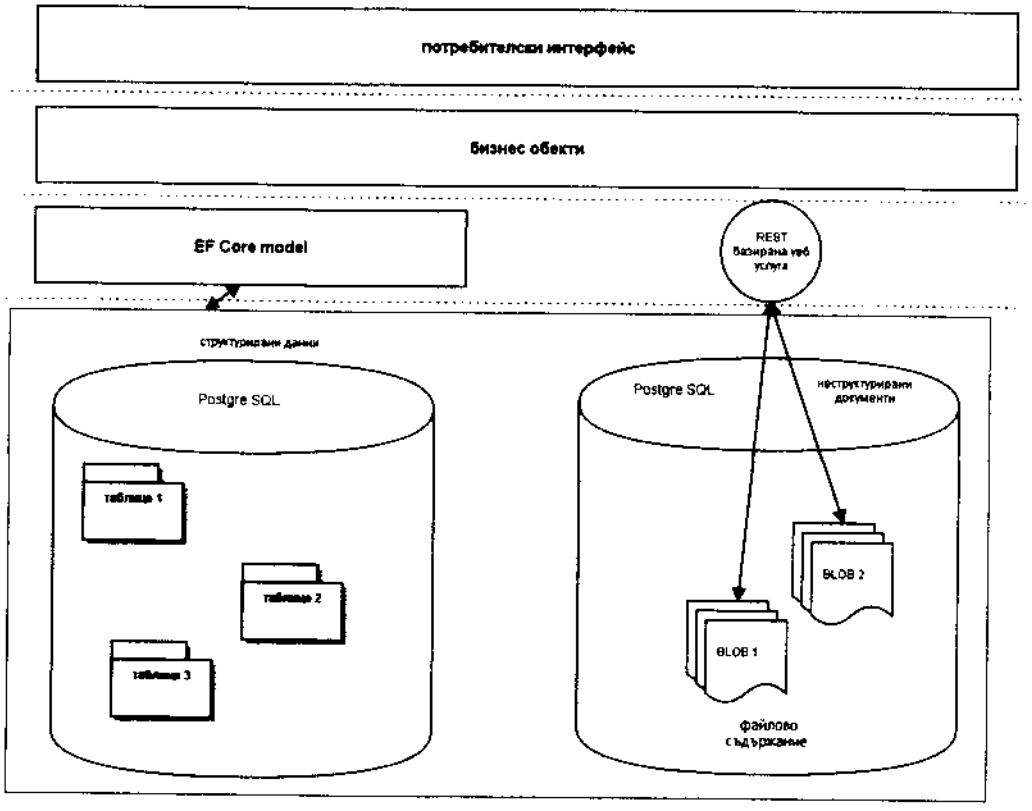
- Данни за Разрешение, комплексно разрешително или регистрационен документ по чл.35 от ЗУО;
- Данни за Идентификационен документ по *Наредба № 1 от 04 юни 2014 г. за реда и образците, по които се предоставя информация за дейностите по отпадъците, както и реда за водене на публични регистри*;
- Данни за одобрена от МОСВ нотификационна форма за износ на отпадъци;
- Данни за подписани от съоръжението за третиране документи за приемане на отпадък;
- Данни за извършено третиране от съоръжението за третиране.

8.2.6.2.Модул „Проверки“

Модулът ще осигури документално и информационно наблюдение и контрол на процеса по третиране на опасни отпадъци чрез извършване на проверки на площадки, налични отпадъци, тяхното движение, действията на изпълнителите. Софтуерният продукт ще позволява на оторизирани потребители да регистрират извършена проверка, както и да въвеждат снимков материал и сканирани документи, съотносими към определена площадка или изпълнител. Приложените файлове ще се съхраняват в модул за управление на документи.

8.2.6.3.Модул „Управление на документи“

Предлаганият от нас подход за реализиране изискванията за управление и поддържане на архив на получените в системата документи, независимо от техния формат (текстови файлове, CSV, XML, ексел файлове, PDF или други), ще бъде реализирано чрез използване на обособен модул за управление и съхранение на неструктурирани документи и файлово съдържание. Реализацията на модула напълно се базира на принципите на SOA, като той ще бъде достъпен като отделна REST базирана електронна услуга за всички останали модули на системата. Модулът е реализиран под формата на вътрешна за системата услуга, използваща вградените в PostgreSQL средства съхраняване и управление на бинарни данни и изградено API, имплементиращо специфична бизнес логика за нуждите на системата. Следната диаграма описва архитектурата на модула за управление на файлово съдържание:



Фигура - Архитектура на модула за управление на документи

Разделянето и съхранението на структурираните и неструктурираните данни се извършва по напълно прозрачен за крайния потребител начин – в основната база данни на системата се пазят не файлове, а връзки идентифициращи адреса на съдържанието в

модула за файлово съдържание. При извикване на файлово съдържание, потребителския интерфейс използва тази информация за да се обърне към електронната услуга на модула за файлово съдържание. Тази услуга реализира логика, която отговаря за локализирането на съдържанието, независимо от начина на разпределяне на подсистемата, който ще бъде определен във фазата на анализ и проектиране. Реализацията на модула под формата на REST услуга позволява разделяне на логиката за съхранение на данните от бизнес логиката и отговаря на добрите практики.

Основните предимства на предлаганото решение са:

- неструктурираните данни се съхраняват послредством стандартните средства на релационната база данни (BLOB тип данни), което дава възможност за включването им в единна система и схема за сигурност и контрол на достъпа
- предлаганата СУБД дава възможност за голяма скалируемост и и съхраняването на голям обем данни, което гарантира жизнеспособността на системата и съхраняването на необходимата информация в целият обозрим времеви хоризонт на експлоатация на системата
- реализирането на модула във вид на вътрешна уеб услуга изолира имплементацията на съхранението и четенето на данните от общата бизнес логика на системата и дава възможност в бъдеще при евентуална необходимост модулът да бъде видоизменян или заменян без това да води до промени и пренаписване на други части от системата.

8.2.6.4.Модул „Дигитална карта“

Като част от уеб приложението за достъп до имплементираните в хранилището справки, ще бъде реализирана и интеграция с функционалността на Google Maps API, JavaScript базиран интерфейс, за интерактивно представяне на данни в уеб приложения, без нужда от поддържане на пълномащабна GIS функционалност. По този начин предлаганото API бива „вграждано“ в уеб приложенията, като се използват готови карти, което дава допълнителни предимства на Възложителя, поради липсата на нужда от

поддържане и актуализиране на детайлна географска информация в базата данни. Модулът ще бъде тясно интегриран с модула за справки, така че визуализацията на картата да бъде обвързвана с наличните в хранилището справки и отчети по динамичен начин, а не чрез фиксирано въвеждане на стойности за визуализация.

8.2.6.5.Модул „Справки“

Изпълнителят ще внедри, пълнофункционален модул за генериране на справки, които да позволяват на потребители с достъп до съответните данни да анализират по разнообразни критерии обобщената информация от компонентите на системата.

Справочният модул ще предостави интерфейс на потребителите за задаване на критерии и представяне на резултатите от изпълнението на конкретни справки. Справочната система ще разполага със следната функционалност:

- Задаване на филтри по различни критерии преди извличане на резултатите;
- Задаване на начин на сортиране на резултатите;
- Представяне на резултатите от справка в табличен вид;
- Комбинирани справки, включващи резултати в табличен вид и диаграми;
- Указване на колоните, които да бъдат включени в справката и техните параметри;
- Възможност за отпечатване по предварително дефинирани образци;
- Справките ще могат да се експортират във файлове в стандартен формат (минимум xls);

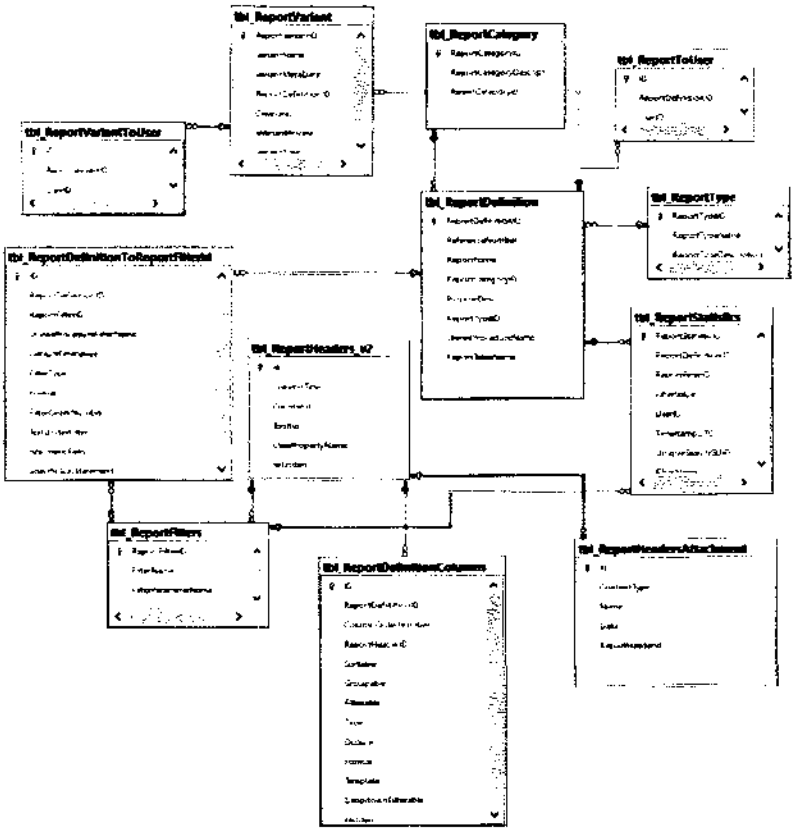
Справките ще могат да се визуализират по прегледен начин на екран, а също така да се съхраняват локално на компютър, минимум във формат на структурирана Excel таблица.

Модулът за справки ще комбинира няколко основни подкомпонента:

- Механизъм за извличане на справки от оперативната база данни;
- Механизъм за динамично дефиниране на справки, запазване на индивидуализирани изгледи към справките;
- Механизъм за представяне на информация от справките;
- Механизъм за съхраняване и/или експорт на резултатите от справките;
- Механизъм за управление на достъпа на потребители до справки.

Реализацията на модула за спрарки се базира на динамичен модел на данните описващи справките. Модула пази информация за всяка справка в таблица tbl_Report_Definition). Потребителите могат да съхраняват индивидуализираните варианти на справките в таблица tbl_Report_Variant. Индивидуализираните варианти на справката (шаблони) позволяват на потребител, който редовно използва дадена справка или отчет да запази в базата вариант на тази справка – конкретните филтри които е приложил, колоните които са избрани за визуализация, реда на подреждане на колоните,

какъвто потребителя използва в момента на съхраняването на варианта. За всяка дефиниция на справка и всеки вариант може да бъдат определяни потребители които могат да я изпълняват. Към дефиницията на справката е приложен и помощния обект tbl_sql, в него се съхранява компресирана информация за таблиците от които извлича данни заявката, на която се базира справката, колоните участващи в нея, фиксираните филтри заложени в справката, параметрите които потребителя има право да подаде при изпълнение на заявката, както и условието за свързване на избраните таблици.



Фигура 21 - Модел на данни на модул справки

Функционалността за извличане на справки ще използва възможностите за динамично изграждане на заявките в платформата .Net, така че да генерира оптимизирани заявки към оперативната база данни на системата, като на база на запазената в дефиницията структура на заявката и подадените чрез потребителския интерфейс параметри на справката се формира заявката за изпълнение, модула използва връзката си към съответната база данни, изпълнява заявката и извлича резултата, като при нужда той може да бъде съхраняван в системен кеш.

Функционалността за дефиниране на справки ще се базира на каталог на данните в модела на данни на оперативната система, и възможността на потребителя да формира заявки спрямо посочения каталог, които след това се съхраняват под формата на дефиниция на справка. Ще бъде осигурена възможност за конструиране на заявки в

потребителския интерфейс, с цел максимално лесно разширяване на набора от справки наличен в системата.

Модулът ще включва функционалност за:

- статистически справки с извличане на агрегирана информация от базата данни на системата;
  - чрез модула за търсене ще бъде осигурено търсене на информация по различни критерии, според вида на информационния обект, за който е справката.
  - чрез модула за търсене ще се осигури извличане на информация от оперативната база данни по различни критерии (поле от структурираните данни за съответния информационен обект) и генериране на оперативни справки
  - за генериране на оперативните и статистически справки ще бъде реализирана функция, която ще предостави възможности за:
- създаване на шаблон чрез дефиниране на структурата и съдържанието му;
- съхраняване на създадения шаблон за последващо многократно ползване;
- редактиране на вече създаден шаблон – добавяне или премахване на елементи структурата му и/или данни от съдържанието му;

Модулът за справки ще визуализира резултатите в интерфейсен елемент с богата функционалност за обработка на таблицата с резултатите на генерираните справки. Посоченият елемент наречен advanced grid, зарежда данните чрез javascript в динамична HTML таблица, като предлага цял ред от възможности за обработка:

- бързо и лесно сортиране по всяка колона, с възможности за определяне на варианти за третиране на числа като текст и обратно
- филтриране на информацията по избрани колони, като филтрите могат да бъдат текстови полета, дропдаун или мултиселект за случаите когато възможния набор от стойности в полето се свеждат до кратък списък, възможности за използване на логически критерии при филтрирането като например съдържа, не съдържа, сравнителни операции за числа и дати и тн.
- разместване на колони и категоризиране на цялата таблица на база колони
- скриване на колони от източника на данни за справката в интерфейса
- графично представяне на информацията под формата на диаграми от различен вид – колонни, линейни, кръгови, стълбовидни, съобразно вида на справката;

Чрез модула за справки ще бъдат осигурени възможности за:

- преглед чрез визуализация в уеб браузъра, разпечатване;
- експорт в различен файлов формат .docx, .xlsx, или .CSV на генерираните справки.

Системата ще извършва разнообразни справки по филтър от различни показатели:

- Отчети за лица, предаващи отпадъци, вид и количества на предадените опасни битови отпадъци. Този модул осигурява и възможност за автоматизирана връзка с други информационни системи от този тип (за управление на опасни отпадъци);

- Количество от даден вид опасни битови отпадци, събрани за определен период в определени общини, област или цялата страна;

- Динамика на движението на даден вид, отпадък – количество постъпило и количество транспортирано от всеки център за произволно зададен референтен период.

Минималните критерии които ще се изпълняват от справките са:
- Справка за моментни количества от всеки вид отпадък за даден център, както и количества от всеки вид отпадък, постъпили или напуснали центъра/пункта/площадка, през зададен от потребител референтен период;
- Справка по горецитираните критерии, но сумарно за всички центрове, от цялата страна, които са включени в системата към момента на исканата справка.

8.2.6.6.Административен модул

Основното предназначение на модула е да осигури възможност на оторизирани потребители да конфигурират и управляват цялостната работа на системата.

Достъп до административния модул ще имат само потребители с роля администратор, в чиито индивидуални права са включени съответните функции на системата.

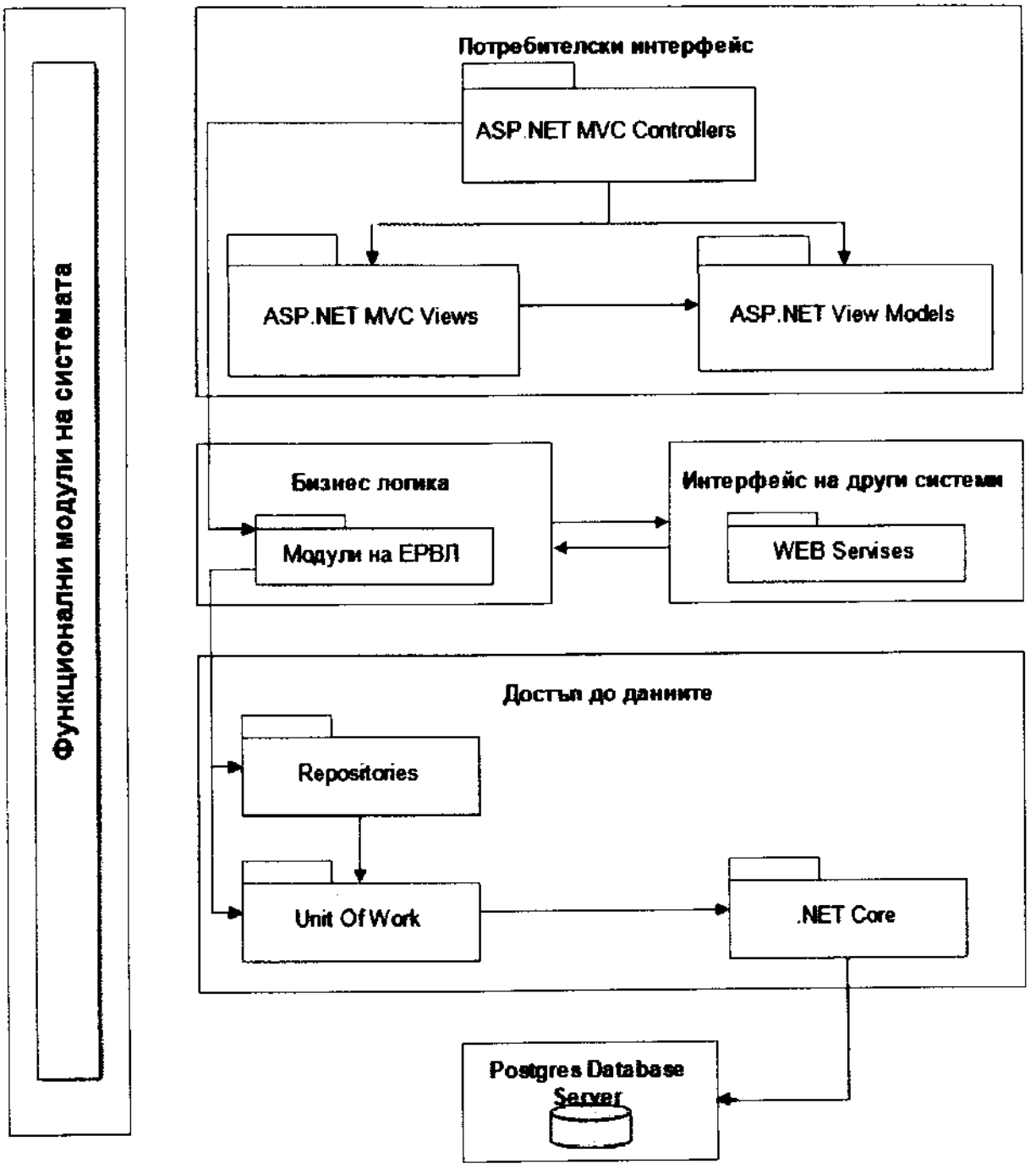
В обхвата на административния модула ще са включени функционалности, които са реализирани и се изпълняват чрез други модули на системата:

- промяна на информацията за потребителите на системата, т.е. добавяне, модификация, деактивиране или блокиране на потребителски профили – ще се извършва чрез модула за управление на потребители или групи;
- създаване на нови елементи в системните номенклатури и редактиране на съществуващи ще се извършва чрез модула за управление на номенклатури,;
- управлението на шаблони ще се извършва чрез функционалностите на модула за изготвяне на стандартизирани отчети;
- промени в параметрите за работата със системата;
- всички действия, извършени от потребителите във връзка с управление на номенклатури, шаблони, потребители и групи, конфигуриране на работата със системата ще се съхраняват чрез модула за следене и регистриране на извършените промени.

8.2.7. Подход за реализацията на уеб приложението за въвеждане на данни, представяне на аналитични справки и администриране на системата

Уеб приложението като компонент на системата ще представлява основната точка за достъп на потребители на системата, която позволява на потребителите интеракция с моделите на данни на оперативната база и базата на хранилището, поддържани от останалите компоненти на резализацията. Уеб приложението ще изпълнява следните основни функции:

- Ще предоставя достъп до функционалността за въвеждане на данни в системата
- Ще предоставя достъп до аналитичните справки и отчети реализирани в справочния модул на системата
- Ще предоставя достъп до административния модул на системата
- Ще предоставя достъп до процеса за мониторинг на системата и архивиране на данните



Фигура 22 Трислойната архитектура на реализацията на уеб приложението на системата – с прилагане на архитектура MVC

За да се постигнат изискванията за модулност, мащабируемост и гъвкавост уеб приложението ще бъде разделено на функционални модули, позволяващи модификация, допълване на нови модули или пълна подмяна на модули без необходимост от внасяне на изменения в останалите и в базисния софтуер на системата.

Отделните компоненти и модули реализиращи функциите на приложението ще бъдат проектирани на база на най-добрите, съвременни и перспективни технологични платформи и архитектури, съгласно принципите на обектно-ориентирания анализ и дизайн. Обектно-ориентирания анализ и дизайн е софтуерен инженерен подход, който моделира системата като група от взаимодействащи си обекти. Всеки обект представя

даден елемент от системата, която се моделира и характеризира със своето състояние и поведение. Проблемния домейн се декомпозира на отделни обекти следвайки принципите на дизайн от общото към частното и свързване на частите съобразно техните отговорности. Обектно-ориентирания анализ и дизайн по същество е процес на последователни действия на опростяване т.е. оперирайки с проблемния домейн в него се „инжектират" структури, които го декомпозират и опростяват. Структурите, които служат за декомпозиция и опростяване, представляват шаблони за дизайн, които са доказали своята ефективност при много различни ситуации и се препоръчват от водещите производители на софтуерни решения. Такъв шаблон за дизайн представлява декомпозицията на софтуерното решение на отделни слоеве комуникиращи помежду си по строго определени интерфейси. Основно предимство при тази архитектура е че позволява в отделните слоеве да бъдат извършвани значителни промени без това да оказва влияние на останалите, което води до изключителна гъвкавост. Слоевете са определени така, че да групират елементите, които искаме да можем да варираме независимо. Слоевете са определени така, че да групират елементите, които искаме да можем да варираме независимо. При уеб базираните приложения доказан подход е разделянето на следните слоеве:

- Слой на базата данни;
- Слой на бизнес логиката;
- Слой на потребителския интерфейс;
- Слой на интерфейса с външни системи.

Всеки слой в последствие се декомпозира на отделни модули, като комуникацията между модулите се осъществява също по строго специфицирани интерфейси.

✓ Слой за достъп до базата данни

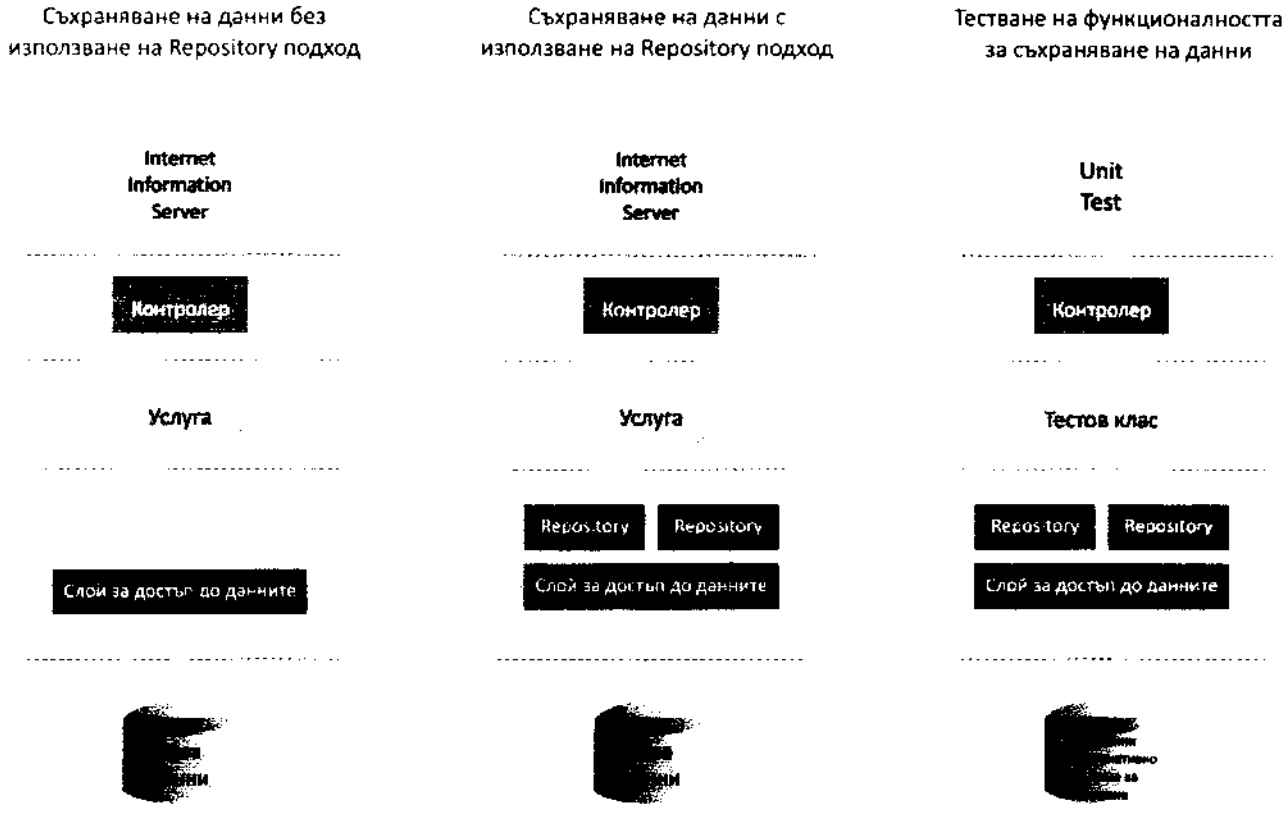
Задачата на слоя за достъп на базата данни е да обслужва и съхранява данните на информационната система. Уеб базираната система ще включва в себе си и система за управление на съдържанието, която ще бъде изградена при спазване на подхода за проектиране Repository design pattern, като функционалните механизми на системата няма да имат директен достъп за манипулиране на данните в базата данни, а ще използват реализираните в слоя за управление на данните методи. Класовете на бизнес логиката на модулите на системата от своя страна ще изпълнява подадените заявки за регистриране, промяна или извличане на информация БД, в съответствие с изискванията за разграничаване на достъпа, сигурност, както и осигуряване на консистентността на данните и неделимостта на транзакциите.
Слоя за управление на данните ще бъде реализиран на базата на механизма за поддържане на обектно- релационни съответствия в Microsoft .Net Framework - Entity Framework Core. Използването на подхода за автоматизирано поддържане на съответствията между информационни обекти и релационната база данни осигурява няколко основни предимства:

- Функционалността за съхраняване на данните става независимо от спецификата на съхраняващото базата данни СУБД, поради факта че Entity Framework работи с различни популярни СУБД;
- Промените в структурата на информационните обекти може да бъде автоматично

отразена в релационния модел;

- Допълнителния слой на абстракция позволява по голяма гъвкавост по отношение на физическия модел на данните – т.е. улеснява скалирането на базата данни и евентуалното му разпределяне в повече бази данни;
- Entity Framework Core осигурява пълна поддържка на транзакции на ниво на слоя за достъп до данните, така могат да се осигури изпълнението на заявката към базата данни на приложението, да отговарят на изискванията ;
    - Atomicity – атомарност на транзакцията (при грешка се отменят всички заявки от транзакцията, за да е успешна транзакцията са изпълнени всички заявки от нея);
    - Consistency – цялост на данните след всяка изпълнена транзакция;
    - Isolation – изолация на данните по отношение на други транзакции;
    - Durability – стабилност на транзакцията, липса на възможност за загуба на данни.

Основното предимство на избрания подход се изразява във възможността да не се влага логика в услугите използващи съдържанието на базата данни, тъй като слоя до данните му дава нужните методи и отделя със слой на абстракция по-ниските нива на работа с информационните обекти и базата данни. На следната схема са показани двете основни предимства пред традиционния подход, липсата на директно извикване на данните от бизнес логиката и възможността за автоматизирано модулно тестване на фунционалността на приложението, дори преди да е завършена реализацията на базата данни:



Фигура 23 Съхраняване на данни

Слоя за достъп до данните ще съхранява електронно съдържание в релационната база данни - Postgress. При проектирането и създаването на базата данни под внимание ще бъдат взети следните аспекти:

- Възможност за ефективна работа с големи обеми от данни;
- Възможност за осигуряване на ефективен механизъм за резервиране на данните;
- Високо ниво на сигурност.

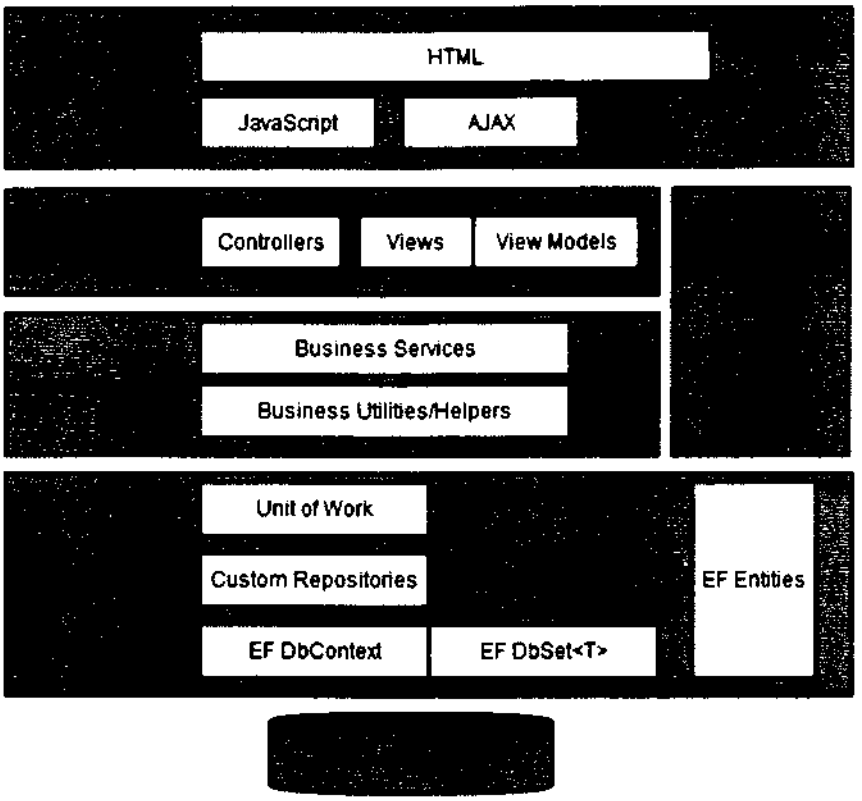
✓ **Слой на бизнес логиката**

Функцията на слоя на бизнес логиката е да обработва заявките за обработка получени от интерфейса на системата, да поддържа тяхната валидност (консистентност) спрямо идентифицираните бизнес процеси и правила и да осигури управлението на потока от данни, съобщения, и документи, свързани с работата на системата, като ги съхранява и извлича от слоя за достъп до данните. Според логическата архитектура на системата в слоя с бизнес логиката се разполагат всички модули, функции и процедури, които реализират функционалността на системата – **модулите за управление на съдържанието, обособени в система за управление на съдържанието, както и индивидуализираните модули за предоставяне на справки и представяне на информацията от изследванията**.

✓ **Слой на потребителския интерфейс**

Задачата на слоя на потребителският интерфейс е да взаимодейства с потребителите на системата, така че да обслужва техните заявки и да представя необходимата им информация в определения формат. В този слой ще се организира и използва интерфейса

на системата, от потребителите, за да извършват желаната от тях работа. Въведените данни в този слой се подават на слоя на бизнес логиката за последващата им обработка и съхранение в базата данни. Той ще бъде реализиран от шаблона за дизайн Model-View-Controller.
ASP.NET MVC контролерите, които посрещат заявките на потребителите ги предават за обработка към бизнес логиката, попълват необходимите данни във View Model-ите и връщат съответното view, което предоставя визуализация на потребителския интерфейс.
За реализирането на богат и удобен за работа потребителски интерфейс, който не изисква постоянно презареждане от сървъра view-то, което се рендира в браузъра ще бъде реализирано като се използва библиотеката за динамично уеб съдържание jQuery.
Слоя на потребителския интерфейс ще отговаря на всички изисквания на техническото задание към потребителския интерфейс.



Фигура 24 - Многослойна Архитектура

Прилагането на MVC парадигмата осигурява следните преимущества:

- **Преизползваемост на компонентите**. Разделението на модела от изгледа позволява компонентите на модела да бъдат преизползваеми за различни изгледи.
- **Скалируемост.** Моделът осигурява както хоризонтална, така и вертикална скалируемост. Позволява промяна на броя потребители, транзакции, обръщения към системата. Функционалността на системата може да бъде променена, без промени във всички слоеве
- **Гъвкавост.** Моделът осигурява лесната заменяемост на отделните слоеве като поддържа капсулируемост на отделните нива и следи да еднопосочност на

връзките между тях. Определя и входно – изходни точки между слоевете.

8.2.8. Технологична платформа за реализацията

За реализация на софтуерните разработки в настоящата обществена поръчка Изпълнителя ще използва платформата Microsoft .NET Framework. За език за програмиране ще бъде използван C#.

За среда за разработка ще бъде използвана средата Microsoft Visual Studio 2017.

Microsoft.NET Framework е популярна и съвременна платформа за реализация на бизнес приложения, която е специфицирана и създадена от Microsoft. Тя е базирана на отворени стандарти, които са общодостъпни и са публикувани на страницата на ECMA. Възможностите на платформата заедно с наличните средства за разработка предоставят мощно средство за разработка, което позволява изграждане на съвременни архитектурни решения. .NET Framework е пуснат за разпространение през 2002г. От създаването си до момента платформата постоянно се развива и подобрява за да се утвърди като водеща технологична платформа за разработка на приложен софтуер.

Предложената платформа отговаря на следните изисквания:

- Бързодействие при изпълнение – програмният код не се интерпретира, платформата разполага със средства за компилация в оптимизиран за бързодействие платформено-зависим машинен код;
- Сигурност – програмният код е изолиран от хардуерната и софтуерната среда, в която работи, като по този начин го предпазва от програмни грешки;
- Лесна инсталация – инсталирането на софтуерните приложения не изисква сложни настройки;
- Поддръжка на утвърдени в индустрията езици за програмиране – MS.NET не е обвързана с конкретен програмен език;
- Отворени стандарти – MS.NET е базирана на отворени стандарти;
- Поддръжка на средства за централизирано съхранение на бизнес-обектите в релационна база данни;
- Поддържа уеб услуги (web services);
- Скалируемост – MS.NET притежава възможност за паралелна работа на множество сървъри, изпълнение на разпределени транзакции и управление на натоварването;
- Средства за мониторинг – т MS.NET притежава средства за наблюдение на натоварването, използваните ресурси, възникналите изключения и др.

- Среда за разработка - MS.NET е интегрирана с предложената среда за разработка MS Visual Studio.

8.2.8.1.Платформена независимост

MS.NET Framework, предоставя библиотека от класове FCL (Framework Class Library) и среда за изпълнение CLR (Common Language Runtime). Библиотеката от класове (FCL) осигурява множество от готови класове и интерфейси, които спомагат за ускоряване и оптимизиране на процесите по разработка и развитие на приложенията. За разработка под .NET Framework се използват езици от високо ниво, като създадения програмен код се компилира до платформено-независим междинен език, наречен CIL (Common Intermediate Language) код. По време на изпълнение на приложението CIL кодът (т. нар. „управляван код") автоматично се компилира от CLR до изпълним код за конкретната хардуерна платформа и операционна система, на която работи приложението. С разработката на версията на .Net Framework Core се реализира пълнофункционална версия на технологичната платформа, която работи независимо от вида на операционната система. Използването на .Net Framework Core ще осигури възможност системата да се реализира на операционна система Linux, което от своя страна ще означава, че няма да се налагат разходи за лицензи за операционна система и приложен сървър.

8.2.8.2.Бързодействие при изпълнение

Платформата разполага със среда за изпълнение CRE (Common Runtime Engine), която позволява приложенията да бъдат компилирани в междинен CIL (Common Intermediate Language) код. При изпълнението си този междинен код не се интерпретиран както при другите виртуални машини напр. Java, вместо това се извършва JIT (Just In Time) компилация в платформено-зависим машинен код (native code).

8.2.8.3.Лесна инсталация

Програмите, създадени на .NET Framework, както и техните компоненти, могат да бъдат инсталирани с просто копиране в желаната директория - процес, известен като XCopy Deployment.

8.2.8.4.Отворени стандарти

Спецификациите на CLI (Common Language Infrastructure), както и езиците C# и C++/CIL са стандартизирани от организациите ECMA и ISO и са отворени стандарти.

8.2.8.5.Поддръжка на утвърдени в индустрията езици за програмиране

Платформата Microsoft.NET не е обвързана с програмни езици. Тя осигурява езикова независимост. Това е възможно благодарение на съвместимостта на типовете данни, които отделните езици поддържат. CTS (Common Type System) дефинира всички

базови типове данни, както и начинът, по който те могат да бъдат конвертирани един в друг. Тези типове са споделени между всички .NET езици и са стандартизирани в CIL. Към момента има създадени .NET версии на всички масово използвани езици за програмиране, в това число: C++, C#, Visual Basic и др.

8.2.8.6.Поддръжка на подсистеми за сигурност, базирани на утвърдени стандарти

Microsoft .NET Framework осигурява базови класове и среда за изпълнение (CLR), които са отговорни за поддържане на подсистемите за сигурност на платформата.

Платформата осигурява следните утвърдени стандартни подсистеми:

- Подсистема за управление на изпълнението на кода – Code Access Security;
- Подсистема за управление на достъпа на потребителите – Role Based Security.

Подсистемата за управление на изпълнението на кода определя нивото на сигурност, което е отредено да всяко парче код, което се изпълнява. Нивото на сигурност зависи от мястото на изпълнение, създателят на кода и от други фактори.

Подсистема за управление на достъпа на потребителите е базирана на потребителски роли, с помощта, на които за всеки потребител може да се определи дали има достъп до определен ресурс и дали ма право да извърши дадена операция. Управлението на потребителите с помощта на роли не зависи от начина по който .NET Framework идентифицира, автентикира и оторизира потребителите. Потребителите може да се автентикират и оторизират спрямо различни източници, в това число спрямо системата за сигурност на локален сървър, домейн или друг източник.

Платформата осигурява възможности за криптиране, генериране на криптографски ключове и хеширане на съобщения. Измежду поддържаните алгоритми са DES, SHA, AES, RC2 и други. Наред с криптографските възможности, платформата предоставя и средства за работа с цифрови сертификати.

Защитата на транспортно ниво може да бъде осигурена чрез осигуряване на защитени канали за комуникация през SSL или IPsec.

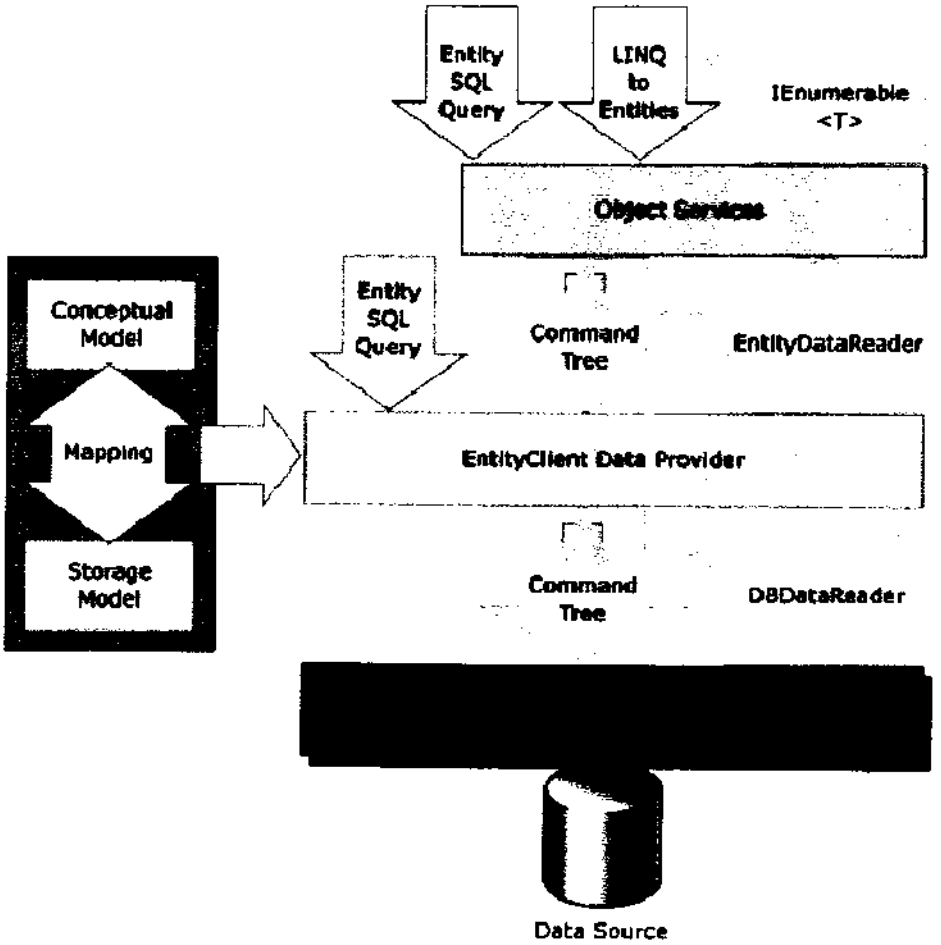
Сигурността на обменяните съобщения може да се гарантира чрез използване на WS-Security и чрез подписване и криптиране на всяко едно съобщение.

Програмният код, написан на .NET Framework се нарича управляван код, а също така и защитен код. Той се изпълнява в специална защитена среда наречена sandbox и е изолиран от хардуерната и софтуерната среда, в която работи. Това го предпазва от програмни грешки, които го правят уязвим за атаки, например препълването на буфера (buffer overflow). Платформата предлага система за сигурност, в която може да се интегрират всички използвани приложения.

8.2.8.7.Поддръжка на средства за централизирано съхранение на бизнес-обектите в релационна база данни

Microsoft .NET поддържа слените средства за централизирано управление и съхранение на обектите:

- ADO.NET Entity Framework Core – това е технология разработена от компанията Microsoft за достъп до данните. Технологията по същество представлява Object Relational Mapping (ORM), което позволява автоматизирана трансформация на релационните данни към проектирания обектен модел ползван от софтуерното приложение ;
- LINQ (.NET Language Integrated Query) – това е технология интегрирана в езиците за програмиране, част от платформата .NET, унифицираща и улесняваща начина на достъп до информация, която по своята същност и дефиниция не е обектно ориентирана, напр. XML документ, релационна база данни и т.н. LINQ дава унифициран подход за изпълнение на заявки, който е приложим към разнообразни източници на информация, като резултатът автоматично се трансформира към ползвания от софтуерното приложение обектно-ориентиран модел.



Фигура 25 - Централизирано съхранение и достъп до бизнес обектите

8.2.8.8.Поддръжка на утвърдени стандарти за обмен на данни с други системи

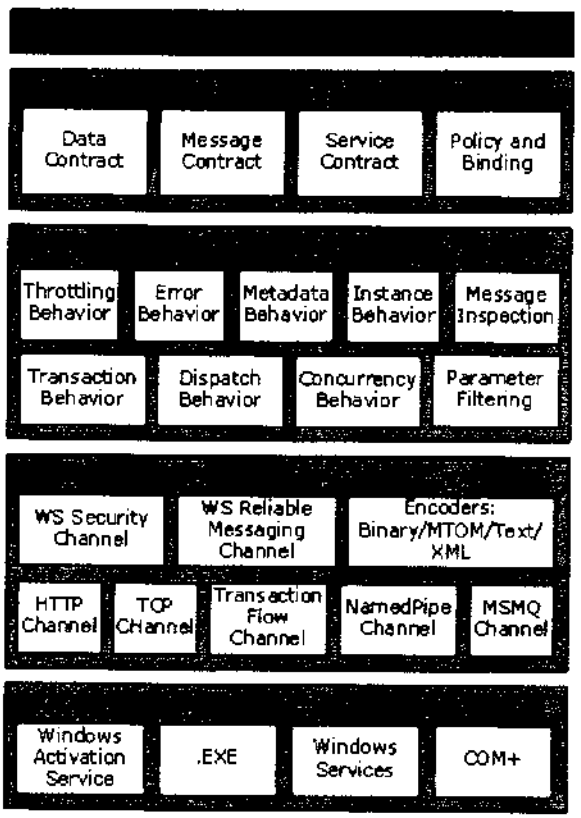
Обменът на данни с други системи ще бъде изграден с помощта на Windows Communication Foundation (WCF). WCF е надграждане на платформата (Microsoft .NET Framework), което е предназначено за изграждане на SOA решения със следните възможности:

- Широка съвместимост. WCF предоставя пълна поддръжка на всички протоколи, които са необходими за изграждане на инфраструктура от Web услуги (WS), в това число SOAP (Simple Object Access Protocol) за предаване на данни и изпълнение на конкретни операции;
- Интеграция с COM/COM+;
- Широк обхват на транзакцията. WCF поддържа следните начини за управления на транзакция: WS-AtomicTransactions, System.Transactions, MSDTC;
- Поддръжка на AJAX и REST;

- Различни начини за обмен на съобщения чрез създаване на контракти;
- WSDL(Web Service Description Language) за описание на интерфейса и предлаганите операции от една уеб услуги
- Сигурност. Технологията позволява използване на добре познатите стандарти SSL и WS-SecureConversation;
- Различни физически канали за обмен на съобщенията – HTTP, TCP, Named pipes и други.

8.2.8.9.Поддръжка на уеб услуги

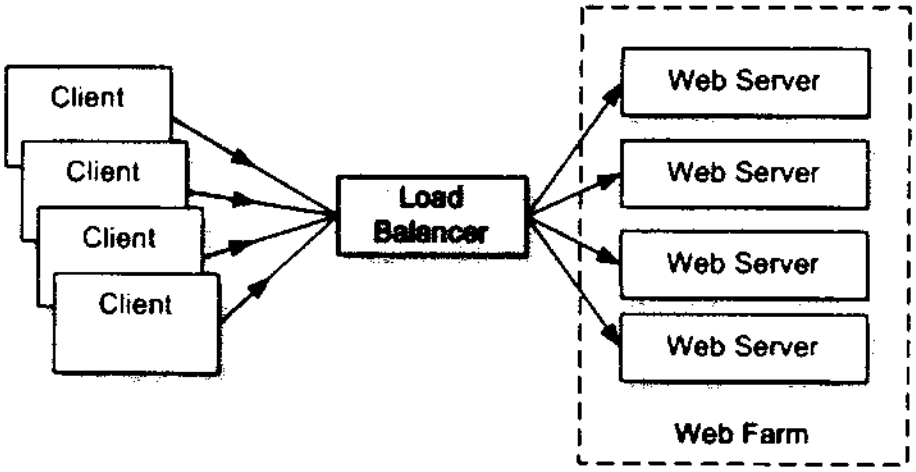
Част от Microsoft.NET е Windows Communication Foundation (WCF), която позволява да се създават интероперабилни уеб услуги. WCF покрива Web services (WS-*) стандартите, което позволява на приложенията да комуникират по платформено независим начин.



Фигура 26 - Архитектура на WCF

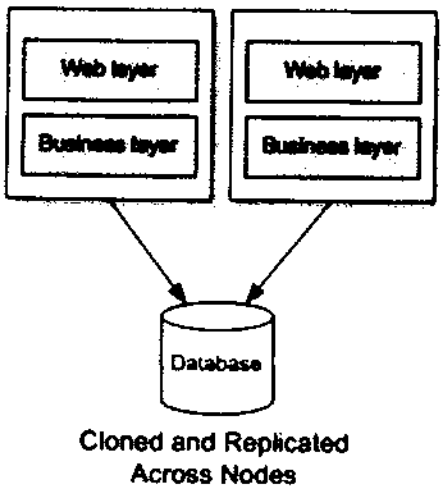
8.2.8.10. Скалируемост

Microsoft.NET разполага с вградени средства за осигуряване на скалируемост, възможност за паралелна работа на множество сървъри, разпределени транзакции и управление на натоварването. Microsoft.NET приложенията могат да работят във ферма от уеб сървъри в режим на разпределение на натоварването.



Фигура 27 - Управление на натоварването

Microsoft.NET дава възможност да се клонира даден приложен сървър, като по този начин софтуерната услуга може да се репликира в множество възли на дадена уеб ферма.



Фигура 28 - Разпределени транзакции

4.1.1. Средства за мониторинг

Microsoft.NET поддържа удобни средства и програмни интерфейси за мониторинг на ресурсите и приложенията. Това са:

- .NET Profiling API;
- Performance Counters;

- Performance Monitor;
- Windows Management Instrumentation (WMI).

4.1.2. Интеграция със средата за разработка Microsoft Visual Studio

Microsoft.NET е напълно интегрирана със средата за разработка Microsoft Visual Studio. Microsoft Visual Studio предоставя интегрирана среда състояща се от разнообразни инструменти и сървърна инфраструктура, която улеснява целия процес за проектиране и разработка на софтуерни приложения върху платформата .NET. Средата дава възможност за постигане на бързи бизнес резултати, чрез използването на продуктивен, предсказуем и адаптивни процес, предоставяйки прозрачност и проследимост през целия жизнен цикъл на разработка. Средата притежава мощни средства за проектиране, прототипиране, тестване и разработка, които позволяват бърз преход от създадената визия към реалното решение.

Microsoft Visual Studio води до увеличаване на производителността на екипа чрез използване на вградените модерни средства за сътрудничество, интегрираните инструменти за тестване и отстраняване на грешки, които позволяват лесното им откриване и бързото им коригиране, създавайки по този начин висококачествени решение с ниска производствена цена.

8.2.9. Предлагани средства за разработка на решението

За реализация на Системата ще се приложи обектно-ориентирания подход с използване на следните продукти и технологии в съответствие с най-добрите практики и разработки:

- ✓ Предлагаме система на управление на база данни (СУБД) PostgreSQL 9.6.. PostgreSQL е високопродуктивна, обектно-релационна система за управление на база данни (СУБД) с отворен код. СУБД се отличава с отлична скалируемост, гъвкавост и еластичност независимо от количеството данни, които управлява и броя на едновременните потребители, с които работи.
- ✓ Ще се използва Visual Studio 2017, като средство за разработка, а C#.NET и JavaScript като езици за програмиране. Посочената развойна среда е снабдена с качествени средства за визуален дизайн, поддържа множество програмни езици и редактори на кода. Позволява разработка и отстраняване на грешки в многопотребителски сървърни приложения чрез обединената среда за разработка;
- ✓ SVN - като хранилище на кода, за поддържане на версиите на софтуера, документацията и друга информация по проекта ИО АД използва внедрената при него система за управление на версия. Тази система позволява едновременна работа на членове на екипа върху едни и същи файлове, поддържа версии и подверсии, позволява сравнение, възстановяване и обединяване на поддържаната информация.

Системата ще бъде разработена на базата на .NET Core 2.0 и .NET Standart 2.0 – платформи с отворен код, позволяващи на приложението да работи под различни операционни системи (MS Windows, OS X, Linux) с помощта на следните технологии:

- ✓ C# и JavaScript – езици за програмиране, използвани при разработката на приложението;
- ✓ Entity Framework Core ORM – технология за достъп до бази данни с отворен код;
- ✓ ASP.NET Core – технология за разработка на Уеб приложения с отворен код;
- ✓ XML, JSON – стандарт за обмен на данни;
- ✓ AJAX - група от свързани техники за разработка на Интернет приложения, позволяващи асинхронно извличане на данни без нарушаване на състоянието на страницата;
- ✓ JQuery, AngularJS и други JavaScript библиотеки позволяващи предоставянето на по-качествен потребителски интерфейс и гарантиращи еднакво поведение на системата във всички модерни браузъри;
- ✓ HTML 5 – език за описание на Уеб страници;
- ✓ CSS 3 – език за описание на стилове в Уеб страници;
- ✓ Source code notation – за използване на единен подход при именуване на променливи, имена на класове, функции и процедури. Следи се и за еднотипно подравняване (alignment) на кода с цел по-лесната поддръжка и бързо ориентиране на програмистите при проследяване и отстраняване на несъответствие.
- ✓ Средства за тестване на програмния код – подробно описани по-долу.

Всички изброени технологии представляват последните тенденции в разработката на уеб приложения и се развиват от едни от най-големите и стабилни компании в света, **като Google** (AngularJS), **Microsoft** (Entity Framework, ASP.NET MVC) и **W3C** (XML, HTML, CSS). Избраните технологии ще гарантират дълга и безпроблемна експлоатация на разработената система, както и лесното й развитие в бъдеще.

8.2.9.1.Инструменти за изпълнение на предлаганата Методика за проектиране на компонентите

- ✓ **Microsoft Visio**

За настоящата поръчка ще бъде използван предоставения пакет от Visio за UML моделиране. Microsoft Visio е среда за проектиране на диаграми на софтуерни системи като предоставя пълен набор от средства за целите на бизнес моделирането. Microsoft Visio поддържа съвместна работа върху една диаграма:

- Коментиране;
- Лесно споделяне на диаграми чрез уеб браузър със записване в SharePoint;
- Коментиране чрез Visio Services;
- Съвместно авторство – двама или повече души работят върху една диаграма едновременно;
- Използване на информация за присъствие и виждане за незабавни съобщения или обаждане на други хора;

- Microsoft Visio представлява професионален инструмент за моделиране на бизнес правила с нотация на стандартни процеси;
- Опция за проверка на диаграми с вградени и разширяеми бизнес правила;
- Поддръжка за нотацията за моделиране на бизнес процеси Unified Modeling Language (UML) 2.4 и (BPMN) 1.2 и 2.0 диаграми;
- Подпроцеси за разбивка на сложни процеси в по-лесни за управление части.

UML нотацията и инструмента Microsoft Visio предоставят възможността за всеки бизнес процес да бъдат създадени графично и текстово описание, разпределено в следните части:

- Участници в процеса;
- Необходими условия за изпълнение на процеса;
- Описание на очакваното състояние на системата след изпълнение на процеса;
- Описание на дейностите извършвани от участниците в процеса. (Описанието може да се разглежда и като базисна функционалност на системата);
- Сценарий за изпълнение на процеса;
- Връзки на процеса с други процеси в системата.

✓ **Entity Framework – MSDN**

Entity Framework е набор от ADO.NET технологии, които подпомагат изграждането на модела на данни при разработката на софтуерни приложения. Архитектите и проектантите на базите данни. Използването на Entity Framework позволява на разработчиците да работят с данните под формата на domain-specific обекти и свойства, без да е необходимо запознаването им с конкретните колони и кортежи на таблиците на базата данни. С помощта на Entity Framework, се работи лесно и качествено на високо ниво на абстракция като същевременно се предоставя възможност за ясна визуализация на таблиците и релациите между обектите-данни.

8.2.10. Система за управление на база данни

При реализацията на модела на данни и изграждането на базата данни Postgres ще бъдат следвани добрите практики за дизайн и взаимодействие с базата данни, в т.ч.:

- дизайнът на схемата на базата данни (ако има такава) ще бъде с максимално ниво на нормализация, освен ако това не би навредило сериозно на производителността;
- базата данни ще може да оперира в клъстър; в определени случаи следва да бъде използван т.нар. sharding;
- имената на таблиците и колоните ще следват унифицирана конвенция;
- ще бъдат създадени индекси по определени колони, така че да се оптимизират най-често използваните заявки; създаването на индекс трябва да е мотивирано и подкрепено със замервания;

- връзките между таблици ще се дефинират чрез foreign key;
- периодично трябва ще бъде правен анализ на заявките, включително чрез EXPLAIN (при SQL бази данни), и ще бъдат предприети мерки за оптимизиране на бавните такива;
- задължително трябва ще се използват транзакции, като нивото на изолация трябва да бъде мотивирано в предадената документация;
- при операции върху много записи (batch) следва ще се избягват дългопродължаващи транзакции;
- заявките ще бъдат ограничени в броя записи, които връщат;
- при използване на ORM Entity Framework ще се минимизира броят на излишните заявки (т.нар. n+1 selects проблем);

В съответствие с изискванията на техническата спецификация предлаганото решение се базира на стандартна релационна база данни PostgreSQL, която покрива съвременните стандарти за бази данни от високо ниво, а същевременно е единствената подобна база, която не изисква лицензи за своите Enterprise версии. Текущата имплементация на SQL в PostgreSQL отговаря на стандарта ANSI-SQL: 2008. Той има пълна поддръжка за подчинените (включително под-селекции в клаузата FROM), read-committed и сериализирани нива на изолация на транзакциите. И докато PostgreSQL има напълно релационен системен каталог, който сам по себе си поддържа множество схеми на база данни, неговият каталог е достъпен и чрез информационната схема, както е дефинирано в SQL стандарта.

Предлаганите средства от избраното СУБД за осигуряване на цялостност на данните включват (комбинирани) първични ключове, чужди ключове с ограничаващи и каскадни актуализации / изтривания, проверка на ограниченията, уникални ограничения, не-нулеви ограничения. При моделирането на данните ще се използва подход, при който първичен ключ за всички таблици е системно генериран уникален идентификатор от тип BIGINT (serial в термините на PostgreSQL). Това касае всички таблици в системата без изключение.

Реализацията на обръщенията от страна на приложението към базата данни (DataAccessLayer) се използва платформата Entity Framework Core. Във въпросната е реализиран т.нар. lazy loading подход към зареждането на данни от СУБД. Това означава, че отделните свойства на даден бизнес обект и особено т.нар навигационни свойства се изчитат от базата данни само при необходимост, а не при всяко инициализиране на обекта.

Освен това избраната СУБД предоставя множество разширения и разширени функции. Сред удобствата са колони за автоматично увеличаване чрез последователности, LIMIT / OFFSET позволява връщането на комплекти за частични резултати. PostgreSQL поддържа комбинирани, уникални, частични и функционални индекси имплементирани на базата на B-tree, R-tree, hash или GiST.

Индексът GiST (Generalized Search Tree) е усъвършенствана система, която обединява широк набор от различни алгоритми за сортиране и търсене, включително B-tree, B + -tree, R-дърво, частично сумирани дървета, класирани B + -trees и много други.

Той също така осигурява интерфейс, който позволява както създаването на персонализирани типове данни, така и разширяемите методи за търсене. По този начин GiST предлага гъвкавостта да посочва какво се съхранява, как се съхранява и как да се дефинират нови начини за търсене чрез него - начини, които далеч надхвърлят тези, предлагани от стандартното B-tree, R-tree и останалите алгоритми за общо търсене.

GiST служи като основа за много публични проекти, които използват PostgreSQL като OpenFTS и PostGIS. OpenFTS (Open Source Full Text Search Engine) осигурява онлайн индексиране на данни и значение на класирането за търсене на база данни .PostGIS е проект, който добавя поддръжка за географски обекти в PostgreSQL, позволявайки му да се използва като пространствена база данни за географски информационни системи (GIS), подобно на SDE на ESRI или на Spatial разширение на Oracle.

Другите допълнителни функции включват наследяване на таблици, системи за правила и събития в базата данни. Наследяването на таблицата поставя обективно ориентиран подход върху създаването на таблицата, което позволява на дизайнерите на бази данни да извличат нови таблици от други таблици, като ги третират като базови класове. Още по-добре, PostgreSQL поддържа едновременно и множествено наследяване на таблици.

Системата за правила (Rules), наричана още система за преразглеждане на заявки, позволява на дизайнера на базата данни да създава правила, които идентифицират конкретни операции за дадена таблица или изглед и динамично ги превръща в алтернативни операции, когато се обработват.

Системата за събития е система за междупроцесна комуникация, в която съобщенията и събитията могат да се предават между клиентите чрез командите LISTEN и NOTIFY, позволяващи както проста комуникация от тип peer to peer, така и усъвършенствана координация на събития в базата данни. Тъй като известията могат да бъдат издавани от тригери и съхранени процедури, клиентите на PostgreSQL могат да наблюдават събития в базата данни, като например актуализации на таблици, вмъквания или изтривания, когато се случват.

PostgreSQL предлага набор от взаимно независими концепции за осъществяване на репликация на данните. Те могат да се използват както самостоятелно така и комбинирано.

- Тригер-базирана репликация - тази техника е остаряла и не се използва.
- Log Based Replication – между сървърите се прехвърля такава информация, която описва промените в данните и така или иначе се създава и съхранява в WAL файлове.
- WAL-File-Shipping Replication (или репликация на базата на файлове) означава прехвърляне на напълно попълнени WAL файлове (16 MB) от един сървър на друг. Тази техника не е много елегантна и е заменена от поточна репликация.
- Streaming или поточна репликация означава прехвърлянето на регистрационни записи от лог файла (информация за единична трансакция) от основния сървър във вторичния през TCP връзка.

Във физически формат прехвърлените записи WAL имат същата структура, каквато се използват в WAL файловете. Те отразяват структурата на файловете на базата данни, включително блокови номера, информация за VACUUM и др.

Логическият формат на кодиране/декодиране на WAL записите е в абстрактен формат, който е независим от версии на PostgreSQL и хардуерни платформи. Това позволява репликация при сървъри с различни версии на PostgreSQL

При асинхронната репликация данните се прехвърлят на друг възел, без да се чака потвърждение за получаването му. При синхронна репликация прехвърлянето на данни чака - в случай на COMMIT - потвърждение за успешната й обработка в режим на готовност.

Архитектурата Master / Standby представлява топология, в която един или много възли на готовност получават данни за промяна от един главен възел. В такива ситуации възлите в режим на готовност могат да възпроизведат получените данни до други възли, така че те са главни и в режим на готовност едновременно.

Архитектурата Multi-Master обозначава топология, при която един или много възли в режим на готовност получават данни за промяна от много главни възли.

8.2.11. Предложение за извършване на ежедневен бекъп на данни от Изпълнителя

Изпълнителят ще реализира системен процес (cron task), който да извършва ежедневен бекъп на системата, да следи правилната работа на компонентите (health check), да събира информация за параметрите на бързодействието на компонентите. В случай на грешка при изпълнението на ежедневния бекъп, администраторите на системата и Изпълнителят ще бъдат уведомени по електронна поща, след което екипът на Изпълнителя ще извърши ръчно бекъпа с изпозлване на команда pg_dump dbname > outfile.

Ежедневния бекъп извършван от автоматизирания процес ще работи на принципа на Continuous Archiving and Point-in-Time Recovery (PITR) – този начин за архивиране се базира на създаване на базов бекъп (base backup), който се използва като начална точка на верига от състояния на базата, към които системата може да бъде възстановена във всеки момент. След създаване на базовия архив се архивират само WAL файловете, които съхраняват лога на транзакциите на системата. По този начин бекъпа може да се изпълнява максимално често и без да се изразходва дисково пространство за пълен бекъп, а поради възможността за максимална честота на бекъпите се елиминира потенциалната възможност за загуба на данните от последния бекъп, която съществува при нормален dump на базата, който не може да се изпълнява толкова често.

8.3.Дейност 1: Изработване на софтуерна платформа на информационната система

8.3.1. Описание на дейността

Информационната система за отчитане на опасни битови отпадъци ще бъде разработена така, че да отчита количеството, вида на предаваните отпадци лицето, което ги предава и лицето, което ги приема. Достъп до нея ще има от администрациите на 22-те целеви общини, включително 5-те пилотни центъра за екологосъобразно събиране и временно съхранение на опасни битови отпадъци, обслужващи тези общини, както и от организацията на Възложителя. Платформата ще поддържа едновременната работа на множество ползватели и доставчици на информация (вкл. администрации) чрез функционирането си във виртуална среда, ще бъде гъвкава и ще има опция за разширяване на възможностите – включване на всички общини в Република България, в т. ч. и съответен брой площадки (не по-малко от 50) за събиране на опасни битови отпадъци, обслужващи тези общини.

8.3.2. Изисквания към изпълнение на дейността

Задача 1.1. Изработване на техническите характеристики на софтуерната платформа на системата

Спецификациите на софтуерната платформа ще съдържат:

- Проектен план, вкл. план за управление на качеството и рисковете;
- Спецификация на изискванията – функционални и нефункционални и Спецификация на потребителски случаи
- Прототипи на потребителските интерфейси за въвеждане на данни;
- Софтуерна архитектура;
- Инфраструктурен модел;
- Модел на данните на логическо ниво;
- Описание на всички източници на данни за хранилището от данни
- Модел за зареждане на данни (ETL процедури)
- Модел на данните в хранилището за данни - datamarts
- Тестови модел.

Спецификация на изискванията – функционални и нефункционални, която ще опише ясно и недвусмислено изпълними изисквания към системата, ориентирани към изпълнението на нейните цели. Тя ще даде възможност на Възложителя да валидира изискванията. Изискванията ще са описани структурирано, така че да е възможно управлението на промените в тях.

Спецификацията на потребителските случаи ще опише специфичните изисквания за взаимодействие на потребителите със системата, включително други системи. Ще бъде дадена връзка между потребителските случаи и структурираните изисквания.

Прототипите на потребителските интерфейси за въвеждане на данни ще се създадат със средство, което позволява създаването на динамични прототипи. Прототипите на потребителските интерфейси ще дадат възможност за изследване на удобството на работа със системата. Методиката за проектиране на потребителския интерфейс в т. 8.1.3.4. на предложението.

Софтуерната архитектура ще отговаря на целите на системата и ще бъде базирана на съвременните тенденции за разработка на бази данни – използване на архитектура ориентирана към услугите (SOA). Инфраструктурният модел ще отговаря на

софтуерната архитектура, ще дава ясна картина за гъвкавостта и разширяемостта на системата.

Моделът на зареждане на данни ще описва необходимите процедури за захранване на хранилището на данни. Тези процедури включват извличане, трансформация и зареждане на данните. Могат да са вътрешни или външни за системата. Подходът за реализация на зареждането на данни (ETL) е описан в т 8.2.4. от предложението.

Тестовият модел ще съдържа подхода за провеждане на тестовете, видовете тестове, както и първоначална спецификация на тестовите случаи, базиран на Спецификация на изискванията – функционални и нефункционални и Спецификацията на потребителските случаи, като опише най-малко критериите за постигане на всяко изискване. Пълната спецификация на тестовите случаи трябва ще се предаде заедно с предаване на демо версия на софтуерната платформа на системата за приемане преди тестването на терен. Методиката за тестване на системата е описана в т. 8.1.6. на предложението.

Задача 1.2. Изработване на софтуерната платформа

Изпълнителят ще сформира екип от специалисти, които да разработят всичките модули на системата. Това предполага успоредна работа на различни екипи по различни модули/елементи на софтуера на системата.

Софтуерът на системата щеосигурява следните основни функционалности:

8.3.3. Събиране, зареждане, организация и съхраняване на данни

а) Събиране

Системата ще събира данни за приетите опасни битови отпадъци на 5-те пилотни общински центъра, както и площадките, които ще бъдат включени допълнително към системата.

Също така ще може да се опише агрегатното състояние на опасните битови отпадъци – течни, твърди, газообразноб под налягане. За тази цел ще се реализира един от следните начини на събиране и зареждане на данни в системата:

- Чрез зареждане на „плоски файлове“ по предварително определени спецификации на данните в тях;
- Чрез първоначално зареждане и последващо обновяване на номенклатури и класификатори, по които ще могат да се преглеждат данните („измерения“);
- Чрез предварително дефинирани форми за въвеждане на структурирани данни от крайни потребители на системата.

При зареждане на данните ще се предвидят необходимите ETL процедури и процедури за наблюдение на зареждането и коректривни дейности при отклонения. Решението ще позволява автоматизиране на ETL процесите, мониторинг и администрация над изпълнението на ETL процедурите. Описание може да бъде намерено в т. 8.2.4. от настоящото предложение.

б) Съхранение на данните в бази данни

Зареждането на данните и последващото им съхранение ще се извърши при прилагане на съвременните методи за организация на данни в хранилища от данни (Data Warehouse). Организацията на данните ще бъде такава, че да отговаря в максимална степен на нуждите от отчети и анализи на целевите администрации и Възложителя.

8.3.4. Анализ на данните

а) Статистически анализ

Системата ще дава възможност за базови функции като:

- филтриране, съпоставяне и кросиране на данни по определени критерии;
- представени в таблици и графики;
- изчисляване на производни индикатори, дефинирани в административния интерфейс;
- заложено автоматично изчисляване на съставни /многомерни показатели, индекси и коефициенти;
- изчисляване на нормално разпределение, процентно разпределение, други статистически обработки;
- изчисляване на трендове;
- разпределени по типове и сегменти; обобщаване и преглед на детайли (Drill-down), Pivot таблици;
- корелационен факторен, сегментационен анализ, хистограми.

Системата ще позволява:

- експорт на данни към външни системи за статистически анализ;
- съхраняване на таблични или графично представяне на резултати от статистически анализ за ползване в документи.

б) Обобщаване и експертен анализ на данните:

- Възможност за специално упълномощени потребители (посредством потребителски интерфейс) да обобщават и дават експертни оценки на изпълнението, използвайки събраните данни по заложените в системата индикатори, изчислените производни индикатори и добавени нови индикатори за даване на експертна оценка с оглед на заложени критерии за изпълнение, ефективност и ефикасност;

За тази цел трябва да се предвиди създаване на връзка от елемент/и от системата на индикаторите към елемент/и от системата на интервенциите.

- Възможност за автоматично изчисляване на показатели или експертна оценка.
- Приложение към йерархични организационни структури или обхвати на мониторинг. Графично представяне.

Функционалността ще бъде реализирана от модула за справки, който ще бъде реализиран в системата, описан в т. 8.2.6.5. от предложението

8.3.5. Представяне на обработени данни в справки и отчети

Информационната система ще дава възможност за представяне на обработената информация в графичен и табличен вид. За тази цел тя ще може да генерира различни видове документи като – справки, отчети, дашборди. Ще бъдат възможни:

- създаване на документ по предварително разработен шаблон;
- запазване на създадените отчети и анализи в системата за управление на документи с набор от предварително дефинирани атрибути и контрол на достъпа;
- експорт на създадените отчети и анализи в Excel и PDF.

Функционалността ще бъде реализирана като част от модула за справки на системата.

8.3.6. Управление на документи

В централизирано документно хранилище следва да се съхраняват:

- Всички постъпили файлове, които са първоизточници на данни, заредени в системата;
- XML документи, постъпили в резултат на уеб услугите, с които се предават данни от други системи;
- Плоските файлове, от които се зареждат данни;
- Първоизточници на данни, от които се зареждат данни чрез потребителския интерфейс;

Функционалността ще бъде реализирана под формата на модул за управление на документи, описан в т. 8.2.6.3. от предложението

8.3.7. Дигитална карта, представяща информацията по териториално местонахождение

Системата ще позволява базово интерактивно представяне на показатели, резултати и данни спрямо моделите на отделните площадки за събиране на опасни битови отпадъци, създадени по проекта на възложителя. Реализацията ще се базира на модула Дигитална карта, описан в т. 8.2.6.4. от предложението

8.3.8. Одит лог на действията – записи на основните операции върху всички документи и обекти. Възможности за извличане на записи на операции по различни критерии.

Съгласно техническите спецификации, системата ще поддържа системен журнал, в който да регистрира всички системни събития, като осигури и механизъм за проследяването им.

Функционалностите във връзка с водене на системен журнал ще бъдат реализирани по следния начин:

- ще бъде създаден списък на възможните събития, като на всяко събитие ще бъде присвоен уникален идентификационен код;
- събитията ще бъдат разделени по категории според резултат (успешни, неуспешни, грешки и др.) и според тяхната критичност. Всяко събитие ще получава код от категорията, съответстваща на характеристиките на събитието.
- всички събития, включително системните грешки, със съответните им кодове ще бъдат документирани в ръководството на администратора на системата;
- ще бъде създаден системен журнал на събитията, като за максимална надеждност ще се използва формата на операционната система на сървъра, на който се води журнала (файл или системна база данни);
- ще бъде разработен и публикуван метод за отразяване на събития в журнала, който ще е достъпен за всички модули на системата;
- при извикване на метода:
    - датата и часът ще се попълват автоматично от сървъра;
    - системният процес ще се попълва автоматично от клиента;
    - клиентът задължително ще подава вид/характер на действие и код на събитието;
    - ако събитието е грешка, клиентът задължително ще подава текст на съобщение за грешка;
    - при други събития, клиентът ще подава пояснителен текст, когато е възможно;
- при успешна или неуспешна актуализация на системните номенклатури, задължително ще се подава информация към метода за отразяване на събитията в журнала;
- ще бъде разработен програмен интерфейс за получаване на информация за събития, отговарящи на различни критерии: период, системен процес, код на събитието, вид/характер на действието и текст (съобщение за грешка или пояснителен текст). При регистриране на нови събития, отговарящи на зададените критерии, информацията за тях ще се подава автоматично.
- ще бъде разработен клиент на интерфейса за получаване на информация, представляващ справка за събития според предвидените критерии, с автоматично обновяване при настъпване на нови събития. Справката ще е достъпна само за администратори на системата.
- ще бъде създаден клиент на интерфейса за получаване на информация, изпращащ съобщения до специални потребители при настъпване на събития, определени чрез предвидените критерии;
- в модула за администриране ще бъде заложена функционалност за настройка на параметрите, необходими за работата на процеса, изпращащ съобщения за настъпили събития, а именно: