одобрява дефинираните задачи и ги разпределя на съответен разработчик за изпълнение. След изпълнение на задачата в другия модул се отразява статус в модула за регистриране на проблеми.

Целта на системата е да се централизира информацията за всички задачи на екипа по разработка – както задачи по разработка на нова и надграждане на съществуваща функционалност, така и задачи във връзка с отстраняване на софтуерни грешки в една база данни. По този начин се осигурява бърз достъп до информацията за всички промени във версията върху която се работи, което облекчава планирането, следенето на напредъка, както и анализа на потенциалните регресионни отражения на промените.

Redmine позволява удобно да се регистрират, разпределят, наблюдават, планират и отчитат задачите. Модула за регистриране на проблеми улеснява както връзката на потребителите (представители на Възложителя) с отделите по поддръжка на Изпълнителя, така и сътрудничеството и комуникацията между специалисти по поддръжката, анализатори и разработчици в самата организация на Изпълнителя.

Всички лица с регистрация в системата имат възможност да проследяват развитието по отстраняване на проблема в реално време посредством уеб интерфейс.

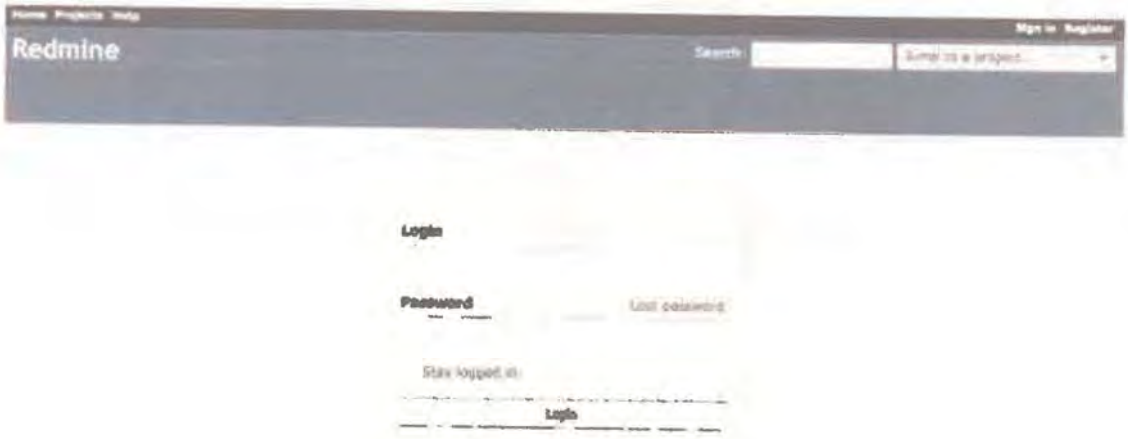
Redmine е безплатен и с отворен код, уеб базиран инструмент за управление на проекти и проследяване на проблеми. Тя позволява на потребителите да управляват множество проекти и свързани с тях подпроекти. Той разполага с уики и форуми за проекти, проследяване на времето и гъвкав контрол на достъпа, базиран на роли. Той включва календар и диаграми на Gantt, които подпомагат визуалното представяне на проектите и техните крайни срокове. Redmine се интегрира с различни системи за управление на версиите и включва браузър за хранилища и дифузор.

Някои от основните характеристики на Redmine са:

- Уеб базиран административен панел, позволяващ настройка и персонализиране на системата за проследяване на проблеми;
- Гъвкав контрол на достъпа, базиращ се на потребителски роли с възможност за определяне на техните права;
- Гъвкава система за проследяване на проблеми, с възможност за дефиниране на статуси и типове проблеми;
- Управление на новини, документи и файлове;
- Известия по имейл;
- Проследяване на времето за изпълнение на дадена задача;
- Персонализирани полета за проблеми, записи, проекти и потребители;
- Множествената поддръжка на LDAP удостоверяване;
- Многоезична поддръжка;
- Поддържат се множество бази данни - MySQL, PostgreSQL или SQLite.

Екран за вход в Redmine:

Чл.36 а, ал. 3 от ЗОП



Страницата за вход се използва за влизане в проекта, в който сте били активирани. Връзката за изгубена парола се показва само ако администраторът я е активирал.

Регистриране на потребител:

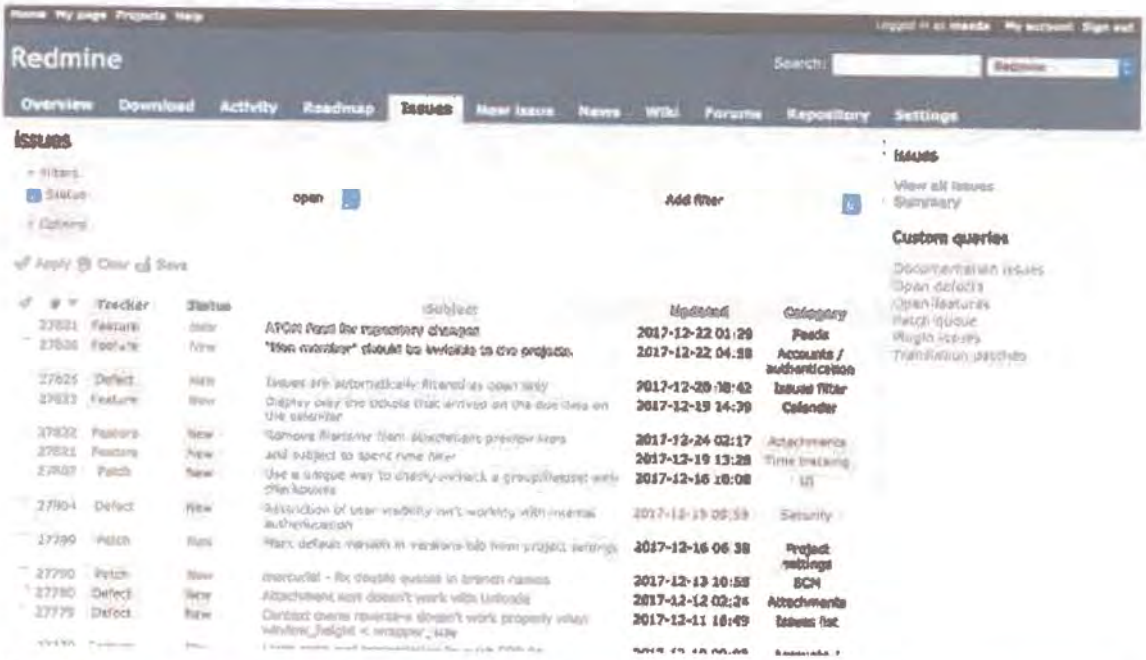
Въз основа на настройките на Redmine, потребителят ще трябва да активира профила си по имейл, да изчака администраторът да потвърди профила или да види активирането му автоматично.

- Активиране на профила по имейл - След като предостави необходимата информация на страницата за регистрация, потребителят ще получи имейл на имейла, предоставен на страницата за регистрация. С кликването върху връзката за активиране, предоставена в имейла, потребителят ще активира своя профил.
- Ръчно активиране на акаунта - След като предостави необходимата информация на страницата за регистрация, потребителят трябва да изчака одобрението на администратор.

Чл.36 а, ал. 3 от ЗОП

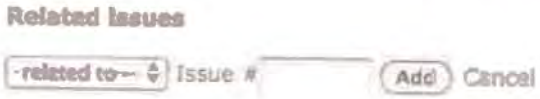
Администраторът ще активира потребителския акаунт на страницата "Администрация"> "Потребители". Когато потребителският профил е активиран, потребителят ще има право да влиза в системата.

Регистриране и проследяване на проблеми:



От избрана страница с проблеми можете да видите текущата работа, която е извършена, по отстраняване на проблема. Съобщенията се показват в хронологичен ред (за да промените реда - вижте настройката в "Моите профили"). Възможно е да цитирате други съобщения, както и да редактирате вашите.

Системата позволява създаване на връзка между свързани проблеми. Това улеснява работата на Изпълнителя по отстраняване на проблеми, като в случай на дублиране, може да се премахнат излишните позиции. В случай, че даден проблем произтича от вече регистриран или има определен ред, в който проблемите следва да бъдат отстранени, това също може да се отбележи посредством такива връзки.



Добавяне на нов проблем

Хората могат да създадат нов проблем, когато отговарят на ролите и правата, конфигурирани от администратора на Redmine (Роля: Проследяване на емисиите> Добавяне на проблеми).

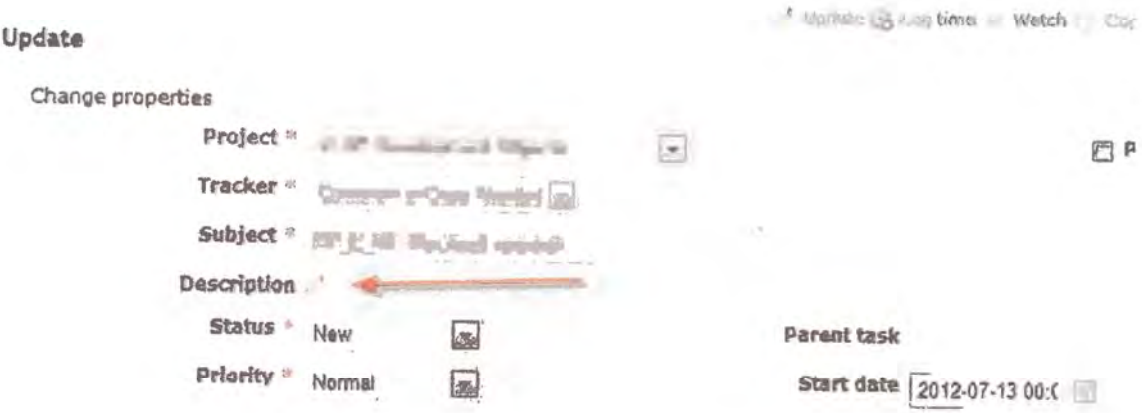
Чл.36 а, ал. 3 от ЗОП

При създаването на нов проблем един от най-важните елементи е полето за проследяване, което определя естеството на проблема. По подразбиране Redmine идва с три различни тракера: бъг, функция и поддръжка.

Актуализиране на съществуващ проблем

В зависимост от правата, с които разполага дадената потребителска роля, ще бъде визуализиран пълен или ограничен набор от свойства на проблеми, които могат да се редактират.

Редактиране на предмет или описание на съществуващ проблем



7.2.1.6. Интегриране (build) на софтуерните модули

Завършените софтуерни модули се интегрират от програмистите в цялостен софтуерен продукт на базата на софтуерните спецификации и дизайн, при спазване на вътрешните конвенции за кодиране. Тук се реализират и интерфейсите за интеграция с външните за софтуерния продукт системи.

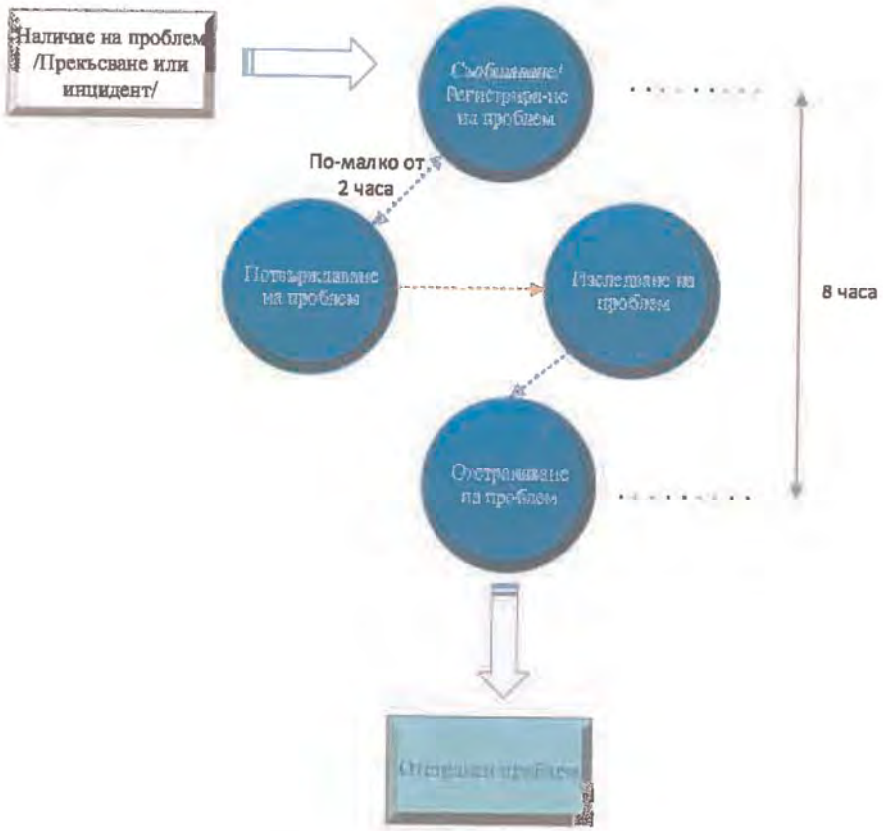
Както и при разработването на софтуерните модули и тук планът на проекта е основата за наблюдение на дейностите и за предприемане на коригиращи действия.

7.2.1.7. Процедурата за управление на софтуерни грешки/несъответствия/проблеми

Подходът, който Изпълнителят ще прилага при откриване на грешки/несъответствия/проблеми, е представен схематично на Фигура 18.

Чл.36 а, ал. 3 от ЗОП

Гаранционна поддръжка



Фигура 25 Процедура за отстраняване на грешки/несъответствия/проблеми

Представеният подход представлява набор от последователно извършвани дейности с оглед осигуряване на безпроблемната и надеждна работа на компонентите и обхваща следните основни стъпки:

- ✓ **Съобщаване/Регистриране на грешка** (несъответствие/проблем). Проблемът се описва чрез Система за регистриране и управление на заявки за проблеми и дефекти, за да бъде разгледан и впоследствие разрешен.
- ✓ **Потвърждение на проблем** – включва дейности по преглед и потвърждаване на изпратения проблем от експерт от екипа на Изпълнителя, като времето от съобщаването до потвърждаването ще е по-малко от 2 часа;
- ✓ **Изследване на проблем** – включва дейности по преглед и анализ на информацията по изпратения проблем – възможни причини и последствия от възникването на проблема, възможни решения и начини за отстраняване на проблема;
- ✓ **Разрешаване на проблем** – включва дейности по отстраняване на проблема, тестване на решението и възстановяване на работоспособността на услугата.

Отстраняването на проблеми ще включва следните под-дейности:

Чл.36 а, ал. 3 от ЗОП

- ✓ **Възпроизвеждане и/или анализ на проблема.**
- ✓ **Намиране решение на проблема и съгласуване с бизнес и ИТ служител/и от екипа на Възложителя.**
- ✓ **Реализация на промените в средата за разработка.**
- ✓ **Тестване на промените в средата за разработка.**
- ✓ **Верификация на промените от страна на Възложителя.**

Изпълнителя предоставя на определен служител от екипа на Възложителя версия с всички корекции, извършени през дейностите на тестване и последваща гаранционната поддръжка на Source-кода върху електронен носител на информация CD/DVD, като се подписва приемо-предавателен протокол.

7.2.2. Методика за внедряване

За изпълнение на дейностите с необходимото качество, в срок и в съответствие с изискванията по проекта, при планиране ще бъдат предприети действия за определяне на:

- процедурата за внедряване, включително подготвителните действия за внедряване на системата;
- ресурсите, необходими за изпълнение, включително екипа от експерти;
- спецификация на тестовете за приемане на системата в окончателен вариант;
- процедура за изграждане на продукционната среда;
- график за изпълнение на дейностите.

Организацията на изпълнение като минимум ще включва:

- разработване на инструкцията за изпълнение на процедурата за внедряване;
- подготовка на експертите, включени в екипа за внедряване, за изпълнение на дейностите по процедурите за изграждане на продукционна среда и за внедряване;
- разработване на образците на документи за отчитане изпълнението на дейностите.

Всички дейности във фазата на планирането ще бъдат съгласувани с Възложителя.

Резултатите от дейностите по планиране внедряването на системата ще бъдат обобщени в План за внедряване, който ще бъде представен на Възложителя за одобрение – на хартия и в електронен формат. Планът за внедряване като минимум ще включва:

- Подготовка за внедряване – описание на подготвителните дейности, които трябва да бъдат извършени преди внедряване;
- Процедура за внедряване – детайлно описание на дейностите с подробни указания за изпълнение на всяка стъпка;
- Процедура за изграждане на продукционна среда с подробно описание на действията

Чл.36 а, ал. 3 от ЗОП

за инсталация и конфигуриране на компонентите на системата;

- Екип за изпълнение – списък на експертите, включени в екипа за изпълнение и данни за контакт (стационарен телефон, мобилен номер, е-мейл адрес, административен адрес за кореспонденция и т. н.);
- Спецификация на тестовете за приемане на системата;
- Образец на протокол за изграждане на продукционна среда;
- Образец на протокол от проведени тестове за приемане на система, в който ще се отразяват резултатите от изпълнение на тестовете;
- Образец на протокол за внедряване за отчитане изпълнението на дейностите;
- График за изпълнение.

✓ **Изграждане на продукционна среда**

Изграждането на продукционната среда ще бъде извършено по начина, определен във фазата на планиране на внедряването и в съответствие с разработената процедурата за изграждане на продукционна среда.

✓ **Изпълнение на дейностите по внедряване**

При внедряване на компонентите ще бъдат извършени действията, определени във фазата на планиране на внедряването, в съответствие с разработената процедура за внедряване.

Дейностите ще бъдат извършени по одобрените от Възложителя План за внедряване и график за изпълнение и по предварителна оценка ще включват:

- параметризация на базата данни, включително създаване на системните класификатори, конфигуриране на системата;
- зареждане в базата данни на мигрираните данни;
- провеждане на тестове за приемане на системата, в хода които ще се извърши и представяне пред екип на Възложителя на реализираните функционалности, което ще гарантира навременно запознаване със системата.

✓ **Отчитане изпълнението на дейностите по внедряване**

Отчитане изпълнението на дейностите по внедряване ще бъде включено в междинния доклад за изпълнение на Етап 4: Внедряване, в който ще бъдат изложени поставените цели и постигнатите резултатите. Ако в хода на изпълнение са възникнали проблеми, те ще бъдат документирани. Ще бъдат описани и предприетите действия за преодоляването им. Към доклада ще бъдат приложени оригинали на:

- протокол за изграждане на продукционна среда;
- протокол от изпълнението на дейностите по внедряване;
- протокол от проведените тестове за приемане на системата.

Чл.36 а, ал. 3 от ЗОП

7.2.2.1. Начин на прилагане на предлагания подход за внедряване на компонентите

✓ **Планиране на внедряването и организация на изпълнението**

Във фазата на планиране на внедряването ще бъдат предприети следните конкретни стъпки за подготовка и изпълнение на дейностите:

- ще бъде разработена процедурата за внедряване с указания за изпълнение на всяка стъпка;
- ще бъдат определени ресурсите, необходими за изпълнение на дейностите в срок и с необходимото качество;
- ще бъде сформиран екип за изпълнение – експерти, които ще извършат дейностите по внедряване;
- ще бъде извършен преглед на изготвената при планиране на тестването Спецификация на тестовете за приемане на системата, включително и на изготвената процедура за връщане на системата в последно работоспособно състояние (Rollback Procedure) и ако е необходимо, ще бъде актуализирани обхвата и вида на планираните тестовете за приемане;
- ще бъде разработен образец на протокол от проведени тестове за приемане на системата, в който ще се отразяват резултатите от тестовете;
- ще бъдат разработена процедурата за създаване на продукционната среда с подробно описание на последователността от действия;
- ще бъде подготвен образец на протокол за присмане на дейностите по изграждане на продукционната среда;
- ще бъде подготвен образец на протокол за внедряване за отчитане изпълнението на дейностите по внедряване;
- ще бъде изготвен детайлен график за изпълнение на дейностите по:
  - изграждане на продукционната среда;
  - внедряване и провеждане на тестове за приемане, съпроводени с представяне пред екип на Възложителя на реализираните функционалности с конкретна информация за период на изпълнение и експертите, които ще извършат дейностите.

Всички дейности във фазата на планирането ще бъдат съгласувани с Възложителя, като част от тях ще бъдат изпълнени със съдействието на екипа на Възложителя.

Резултатите от дейностите по планиране внедряването на системата ще бъдат обобщени в План за внедряване, който ще бъде представен на Възложителя за одобрение, в 2 екземпляра на хартия и в редактируема електронна форма ('.doc' или '.docx' файлов формат). Планът за внедряване като минимум ще включва:

Чл.36 а, ал. 3 от ЗОП

- Подготовка за внедряване – описание на подготвителните дейности, които трябва да бъдат извършени преди внедряване;
- Процедура за внедряване – детайлно описание на процедурата и дейностите, които ще бъдат извършени, с подробни указания за изпълнение на всяка стъпка;
- Процедура за изграждане на продукционна среда с приложено ръководство за администратора за инсталация и конфигуриране на системата с подробни указания от типа „стъпка по стъпка“ за последователността от действия за подготовка на продукционната среда
- Екип за изпълнение – списък на експерти с данни за контакт (телефон, мобилен номер, е-мейл адрес, адрес за кореспонденция и т. н.);
- Спецификация на тестове за приемане на внедряването, включително и на изготвената процедура за връщане на системата в последно работоспособно състояние (Rollback Procedure);
- Образец на протокол от проведени тестове за приемане на внедряването, в които ще се отразяват резултатите от тестовете;
- Образец на протокол за внедряване, който ще послужи за отчитане изпълнението на дейностите по внедряване;
- Процедура за създаване на продукционната среда с приложено ръководство;
- Образец на протокол за отчитане изпълнението на дейностите по изграждане на продукционната среда;
- Детайлен график за изпълнение на дейностите по внедряване.

✓ **Изграждане на продукционна среда**

Изграждането на продукционната среда, в която ще функционира Системата, ще се извърши в съответствие с одобрения План за внедряване и разработената Процедура за създаване на продукционна среда, която като минимум ще включва следните действия:

- инсталация и настройка на операционната система и друг системен софтуер;
- инсталация на компонентите (софтуерни компоненти и база данни);
- настройки на конфигурационните файлове;
- изпълнение на сервизни скриптове, ако е необходимо и т. н.

За изпълнение на дейностите по изграждане на продукционната среда ще бъде изготвен протокол.

✓ **Изпълнение на дейностите по внедряване**

Цялостното внедряване на компонентите ще бъде проведено по одобрените от Възложителя План за внедряване и детайлен график за изпълнение на дейностите.

Процедурата по внедряване на компонентите по предварителна оценка като минимум включва:

Чл.36 а, ал. 3 от ЗОП

- параметризация на базата данни, включително създаване на системните класификатори, конфигуриране на системата;
- зареждане в базата данни на „плоски файлове“ с агрегирани данни;
- провеждане на тестове за приемане на внедряването и отразяване на резултатите в съответния документ.

Тестовете за приемане на внедряването ще бъдат извършени съвместно с екипа на Възложителя, като резултатите ще бъдат отразени в протокол.

✓ **Процедура за създаване на продукционната среда**

Описание ще включва подробна информация за процедурата и действията по създаване на продукционната среда. Към описанието ще бъде приложено ръководство за инсталация и конфигуриране за администратора с подробни указания стъпка по стъпка за действията, които трябва да бъдат извършени.

Действията по изграждане на продукционната среда, в която ще функционира системата като минимум ще включват:

- инсталация и настройка на операционната система;
- инсталация на система (софтуерни компоненти и база данни);
- параметризация на базата данни;
- настройки на конфигурационните файлове;
- изпълнение на скриптове, ако е необходимо и т. н.
- подпис на протокол за отчитане изпълнението на дейностите по изграждане на продукционната среда.

В хода на планиране на внедряването с Възложителя ще бъде обсъден подход за параметризация на системата.

Процедурата и дейностите за изграждане на продукционната среда ще бъдат специфицирани при планиране на внедряването, ще бъде разработено и ръководството с детайлна информация за действията, които следва да бъдат извършени с подробни указания от типа „стъпка по стъпка“. Процедурата и приложенията ще бъдат представени на Възложителя за одобрение като част от Плана за внедряване.

✓ ***Образец* на *протокол за изграждане* на *продукционна среда***

Най-общо протоколът ще съдържа информация за:

- Възложител;
- Място на изпълнение на дейностите по изграждане на продукционна среда;
- Представител(и) на Възложителя, присъствал(и) и осъществил(и) контрол на изпълнение на дейностите по изграждане на продукционна среда;
- Експерт(и), извършил дейностите по изграждане на продукционна среда;
- Период на изпълнение;

Чл.36 а, ал. 3 от ЗОП

- Извършени действия – по процедурата за изграждане на продукционна среда и допълнителни дейности, ако е възникнала необходимост от такива;

Образецът ще бъде разработен в окончателен вид при планиране на внедряването и ще бъде представен на Възложителя за одобрение като част от Плана за внедряване.

✓ ***Образец на* протокол *за провеждане на* тестове *за приемане на системата***

Най-общо протоколът ще съдържа информация за:

- Възложител;
- Място на изпълнение на тестовете за приемане на системата;
- Представител(и) на Възложителя, участвал(и) в провеждането на тестовете за приемане на системата;
- Експерт(и), провели тестовете за приемане на системата;
- Таблица с планираните тестове, като се всеки тест ще има описание на критерият, въз основа на който се оценява резултатът от изпълнение. Например:

№ по ред	Тест	Критерий за успешно изпълнение	Получен резултат (да/не)
1	2	3	4
1.			
2.			
3.			
4.			

Забележки:

1. Колони 1-3 се попълват от Изпълнителя при планиране на тестовете за приемане на системата.

2. Колона 4 се попълва от представители на Възложителя при провеждане на тестването.

Образецът ще бъде разработен в окончателен вид при планиране на внедряването и ще бъде представен на Възложителя за одобрение като част от Плана за внедряване.

✓ ***Образец на* протокол *за внедряване***

Най-общо протоколът ще съдържа информация за:

- Възложител;
- Място на изпълнение на дейностите по внедряване;
- Представител(и) на Възложителя, присъствал(и) и осъществил(и) контрол на изпълнение на дейностите по внедряване;
- Експерт(и), извършил дейностите по внедряване;
- Период на изпълнение;
- Извършени действия – по процедурата за внедряване и допълнителни, ако е установен

Чл.36 а, ал. 3 от ЗОП

необходимост от такива;

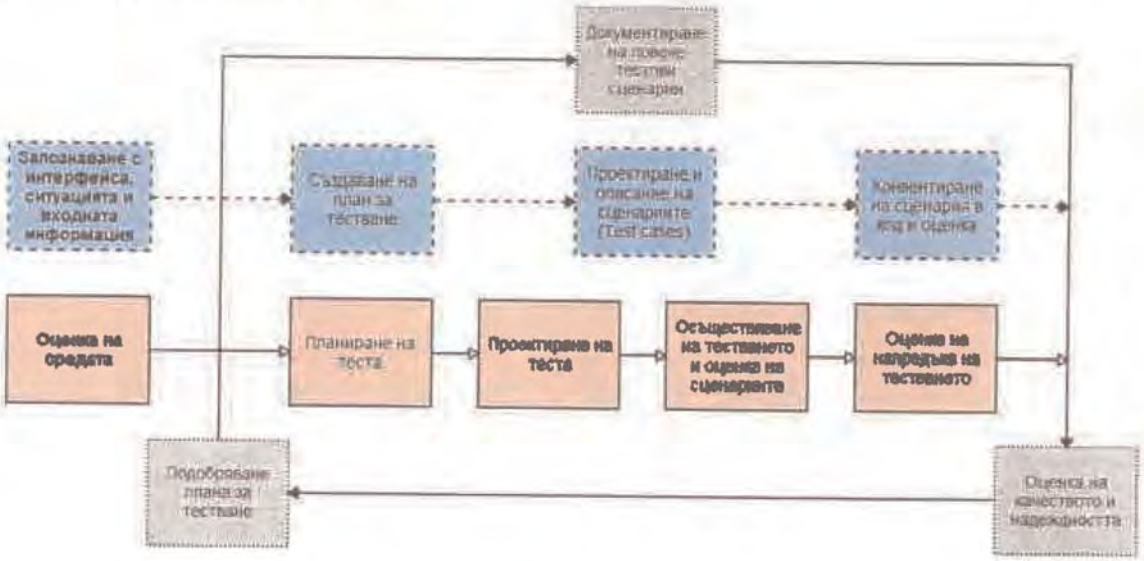
Образецът ще бъде разработен в окончателен вид при планиране на внедряването и ще бъде представен на Възложителя за одобрение като част от Плана за внедряване.

7.2.3. Методология за тестване

Изпълнителя ще проведе тестване на софтуерното решение в създадената тестова среда, с цел да потвърди, че разработеното решение отговаря на функционалните и нефункционалните изисквания. Това се постига чрез осъществяване на следните подцели на тестването:

- Откриване на всички грешки в кода, които екипът трябва отстранява;
- Откриване на грешки при дизайна;
- Откриване на повреди от неочаквано потребителско поведение;
- Тестване на всички елементи на решението.

Разработения софтуер ще се тества обстойно, за да се провери дали покрива изискванията на Възложителя. На следващата диаграма е показан процеса на тестване, който екипът на Изпълнителя ще следва:



Фигура 26 Процес на тестване

Процесът на тестване съпътства изпълнението на проекта и се състои от етапите: Планиране, Анализ и проектиране, Реализация и изпълнение, Анализ на резултатите и Заключителни дейности.

7.2.3.1. Тестов план

Настоящият тестов план е предварителен и ще бъде надграден във фазата на анализ.

Етапи на тестване:

Чл.36 а, ал. 3 от ЗОП

1) Планиране

В етапа на Планиране се определят целите, използваните техники и методология за тестване, извършва се планиране и разпределение на ресурсите, изготвяне на график за провеждане на тестването, подготовка и приемане на тест план. Тест планът включва кратко описание на типовете тестване, базирани на анализа на изискванията, описание на различните тестови среди, структура на тест екипа, времевата рамка на тестовите задачи. Изборът на подходяща методология за тестване се базира главно на определяне на основните модули, под модули и компоненти на програмната система и идентифициране на критичните точки за бизнеса и отделните групи потребители на системата.

2) Анализ и проектиране

В етапа на Анализ и проектиране се определя последователността на тестовете и изискванията към тестовата среда, проектиране на тестовете и подготовка на тестови данни (валидни и невалидни).

Проектирането на тестове включва:

- Определяне на групите свойства (features) на програмната система;
- Определяне на основните части и подчасти на програмната система с цел по-лесно проектиране на тестовете чрез разделяне на множества, ориентирани към съставните части;
- Определяне на критичните точки за бизнес процесите, реализирани в програмната система;
- Определяне на типичните ежедневни сценарии за работа на различните групи потребители на програмната система;
- Дефиниране на критичните свойства (critical features), които трябва да бъдат тествани многократно през процеса на разработка;
- Дефиниране на задължителните свойства (required features), които трябва да бъдат тествани на отделни фази през процеса на разработка;
- Дефиниране на допълнителните свойства, подпомагащи процесите в програмната система (additional features), които могат да бъдат тествани в зависимост от времето и ресурсите;
- Определянето на критериите за приемане на програмната система (acceptance criteria).

След изготвянето им те се обсъждат с Възложителя. На базата на договорените критерии за приемане на системата тест екипът подготвя План за провеждане на приемателни тестове на системата. Този план бива съгласуван и одобрен от Възложителя и след това става база за проектиране на приемателните тестове (acceptance testing).

Подходът при проектиране и подготовка на тестови случаи е тясно обвързан с изискванията към системата и се изпълнява в следната последователност:

- Определяне на основните части и подчасти на програмната система – Така се постига по-гъвкаво и ефикасно проследяване на тестовете за функционалното покритие на свойствата на програмната система (т.н. functional coverage).

- Идентифициране на критичните точки за бизнеса и отделните групи потребители на програмната система – По този начин се определят критичните точки за бизнес процесите, реализирани в програмната система (т.н. business critical points), както и типичните ежедневни сценарии за работа на различните групи потребители на програмната система (т.н. everyday business scenario).
- Определяне на групите свойства на програмната система (critical, required additional features) – Това значително подпомага процеса на разработка и тестване.

Избор на подходящи техники за проектиране на тестовете – Тази дейност включва преценка за спецификата на програмната система: дали е публична или критична откъм сигурността; доколко е сложна, комплексна или обикновена; има ли специфика във входните тестови данни; кои от избраните техники за проектиране на тестовете тест екипът владее добре.
Всички изготвени тестови сценарии подлежат на одобрение от оторизиран представител на Възложителя като след това стават база за проектиране на тестовете.

3) Реализация и изпълнение

Изпълнителя ще подготви тестова среда на сървъри предоставени от Възложителя.

Етапът на Реализация включва избор на тестове, генериране на тестови данни, изготвяне на тестовете и реалното им изпълнение. Реализацията на тестването включва:

- Избор на тест;
- Уточняване на основен и алтернативен начин на изпълнение, както и типичните изключения;
- Създаване на тест процедури (валидни комбинации на тестовете с подходящи тестови данни);
- Изпълнение на одобрените тест процедури.

4) Анализ на резултатите

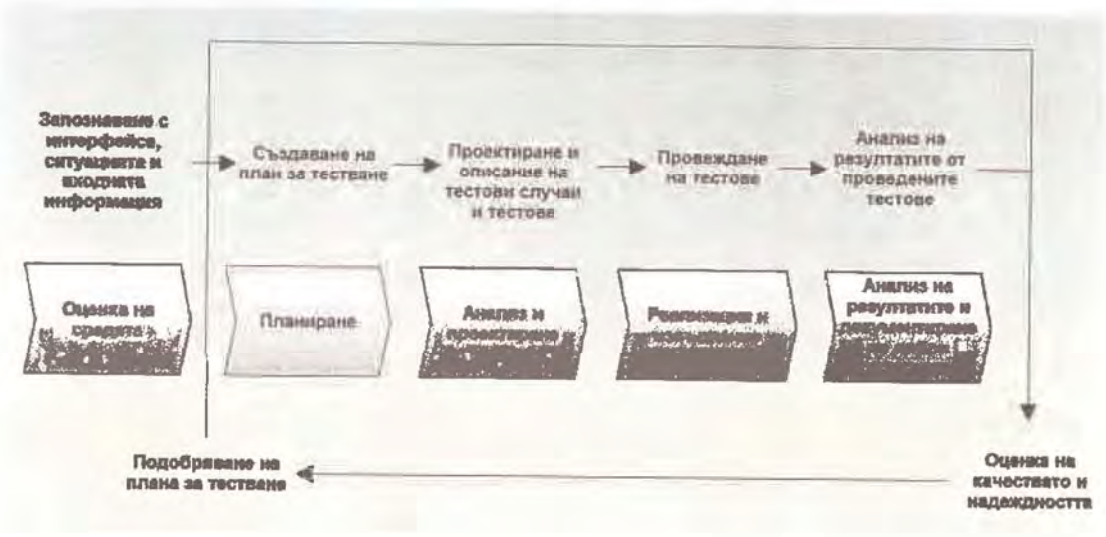
Етапът на Анализ на резултатите се състои от отчитане на получените резултати в избрания формат и проверка на условията за завършване на тестовете.

5) Заключителни дейности

Заключителните дейности обхващат изготвянето на обобщени справки, описание на добрите практики, оценка на проекта с цел подобряване на фирмените тестови процеси, архивиране на материалите.

Обобщените дейности по реализиране на тестовия процес са схематично представени на следната фигура:

Чл.36 а, ал. 3 от ЗОП



Фигура 27 Обобщени дейности по реализиране на тестовия процес

Съществуват четири основни понятия, свързани с подготовката на тестовите случай, реализацията на тестването и изграждането на тестовата среда:

Тестов случай (Test case)

Тестовият случай има за цел да тества поведението на определен модул (клас) в дадена ситуация. Обикновено се реализира чрез тестова функция. Всеки тест поставя тествания обект в подходящо за теста състояние, след което следи поведението и резултатите му в тестваната ситуация. Тестовите случаи извършват същинското тестване и трябва да са независими един от друг и от реда им на изпълнение;

Тестово множество (Test suite)

Тестовото множество се състои от набор от тестови случаи, които са логически свързани. Има за цел да тества поведението на определен модул (клас) в различни ситуации. Показва кои тестове са логически свързани. Реализира се чрез тестов клас, който има достъп до всички данни на тествания. Всеки тест поставя тествания обект в подходящо за теста състояние и следи поведението и резултатите му.

Тестова база (Test fixture)

Тестова база представлява обвивка на тестови множества. Има за цел да създаде необходимите условия за провеждане на тестовете. Отговорен е за инициализиращите и завършващите действия, както и за същинското изпълнение на тестовете. Отново, тестовете трябва да са независими един от друг и от реда им на изпълнение. Реализира се чрез йерархия от тестови класове. В базовите класове се дефинират общите операции, свързани с тестването (инициализация и завършване на теста). Те се предефинират в наследниците (тестови множества), ако е необходимо. Всеки тест поставя тествания обект в подходящо за теста състояние, инициализира го по подходящ начин, провежда теста и разрушава тествания обект. Изключително важно е, тестваната система (компонент) да възстанови първоначалното си състояние след завършване на теста.

Тестова среда (Test harness)

Тестовата среда съдържа тестовата база и предоставя условия за създаване, добавяне и изтриване на тестови множества, както и за провеждане на съответните им тестове. Разполага със средства за контрол на тестването, запазване и анализ на резултатите.

Като добри практики при подготовката на тестови случаи може да посочим:

- Идентифициране на основните функционалности и идентифициране на ключовите функционалности за бизнес процеса;
- Идентифициране на основните части и подчасти на програмната система с цел по-лесно проектиране на тестовете чрез разделяне на множества, ориентирани към отделните модули;
- Идентифициране на основните групи тестове и тестови сценарии за итерация със системата;
- Идентифициране на критичните функционалности (critical features), които трябва да бъдат тествани многократно през процеса на разработка;
- Идентифициране на задължителните функционалности (required features), които трябва да бъдат тествани на отделни фази през процеса на разработка;
- Идентифициране на допълнителни функционалности, подпомагащи процесите в програмната система (additional features), които могат да бъдат тествани в зависимост от времето и ресурсите;
- Документиране на тестовите сценарии;
- Използване на различни графични инструменти за описание на тестовите сценарии и инструменти за създаване на обвързаност между различните тестови сценарии и обвързаност между тестови сценарии и системна функционалност;
- Подготовка на тестови данни (валидни и невалидни);
- Определяне на условията за провеждане на тестовите сценарии;
- Определя последователността на тестовете и изискванията към тестовата среда, проектиране на тестовете и подготовка на тестови данни
- Определяне на критериите (изискванията към резултата от проведения тест) за преминаване/непреминаване на всеки тест.

Видове тестове залегнали в тестовия план

- Единица тестване – изпълняван се за най-малките тестови софтуерни единици – класове и методи;
- Компонентно тестване – изпълнява се за индивидуални компоненти и модули за да увери, че те коректно реализират бизнес функционалността;
- Системно тестване – след като компонентите и модулите са обединени системата се тества като цяло:
    - Функционално тестване;

- Тестване на потребителския интерфейс;
- Тестване на интерфейсите с останалите системи;
- Тестване на производителността;
- Тестване на сигурността и контрола на достъпа;
- Тестване на възстановяемостта на системата след срив;
- Тестване конфигурацията;
- Регресивно тестване.

Цели на тестването

- Цели на Единица (Unit) тестването - проверка на единицата (клас или метод) според Дизайн модела и Модела на Имплементацията. Проверка на правилната обработка на въведените данни и получаване на очакваните резултати от всяка единица. Тестват се класовете с ключова функционалност;
- Цели на компонентното тестване - целите на компонентното тестване са изпълними прототипи, като резултата на всяка итерация. Функционалността на компонентите се тества и за тяхното съответствие с бизнес изискванията и стандартите;
- Цели на системното тестване.
  - Функционално тестване - основна цел на Функционалното тестване е Модела на потребителските случаи. Разработват се тестови случаи на основата на този модел като стремежът е да се покрият голямо количество случаи с различни комбинации от входно- изходни данни. Тези тестови случаи могат да служат и като метрики за напредъка на проекта;
  - Тестване на потребителския интерфейс - проверка за наличието на всички полета от формите и проверка за дължината и типовете на данните в тези полета; проверка, дали интерфейса съдържа всички списъци от предефинирани стойности, специфицирани в документацията; проверка на правилната навигация между екраните; проверка за поведението на интерфейса при въвеждане на невалидни и валидни данни; проверка на изгледа на интерфейса;
  - Тестване на производителността - проверка времето за отговор; проверка на времето на отговор за оперативните справки; проверка на времето на разпространение на данните;
  - Сигурност и тестване контрола на достъп - защита на данните при трансфера им до АМ; защита на данните при разпространението им; защита от неоторизиран достъп да системната функционалност; валидация на различните нива на достъпа на данни; валидиране на журналите;
  - Тестване при възстановяване след срив - тестване на времето за възстановяване след срив; проверка на необработените съобщения по време на срив;
  - Тестване на комуникационната инфраструктура между модулите и връзката с

Чл.36 а, ал. 3 от ЗОП

външните потребители;

- Регресионно тестване - тестване коректното функциониране на системата след промени, когато е създадена нова версия и нови характеристики са прибавени или са поправени съществени дефекти.

Описание

- Единица тестване - най-ниските нива на единиците се тестват първи и тогава се използват за тестване на по-високите нива. Операцията се повтаря, докато не се стигне до най-горните нива.
  За тестване на всяка единица:

Входен критерий	Веднага след имплементацията на ключова единица, тя трябва да бъде тествана от разработчика
Цел на теста	Да се осигури правилна навигация, вход на данните, обработката им и получаване на очакваните резултати
Техника	Изпълнение на всяка функция, използвайки валидни и невалидни данни. Необходимите съобщения за грешки се генерират, при вход на невалидни данни.
Необходими инструменти	Инструмент за автоматизирано тестване Инструмент за проверка на кода Инструмент за следене на грешките
Критерии за завършване	Всички планирани тестове са изпълнени. Откритите дефекти са коригирани.

- Компонентно тестване - тестването на компонентите е подобно на системното функционално тестване, свързано е с реализацията на потребителските случаи;
- Системно тестване

✓ Функционално тестване - базирано е на метода на „черната кутия", въвеждат се данни и се следи изхода:

Входен критерий	След приключване на Модела на потребителските случаи, дефиниране на потребителския графичен интерфейс. Изпълним прототип съществува.
Цел на теста	Да се осигури правилна навигация, вход на данните, обработката им и получаване на очакваните резултати. Да се увери, че цялата функционалност на потребителските случаи е представена и стандартите са спазени.

Чл.36 а, ал. 3 от ЗОП

Техника	Изпълнение на сценарий на потребителски случай с валидни и невалидни данни за да се увери, че: – Очакваните резултати се получават, при вход от валидни данни; – Необходимата грешка или съобщение се визуализират при невалиден вход; – Всяко бизнес правило е правилно приложено; – Всички стандарти са спазени.
Необходими инструменти	Автоматизиран инструмент за тестови скриптове; Инструменти за генериране на данни; Средство за следене на дефектите.
Критерии за завършване	Всички ключови потребителски случаи са тествани; Всички изисквания на приложените стандарти са спазени; Всички критични дефекти са отстранени.

✓ Тестване на потребителския интерфейс:

Входен критерий	Изискванията за потребителския интерфейс са дефинирани и съществува прототип
Цел на теста	Проверка чрез навигация по приложението дали правилно отразява бизнес функциите и изискванията, включително от прозорец към прозорец, от поле към поле. Проверка, че обектите и характеристиките на прозорците, такива като менюта, размери, статус, фокус отговарят на стандартите.
Техника	Създаване на тестове за всеки прозорец за проверка на неговите обекти. Създаване на тестове за проверка поведението на екраните при вход на валидни и невалидни данни.
Необходими инструменти	Автоматизиран инструмент за тестови скриптове; Инструменти за генериране на данни; Средство за следене на дефектите.
Критерии за завършване	Всеки екран отговоря на специфицираните софтуерни стандарти.

✓ Тестване на производителността - целта му е да провери изискванията за производителността до каква степен са удовлетворени. Изпълнява се многократно като се променя нивото на натовареност на системата. Изпълняват се след системните тестове във времето, на преди тестовете за конфигурацията и тестовете при срив на системата:

Цел на теста	Валидиране времето за отговор на системата при нормални условия.
Техника	Използват се наборите от тестови случаи разработени при функционалното тестване. Тестовете вървят на една машина и се повтарят за много клиенти.
Необходими инструменти	Автоматизиран инструмент за тестови скриптове; Инструменти за генериране на данни; Средство за следене на дефектите.
Критерии за завършване	Тестовете са изпълняват без грешки и времето за отговор е в рамките на границите зададени в допълнителните изисквания.

- ✓ Стрес тестове:

Цел на теста	Валидиране времето за отговор на системата при условие, че максималния брой потребители извършват действия със системата. Валидиране времето за отговор на системата при условие, че много потребители модифицират едни и същи данни.
Техника	Използват се наборите от тестови случаи разработени при функционалното тестване. Тестовете вървят на една машина и се повтарят за максимален брой клиент.
Необходими инструменти	Автоматизиран инструмент за тестови скриптове; Инструменти за генериране на данни; Средство за следене на дефектите.
Критерии за завършване	Тестовете са изпълняват без грешки и времето за отговор е в рамките на границите зададени в допълнителните изисквания.

- ✓ Сигурност и тестване на контрола за достъп - фокусира се към две основни области на сигурността; сигурност на ниво приложение, включваща рестриктивен достъп до бизнес функциите; сигурност на ниво система, включваща проверка на потребителите:

Входен критерий	Стартира при наличието на прототип с вграден контрол на достъпа
Цел на теста	Сигурност на приложението: Проверка на операциите – Създаване, Четене, Редактиране, Изтриване в зависимост от правата на потребителите и техните роли. Системна сигурност: Достъп само за автентикирани потребители.

Чл.36 а, ал. 3 от ЗОП

Техника	Идентифициране на всеки тип потребител и определяне да видовете разрешени функции. Създаване на тестове за проверка. Промяна на типа на потребителя и повторно изпълнение на тестовете.
Необходими инструменти	Автоматизиран инструмент за тестови скриптове. Инструменти за генериране на данни. Средство за следене на дефектите.
Критерии за завършване	Функционалните тестове преминават без грешка, а при неавтентикиран или неоторизиран опит за достъп се извежда съответното съобщение.

✓ Тестване на системата след срив и за възможности за възстановяване:

Входен критерий	След успешно преминаване на системните тестове.
Цел на теста	Проверка процесите на възстановяване – автоматизирани или ръчни и състояние на системата след срив и последващо възстановяване.
Техника	Използва се набора то тестови случаи за функционално тестване, като се пускат след: Спиране на тока на клиентската станция; Спиране на тока на сървъра на приложението; Спиране на тока на сървъра на базата данни; Прекратяване на мрежовата връзка.
Необходими инструменти	Същите инструменти като при Функционално тестване.
Критерии за завършване	След задействане на процедурата по възстановяване системата трябва да е в правилното състояние.

✓ Тестване на конфигурацията:

Входен критерий	След успешно преминаване на системните тестове.
Цел на теста	Валидиране и проверка , че функциите на приложението се изпълняват правилно в средата на конфигурацията.

Чл.36 а, ал. 3 от ЗОП

Техника	Използване на системни скриптове. Отваряне, затваряне на различни приложения по време на тестовете или преди започването им.
Необходими инструменти	Същите инструменти като при Функционално тестване.
Критерии за завършване	За всяка конфигурация тестовете преминават успешно с необходимите резултати.

- ✓ Регресионно тестване - регресионното тестване не е отделен тип тестване. То е повторение на тестовете, с цел проверка коректната работа на системата след като в нея са въведени промени:

Входен критерий	Нова версия на системата след като в нея са направени промени.
Цел на теста	Проверка поведението на системата след внедряване на промените.
Техника	Използват се вече разработените тестови случаи, набори от случаи, данни и скриптове за повторно тестове на системата. Възможна промяна и на тестовите случаи или данни в зависимост от вида на направените промени.
Необходими инструменти	Същите инструменти като при Функционално тестване.
Критерии за завършване	Същите критерии като изпълняваните тестове.

Контекст за извършване на тестовете

Източниците на информация за провеждане на тестовете:

- План за разработка на софтуерния продукт – включва очакваните дати за провеждане на тестовете, както и резултатите, които следва да се представят от тестването.
- Спецификация на изискванията – включва нефункционалните изисквания към системата.
- Визия – Във Визията са заложени основни нужди и характеристики и продукта
- Модел на дейностите - Моделът на дейностите включва Бизнес модел и Модел на потребителските случаи. Моделът на потребителските случаи представя предвижданите функции и среда на системата и отразява нейните функционални спецификации, а Бизнес моделът описва начина, по който се изпълняват бизнес потребителски случаи.
- Документ Софтуерна архитектура - представя комплексен архитектурен изглед на системата, използвайки за целта редица различни архитектурни разрези, показващи отделни нейни аспекти.

Чл.36 а, ал. 3 от ЗОП

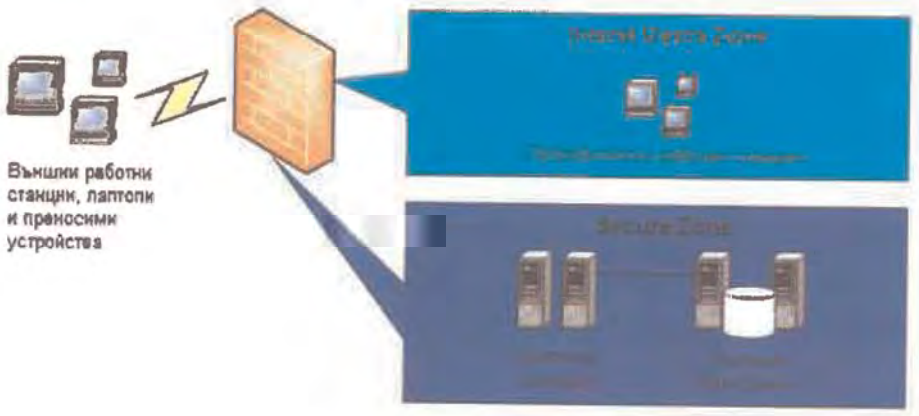
- Дизайн модел – обектен модел, който описва реализацията на потребителските случаи и служи за извеждане на Модела на имплементацията и неговия програмен код.
- Модел на данните – Моделът на данните е подмножество на имплементационния модел, което описва логическия и физически вид на постоянните (персистентни) данни в системата.
- Модел на имплементацията - Моделът на имплементацията събира на едно място компонентите и съдържащите ги имплементационни подсистеми. Компонентите включват както тези, които подлежат на предаване като отчетни резултати (например изпълнимите компоненти), така и тези, от които се извеждат предаваните компоненти (например файлове с програмен код).

7.2.4. Описание на архитектурата и подхода за реализация на предлаганото софтуерно решение

7.2.4.1. Архитектура

Системата ще бъде проектирана като централизирана уеб базирана система, комбинираща в себе си функционалност за първоначален импорт на данните, механизми за валидиране и изчистване на данните, трансформация на данните в релационен модел, използван за крайните регистри и многомерен модел, използван за представяне на OLAP кубове и съотвената функционалност за аналитична обработка на данните от крайните потребители (Business intelligence). За да се постигнат изискванията за модулност, мащабируемост и гъвкавост софтуерът ще бъде разделен 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.

7.2.4.1.1. Физическа архитектура на системата



Фигура 28 Физическа архитектура на системата

Чл.36 а, ал. 3 от ЗОП

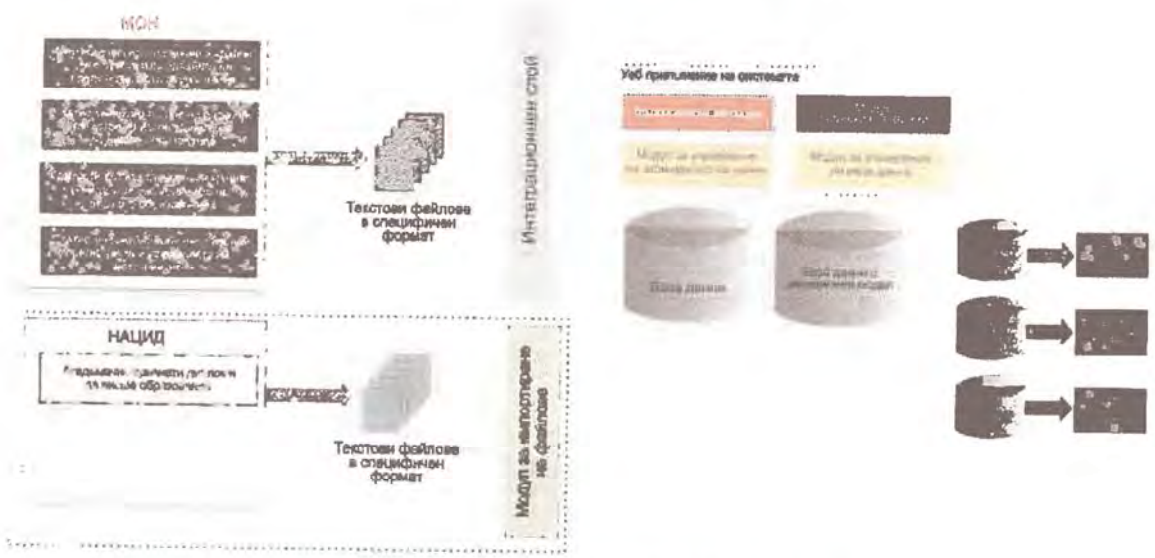
Предлаганото решение ще бъде адаптирано към съществуващата хардуерна, мрежова и приложна среда в НСИ, като в примерния вариант на физическа архитектура изложен в настоящото предложение, предлагаме да има като два основни физически компонента:

- Приложен сървър изпълняващ заявките към уеб приложението на системата, чрез който потребителите работят със системата
- Сървър за управление на базите данни на системата – описания в спецификацията Microsoft SQL Server Standard Edition с реализирани три бази данни – staging, релационен модел на регистрите и база хранилище за данни (Analysis services).

Предлаганата физическа архитектура, ще бъде реализирана чрез създаване на виртуални сървъри чрез предоставения от Възложителя лиценз за платформата за виртуализиация VMWare, на физическия хардуеърен ресурс предоставен от Възложнителя

Съображенията за максимално добро обособяване и осигуряване на информационна сигурност предопределят препоръчително отделяне на сървърите в т.нар. DMZ – зона от мрежата, която е изолирана от останалите части на вътрешната мрежа на НСИ отделена от външната среда с външна защитна стена, но и с вътрешна защитна стена, така че евентуален пробив да не доведе до риск за сървъра с приложението на системата или сървъра съхраняващ неговите бази данни.

7.2.4.1.2. Предлагана логическа архитектура



Фигура 29 Логическа архитектура на предлаганото решение

Описаните в техническото задание изисквания специфицират система за извличане, зареждане, валидиране и обработка, трансформиране и анализиране на данни от външни системи - източници на информация, което предполага реализация на решението като OLAP (Online analytical processing) система, ориентирана към трансформиране на готови масиви

Чл.36 а, ал. 3 от ЗОП

данни във формат максимално оптимизиран за бъдеще статистическа и аналитична обработка, а не към представяне на функционалност за първично въвеждане на транзакционни данни (Online Transaction Processing).

Реализацията на подобно решение се базира на реализация на няколко основни взаимодействащи бази данни и ETL процес, при който данните се обработват в различни структури, докато бъдат трансформирани в многомерен дименсионален модел, оптимизиран за статистическа и аналитична обработка на агрегирани данни. След зареждане на резултатния многомерен модел данните биват представени под формата на OLAP кубове, които от своя страна се използват в съответни системни справки, имплементирани и изпълнявани чрез модул за справки и анализи, представящ информацията на крайните потребители на системата.

Предлаганото решение структурира в системата следните бази дани, които са интегрирани, но се различават по своята структура

- Staging area – база данни, съхраняваща т.нар. сурови данни – тук данните се зареждат в първоначалния си вид, такива каквито са подадени от съответните източници на информация. Моделът на данни на тази база данни максимално се приближава да формата на данни предоставян от външните системи. Данните постъпват през интерфейса на модула за импорт на данни на системата, чрез импорт и обработка на текстови файлове по предварително дефиниран формат или чрез уеб услуга позволяваща междусистемна обработка на файлове в същия предефиниран формат. В тази база данни се съхраняват файловете източници на информация в оригиналния им вид, като за всеки запис от файла се запазва връзката към файла от който е постъпил този запис. Това позволява правилно управление на операциите по отмяна и повторно изпълнение на зареждането на данни.
- Релационна база данни – импортираните данни в staging area базата се валидират, редактират и при нужда се зареждат многократно докато отговорят на изискванията за валидация и изчистване на данните, след което се импортират в релационния модел за крайните регистри, който е реализиран с нужните ограничения и връзки и е обвързан с класификационните мета данни на Възложителя, така че да позволи дименсионален анализ на агрегирано ниво. Отново се запазва и връзка към записа от staging базата, който е свързан със запис от крайния регистър, който да се използва при регулярно или инизиирано от потребителя зареждане на данни от staging към релационния модел.
- Многомерна база данни – данните от релационния модел се трансформират чрез възможностите на SQL Server Analysis services в многомерен модел за хранилище данни (data warehouse), чрез реализиране на OLAP кубове в които се използват различни изброими стойности (measures), които се прекалкулират за всички стойности от класифициращите ги измерения (dimensions).

Постъпковата обработка на входните за системата дани от staging базата, към релационната база, а от там в многомерната база се имплементира като един цялостен логически свързан процес, т.нар. ETL – процес имплементиращ логиката за извличане (Extract) на входните данни от файлове, трансформация (Transform) и зареждане на

Чл.36 а, ал. 3 от ЗОП

данните в многомерния модел (Load). За целите на реализацията на този процес ще се използват вграденото средство на Microsoft SQL Server за реализация на ETL процеси – SQL Server Integration Services.

Работата на потребителите по управление на данните на системата, както и консумирането на представяната резултатна информация няма да бъде свързана с директна интеракция с така описаните бази данни и ETL процеса, реализиращ преноса на данни между тях, а чрез съответните основни модули на уеб приложението на системата:

- Модул за импортиране на данни – обслужва първоначалното зареждане на данни, включително, включвайки нужните предефинирани формати за данни, валидационна логика за първоначална проверка за конситентност на входните данни, възможности за визуализация на проблеми с данните при неуспешен импорт или резултатни записи при успешна операция.
- Модул за управление на данни – предлага изглед към данните от staging базата, позволявайки редакция на конкретни записи от данните, повторно зареждане или други актуализации чрез потребителския интерфейс на системата.
- Модул за управление на мета данни – поддържа определените от техническото задание класификационни мета данни поддържани от Възложителя, позволявайки тяхното версиониране, актуализация и позволяващ преизчисляване на данните на релационния модел и многомерния модел при промяна в мета данните.
- Модул за администриране – служи за цялостно администриране на конфигурацията, настройките, достъпа и отворените номенклатури в системата
- Модул за справки и анализи – представя информацията от OLAP кубовете на многомерната база данни под формата на справки, отчети и dashboards, които потребителите могат да използват и анализират.

7.2.4.2. 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

Чл.36 а, ал. 3 от ЗОП

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.

7.2.4.2.1. 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 фиксирано в програмния код

Чл.36 а, ал. 3 от ЗОП

логиката за изпълнение на процеса по обработка на данни, да се реализира процеса изцяло със средствата на СУБД, чрез съхранени SQL процедури, или реализация чрез специфичен продукт за реализация на ETL процеси и интеграция на данни. 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 интегриран в предлаганото от нас СУБД, за което НСИ притежава лиценз (Microsoft SQL Server), а именно -SQL Server Integration Services.

Втория избор при реализацията на ETL процеса е в избора на начина за извършване на трансформацията на данните – при подхода ETL, трансформацията се извършва преди зареждането на данни в многомерната база данни (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 SQL Server Integration Services като средство за дефиниране на ETL позволява дефинирането на параметризиран и конфигуриран процес на реализация на 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.

7.2.4.2.2. Предлаган процес на реализация на ETL

Чл.36 а, ал. 3 от ЗОП

Фигура 30 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 staging базата на системата (модул за импорт, модул за управление на входните данни), като събраните данни ще бъдат прехвърлени след одобряване на резултата от процеса в релационната база данни с крайните регистри, като се запазят връзките между файла първоизточник и записите за него в staging и релационната база.

2. След импортирането на файл от потребителския интерфейс или 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 staging базата данни, в рамките на следващата стъпка на ETL процеса. Оттам се преминава към зареждането им в релационната база на крайните регистри. Релационния модел на крайните регистри комбинира данните с класификационните мета данни за хранилището.

Чл.36 а, ал. 3 от ЗОП

4. Стъпка на валидиране и изчистване на данните – в тази стъпка се извършват операции свързани с осигуряване на интегритета на данните и тяхната коректност. 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 релационната база данни, съхранени в типичен релационен модел, ще бъдат заредени в модела за данни на многомерната база данни, при което данните от релационния модел се трансформират с помощта на OLAP средството на избраното СУБД – SQL Server Analysis Services, в което се описва откъде в модела на релационната база се зарежда всеки OLAP куб, в модела на данни за многомерната база данни (т.нар. Logical Data Map). В многомерната база данни се поддържат кубовете със съответните факт таблици и димeнсии. Впоследствие OLAP сървъра ще зареди в паметта нужните кубове от предварително агрегирани данни, така че извличането на информация от тях при справки и анализи да става без допълнителни обръщения към входните данни, а само върху агрегираните прекалкулирани данни в кубовете.

7.2.4.2.3. Инструмент за реализация на ETL процеса – SQL Server Integration Services

За реалзиацията на ETL процеса предлагаме SQL Server Integration Services, включени във функционалностите на SQL Server Standard Edition, които предлагат нужния инструментариум за изграждане на многомерната база данни, със зареждане на данни от релационния модел, конфигурируеми трансформации на данните, както и възможност за визуално моделиране.

Основното предназначение на SQL Server Integration Services (SSIS) е чрез т.нар. трансформационни пакети да предложи средство за извличане на данни от разнообразни източници (релационни бази и файлове в различни формати като текст, CSV, XML), трансформацията им, съобразно бизнес логиката на пакета и зареждане в бази данни. По своето същество SQL Server Integration Services (SSIS) е платформа за изграждане на решения за data integration и изграждане на ETL процес за зареждане на хранилища от данни с високо бързодействие и ниво на параметризация и лесна поддръжка.
Основните предимства на предлагания инструмент за ETL са:

Чл.36 а, ал. 3 от ЗОП

- Бърз и ефективен начин за разработка на ETL процеса, поради използването на готовите възможности на средството, както и поради наличието на визуални редактори за дефиниране на трансформациите и стъпките на процеса
- Богати възможности за настройване на изпълнението и поведението при грешки
- Оптимизирано бързодействие чрез отделяне на контролния процес от процесите на зареждане на отделни набори данни и използване на паралелизация
- Лесно преизползване на функционалността и разширяване без промяна в програмен код или SQL процедури

7.2.4.3. Подход към реализация на многомерната база данни (хранилището от данни)

В модерните тенденции за разработката на хранилища на данни (data warehouse) все по-често се утвърждава подхода към поддържане на обособени складове на данни (data marts) като част от модела на хранилището, за сметка на традиционния подход, при който справките и извличането на данни се изпълняват директно срещу единен многомерен модел на хранилището.

Предлагания подход при изграждане на складовете за данни за системата, ще се базира на това че те могат да бъдат най-добре определени като „подмножество“ или „разрез“ (често до известна степен или напълно агрегирани данни, обработвани като OLAP куб). Въпросният агрегиран куб може да бъде използван за обръщения независимо от другите елементи от модела на хранилището, като периодично да бъде обновяван от източниците на данни. Погледнато така складовете за данни могат да бъдат разглеждани, като производни или прилежащи елементи на хранилищата за данни.

При предлагания подход складовете за данни ще се изграждат успоредно с релационния модел за регистрите, които служат за източник на информация. Така в рамките на итеративното подобрение на модела на данни и качеството на наличните данни от източниците, разработката на кубовете не се налага да „изчаква“ интегрирането на въпросните данни в модела на хранилищата за данни, тъй като в начален етап може да се използва подхода ROLAP, при който OLAP кубът още не е създаден като многомерен модел в базата данни на хранилището, а трансформира MDX заявките към куба в SQL заявки към стандартна релационна база, подход поддържан от кубовете на SQL Server Analysis Services. Едва след достигането до ясен и зрял краен модел на релационната база данни и яснота за

нужните заявки към кубовете, те вече ще могат да бъдат изградени като MOLAP кубове, които се материализират в многомерен модел в хранилището на данни. Причината за този подход е, че в случая работим за регистрови данни с относително ясна структура, т.е. входните данни могат да бъдат описани в стабилен релационен модел, а всички кубове ще се ръководят от наличната информация в релационния модел, така че до неговото завършване, при друг подход всяка промяна ще трябва да се прилага и на релационния и на многомерния модел. При спазване на този принцип, даден куб от данни може да бъде тестван, интегриран в справки и подобряван преди да завърши реализацията на основния модел, след което материализиран след завършване на изграждането на цялостния модел на данни.

Причината поради която търсим максимално ранното използване на кубовете от данни от потребителите в предлагания от нас подход за развитие на хранилището от данни е желанието за интегрирането на промени наложени от потребителската обратна връзка в процеса на изграждане, а не да разглеждаме модела на хранилището за данни и релационния модел като относително статична величина.

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 ETL процесите. Хранилището за данни интегрира всички данни в общ формат и обща софтуерна среда. Всички ресурсни данни 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

Чл.36 а, ал. 3 от ЗОП

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 крайните регистри, които са включвни в релационния модел, който е източника за изграждане на многомерния модел на хранилището. От друга страна се оставя независимост при разработката на складовете и съответните кубове, 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.

7.2.4.3.1. 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, като общият набор от обекти и атрибути е специфициран още в техническото задание, а предвидените дейности по анализ и проектиране, ще дадат ценна и относително пълна информация за качеството на информация идваща от източниците на данни, които ще се използват. Дефинираните цели пред проекта са предварително определени. При наличната основа от информация, налична за анализ, смятаме, че подхода на изграждане „отгоре - надолу" с обратна връзка (top – down with feedback), ще позволи да се създаде бързо релационен модел и да се реализира на база него чрез ROLAP многомерен 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 финализира многомерния модел преди окончателното внедряване в продуктивна среда.

7.2.4.3.2. Предлаган инструмент за реализация на OLAP кубове – SQL Server Analysis Services

SQL Server Analysis Services е средството за изграждане на OLAP кубове интегрирано в СУБД Microsoft SQL Server.

SQL Server Analysis Services поддържа OLAP кубове, като за разлика от други OLAP инструменти се поддържат не един, а всички концептуални подходи за реализация на кубове, което дава максимална гъвкавост при изграждане на многомерния модел:

- ROLAP – Relational OLAP, при който данните се съхраняват в релационен модел и не се съхраняват в базата данни с многомерен модел.
- MOLAP – Multidimensional OLAP, при който данните се извличат от релационен модел и се съхраняват в многомерен моде
- HOLAP – Hybrid OLAP, при който основните данни са в релационния модел, но част от предварителните агрегации и индекси се съхраняват в многомерната база за оптимизация на бързодействието

Чл.36 а, ал. 3 от ЗОП

SQL Server Analysis Services поддържа множество алтернативи за изграждане на заявките към OLAP кубовете:

- MDX – специализиран език за заявки към многомерни кубове
- SQL – лимитирана възможност за стандартни заявки към многомерните данни
- DMX – език за заявки към модели за data mining
- LINQ – възможност за изграждане на композитни, динамично структурирани заявки към кубовете от .NET приложения, елиминираща нуждата от работа с фиксирани SQL заявки и елиминиращо нуждата от кодиране на бизнес логика в слоя на базата данни.

SQL Server Analysis Services е естествено интегрирано в предлаганата база данни, в която по идентичен начин е интегрирано и предлаганото ETL средство за реализация на системата. SQL Server Analysis Services в комбинация с базата данни на SQL Server напълно отговаря на изискванията на техническото задание за изграждане и поддържане на аналитична база данни:

- Дефиниране на многомерна база данни;
- Дефиниране на дименсии в многомерна БД;
- Дефиниране на кубове с агрегирани данни.
- Извличане на данни от релационни и нерелационни източници на данни;
- Агрегиране на извлечените данни;
- Зареждане на агрегирани данни в многомерна БД;
- Съхраняване на многомерна БД като таблици в релационна СУБД;
- Създаване на OLAP-модели;
- Наличие на графичен интерфейс за работа с OLAP-структурите;
- Притежава система за архивиране и възстановяване на данни и експортиране в ASCII файл;
- Извършване на многомерен анализ върху многомерни БД;
- Притежава набор от визуални и аналитични средства, осигуряващи възможност за избор, представяне, манипулиране, визуализация и анализ на данните;
- Притежава средство за генериране на отчети (Analysis Services се интегрира в рамките на платформата с SQL Server Reporting services);
- Притежава административно средство за управление на потребители, групи от потребители и връзки към базата данни (SQL Management Studio Express).

Чл.36 а, ал. 3 от ЗОП

Всички посочени възможности се базират на наличния лиценз притежаван от Възложителя за SQL Server Standard Edition 2016 и по нов.

7.2.4.4. Подход за изграждане на уеб приложението на системата

Уеб приложението на системата ще представлява пълнофункционално уеб приложение с архитектура MVC, предлагащо на потребителите единна точка за достъп до три основни групи функционалности на системата:

- Функционалността и модулите свързани със зареждането, валидацията, изчистването и обработка на данните (модул за импорт, модул за управление на данните)
- Модул за управление на мета данните
- Административен модул
- Функционалността за конфигуриране, изпълнение и представяне на резултата от реализираните в системата справки, отчети и dashboards – модул за справки и анализи

7.2.4.4.1. Подход при проектиране на архитектурата на уеб приложението

Отделните модули реализиращи функционалността на уеб 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.

7.2.4.4.2. Слоеве (нива) на архитектурата уеб приложението на системата

Предлаганото уеб приложение ще има архитектура която може да бъде декомпозирана на отделни слоеве (нива) комуникиращи помежду си по строго определени интерфейси. Основно предимство при този подход 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

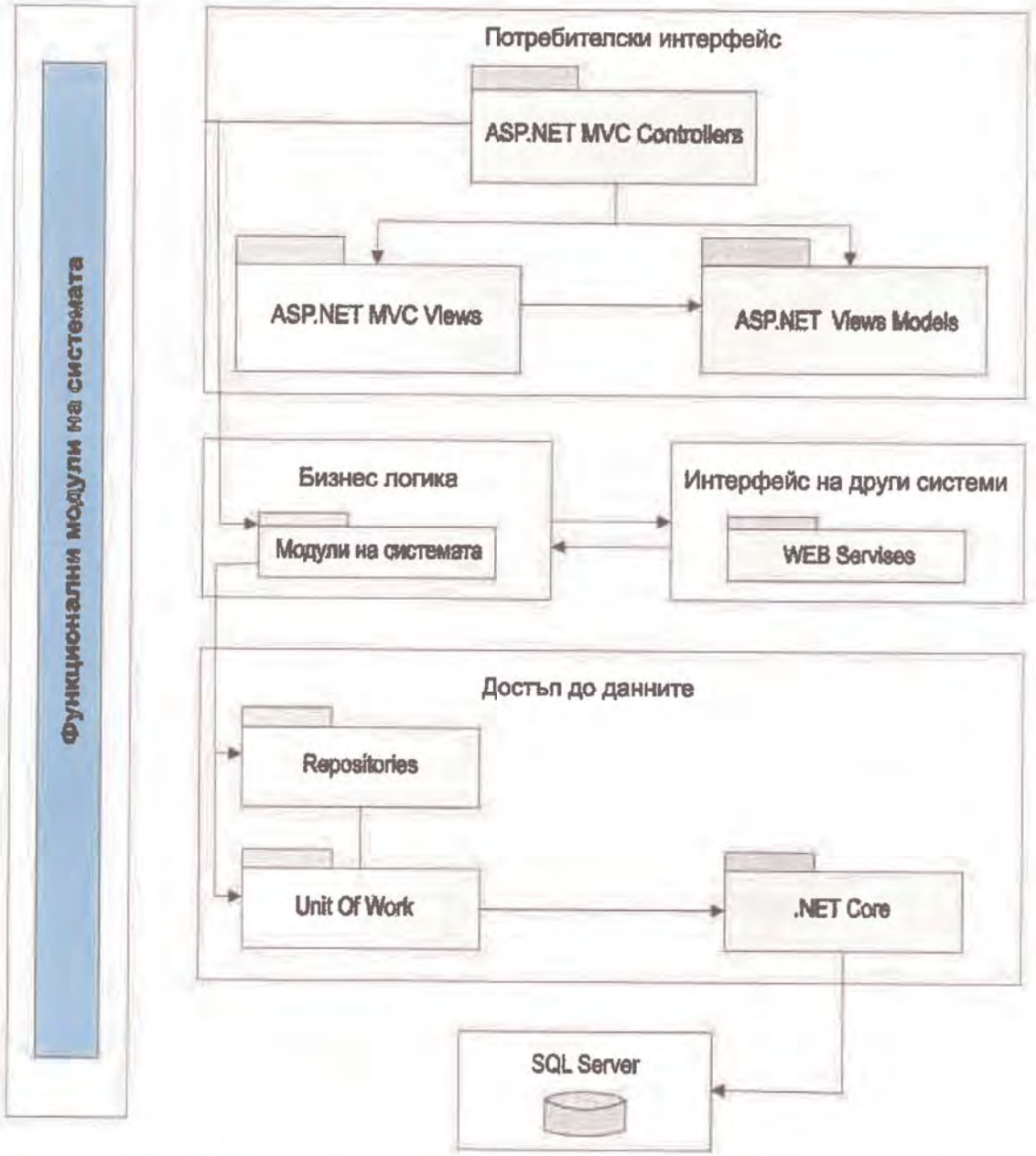
Чл.36 а, ал. 3 от ЗОП

искаме да можем да варираме независимо. Слоевете са определени така, че да групират елементите, които искаме да можем да варираме независимо. При централизираните системи доказан подход е разделянето на следните слоеве:

- Слой на базата данни;
- Слой на бизнес логиката;
- Слой на потребителския интерфейс (презентационен слой);

Всеки слой в последствие се декомпозира на отделни модули, като комуникацията между модулите се осъществява също по строго специфицирани интерфейси. Разделението на ясно разграничени слоеве и обособяването на слоя на базата данни от слоевете на бизнес логиката позволява да се осигури възможността цялостното решение да бъде съвместимо както със съществуващата инфраструктура в НСИ както и с виртуална инфраструктура, съответно върху Държавния хибриден частен облак.

Чл.36 а, ал. 3 от ЗОП



Фигура 31 Системна архитектура на реализацията на ИС – трислойна архитектура MVC

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

Чл.36 а, ал. 3 от ЗОП

реализираните в слоя за управление на данните методи. Системата за управление на съдържанието 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. 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;

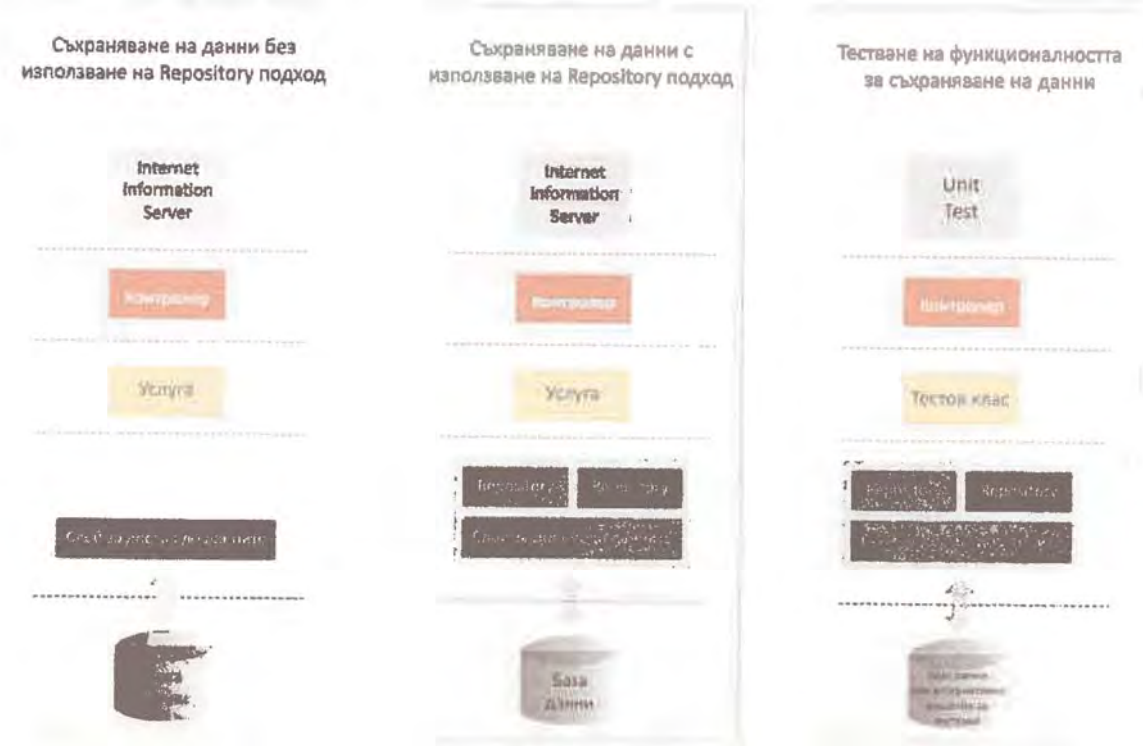
Допълнителния слой на абстракция позволява по голяма гъвкавост по отношение на физическия модел на данните – т.е. улеснява скалирането на базата данни и евентуалното му разпределяне в повече бази данни;

Entity Framework 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:

Чл.36 а, ал. 3 от ЗОП



Фигура 32 Съхраняване на данни

Слоя за достъп до данните ще съхранява електронно съдържание в релационната база данни - Microsoft SQL Server. 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.

Реализацията на слоя на бизнес логиката ще включва набор от отделни компоненти които изграждат това ниво:

- От гледна точка на по-добра сигурност и гъвкавост на приложението вътрешната и външната част на приложението ще се обособят като отделни MVC приложения, които ще дават достъп до различна част от функционалността на системата, но ще се реализират използвайки общата

Чл.36 а, ал. 3 от ЗОП

основа от модули и общ модел на данните, така че да се улесни максимално бъдещото развитие и на двете части на системата

- Бизнес логиката за въвеждане, актуализация и представяне на информацията от модулите на системата ще се реализира съгласно обектно ориентирания подход в гранулярно дефинирани независими компоненти – модули на системата, които си взаимодействат спрямо дефинирани между тях интерфейси. Модулите на системата са детайлно описани в секция „Функционални модули на информационната система“ на настоящото предложение. В допълнение ще бъде реализиран и модула за интеграция с външни системи, който ще позволява обмен на данни и изпълнение на части от функционалността на модулите на бизнес логиката при интеграция с дефинираните в техническото задание системи.

Слой на потребителския интерфейс

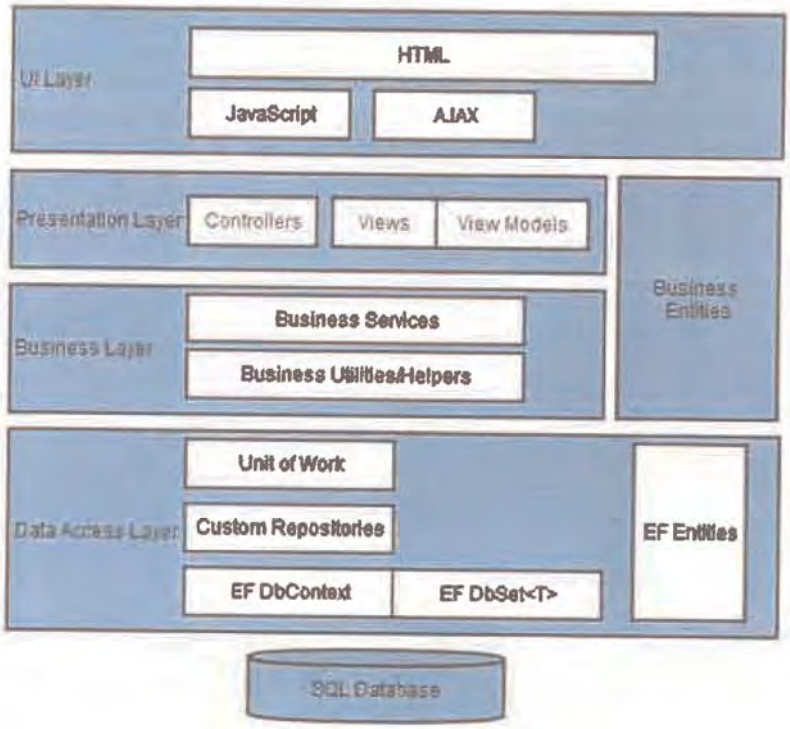
Задачата на слоя на потребителският интерфейс е да взаимодейства с потребителите на системата, така че да обслужва техните заявки и да представя необходимата им информация в определения формат. В този слой ще се организира и използва интерфейса на системата, от потребителите, за да извършват желаната от тях работа. Въведените данни в този слой се подават на слоя на бизнес логиката за последващата им обработка и съхранение в базата данни.

7.2.4.4.3. Прилагане на Model-View-Controller.

При реализацията на уеб приложенията на системата ще се използва архитектура Model – View – Controller, чрез реализацията и в избраната технологична платформа .Net Framework, а именно ASP.NET MVC. При тази архитектура контролерите на приложението 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.

Чл.36 а, ал. 3 от ЗОП



Фигура 33 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.

7.2.4.4.4. Предлаганата техническа платформа за изграждане на уеб приложението на системата – описание, обосновка и предимства

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

Чл.36 а, ал. 3 от ЗОП

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.

4.1.1. 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.

4.1.2. Бързодействие при изпълнение

Чл.36 а, ал. 3 от ЗОП

Платформата разполага със среда за изпълнение CRE (Common Runtime Engine), коятопозволява 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).

4.1.3. 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.

4.1.4. Отворени стандарти

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

4.1.5. 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.

4.1.6. 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.

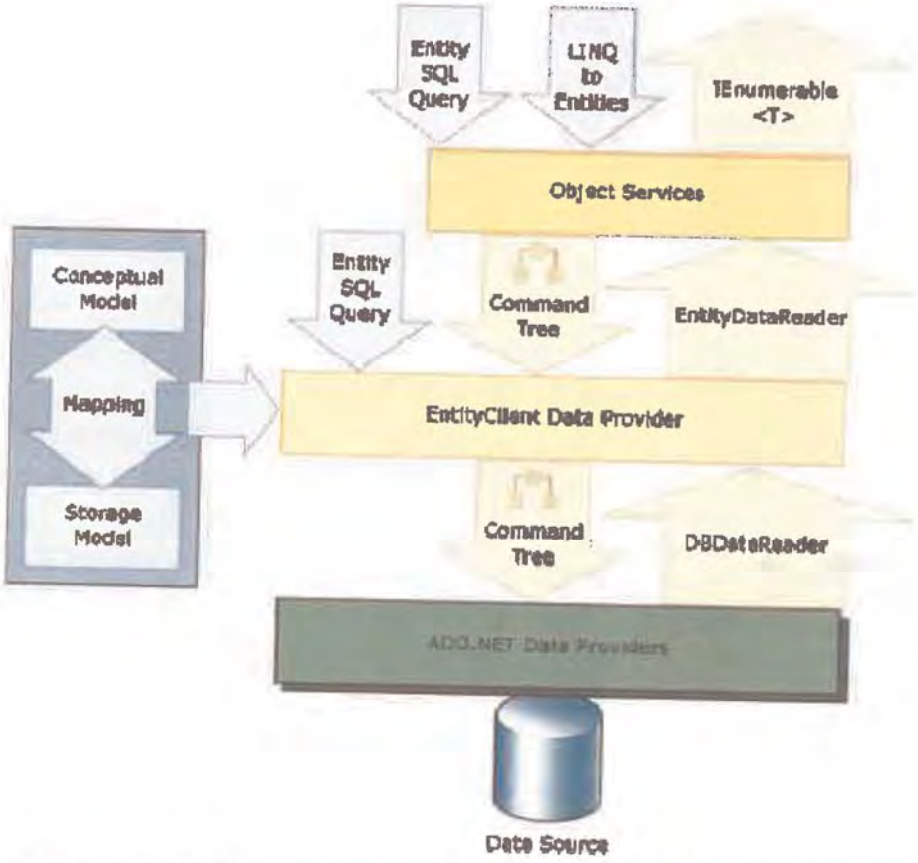
Програмният код, написан на .NET Framework се нарича управляван код, а също така и защитен код. Той се изпълнява в специална защитена среда наречена sandbox и е изолиран от хардуерната и софтуерната среда, в която работи. Това го предпазва от програмни грешки, които го правят уязвим за атаки, например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

4.1.7. 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 – 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.

Чл.36 а, ал. 3 от ЗОП



Фигура 34 Централизирано съхранение и достъп до бизнес обектите

4.1.8. 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:

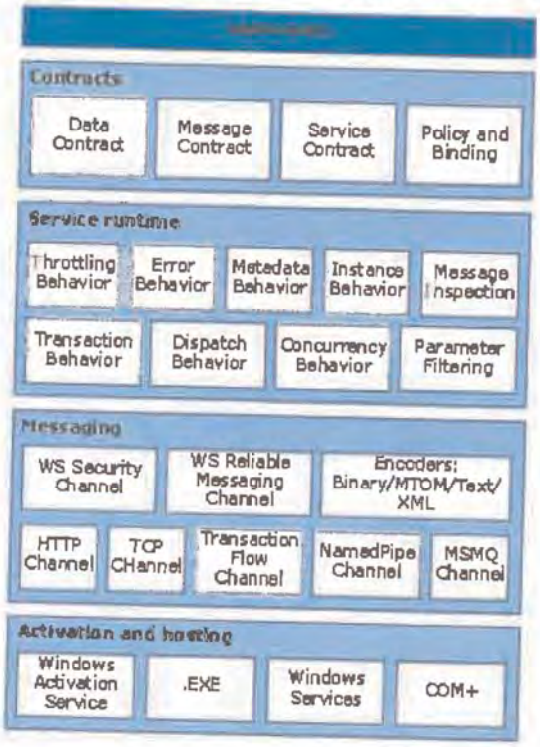
- Широка съвместимост. WCF предоставя пълна поддръжка на всички протоколи, които са необходими за изграждане на инфраструктура от Web услуги (WS), в това число SOAP (Simple Object Access Protocol) за предаване на данни и изпълнение на конкретни операции;
- Интеграция с COM/COM+;
- Широк обхват на транзакцията. 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;

Чл.36 а, ал. 3 от ЗОП

- 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.

4.1.9. Поддръжка на уеб услуги

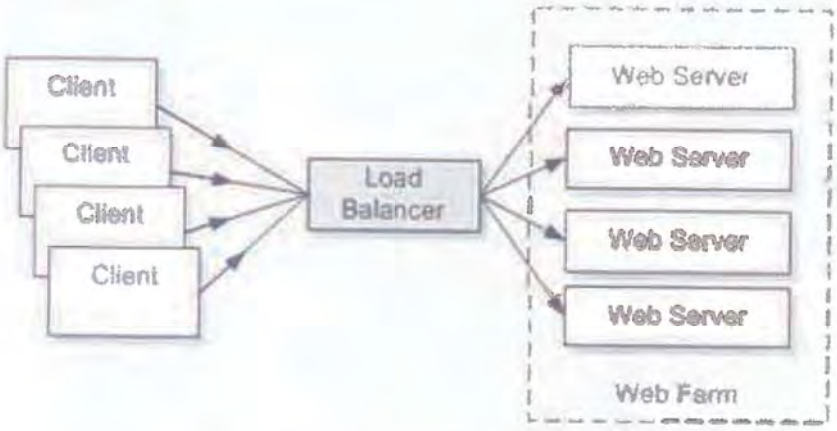
Част от Microsoft.NET е Windows Communication Foundation (WCF), която позволява да се създават интероперабилни уеб услуги. WCF покрива Web services (WS-*) стандартите, което позволява на приложенията да комуникират по платформено независим начин.



Архитектура на WCF

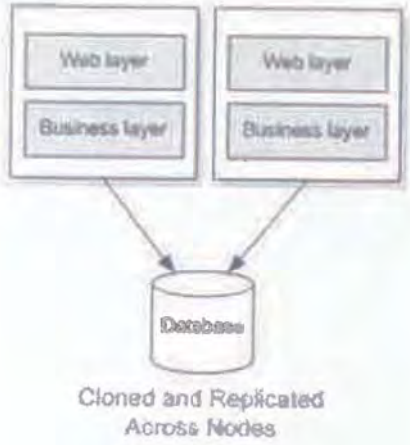
4.1.10. Скалируемост

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.



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.



Разпределени транзакции

4.1.11. 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).

4.1.12. Интеграция със средата за разработка Microsoft Visual Studio

Чл.36 а, ал. 3 от ЗОП

Чл.36 а, ал. 3 от ЗОП

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.

7.2.4.4.5. Предлагани средства (инструменти) за разработка на уеб приложението

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:

- ✓ Изпълнителят ще реализира предлаганото решение на базата на система за управление на бази данни – Microsoft SQL Server, водещ продукт в областта на комерсиалните СУБД, сертифициран в съответствие с международния стандарт ISO/IEC 15408:2005, определящ т.нар. "Common Criteria for Information Technology Security Evaluation (CC)". Предлаганото СУБД Microsoft SQL Server отговаря напълно на техническата спецификация:
  - Поддържа сървър с повече от един процесор
  - Поддържа всички стандартни релационни типове данни, а също и собствени типове за съхранение на XML данни, текст, документи, изображения, аудио и видео данни, географски векторни и растерни данни;
  - Предлага надежден failover механизъм
  - Позволява инсталиране в клъстер
  - Предоставя графичен интерфейс за администрация на управляваните бази данни
  - Поддържа изгледи, които съдържат агрегирани стойности от една и повече таблици и да предоставя механизми за прозрачно обновяване на агрегираните стойности в момент на промяна на данните в изходните таблици;
  - Предлага ефективен начин за работа с големи обеми от данни, като позволява да се поддържа логическо разделяне на физическите таблици на няколко логически, с цел бързодействие;
  - Поддържа съдържание на кирилица
  - Пълнотекстова индексация на съдържанието на файлове във формат Excel, Word, PDF, Text

Чл.36 а, ал. 3 от ЗОП

Чл.36 а, ал. 3 от ЗОП

- ✓ 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 Изпълнителя 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, VueJS 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 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.

7.2.5. Основни функционални модули на реализацията

Предлаганата архитектура на системата предполага строго спазване на принципите на обектно ориентираното проектиране и програмиране и компонентно базираната

Чл.36 а, ал. 3 от ЗОП

архитектура, която е в сърцето на обектно-ориентирания подход. От тази гледна точка системата ще реализира всички модули специфицирани в заданието на системата (категоризирани като основни модули на системата), но също така и някои допълнителни системни модули на системата, които са нужни за реализация на функционалности в основните модули, които са достатъчно генерични и могат да бъдат споделени и от други модули на системата, поради което реализираната функционалност се обособява и реализира като системен модул.

7.2.5.1. Модул "Статистическите първични и крайни регистри за изследването на предучилищното образование"

Описание на модула

Модул "Статистическите първични и крайни регистри за изследването на предучилищното образование" съдържа информация за всички предучилищни образователни институции (Детски градини), за всички записани деца, както и за персонала в тези детски градини.

Диаграма на на модела на данни

Основни информационни обекти и минимален обхват на поддържаните данни

Чл.36 а, ал. 3 от ЗОП