- Всички уебстраници (вътрешни и публично достъпни в Интернет) ще бъдат достъпни единствено и само през протокол HTTPS. Криптирането ще се базира на сигурен сертификат с валидирана идентичност (Verified Identity), позволяващ задължително прилагане на TLS 1.2, който е издаден от удостоверителен орган, разпознаван от най-често използваните браузъри (Microsoft Internet Explorer, Google Chorme, Mozilla Firefox). Идентификацията се осъществява двустранно по протокол TLS (Transport Layer Security –Сигурност на транспортния слой), версия 1.2 или по-висока, дефиниран в Препоръка RFC 5246, приета от IETF (The Internet Engineering Task Force –Целева група за Интернет инженеринг) през август 2008 г. Ежегодното преиздаване и подновяване на сертификата трябва да бъде включено като разходи и дейности в гаранционната поддръжка за целия срок на поддръжката;
- Ще бъдат извършени тестове за сигурност на всички уебстраници, като минимум чрез автоматизираните средства на SSL Labs за изпитване на сървърна сигурност (https://www.ssllabs.com/ssltest/). За нуждите на автентикация с КЕП трябва да се предвиди имплементирането на обратен прокси сървър (Reverse Proxy) с балансиране на натоварването, който да препраща клиентските сертификати към вътрешните приложни сървъри с нестандартно поле (дефинирано в процеса на разработка на Системата) в HTTP Header-а. Схемата за проксиране на заявките трябва да бъде защитена от Spoofing;
- Като временна мярка за съвместимост настройките на уебсървърите и Reverse Proxy сървърите ще бъдат балансирани така, че Системата да позволява използване и на клиентски браузъри, поддържащи по-стария протокол TLS 1.1. Това изключение от общите изисквания за информационна сигурност не се прилага за достъпа на служебни потребители от държавната администрация и доставчици на обществени услуги, които имат служебен достъп до ресурси на Системата;
- При разгръщането на всички уебуслуги (Web Services) ще се използва единствено протокол HTTPS със задължително прилагане на минимум TLS 1.2;
- Програмният код ще включва методи за автоматична санитизация на въвежданите данни и потребителски действия за защита от злонамерени атаки, като минимум SQL инжекции, XSS атаки и други познати методи за атаки, и ще отговаря, където е необходимо, на Наредбата за оперативна съвместимост и информационна сигурност;
- При проектирането и разработката на компонентите на Системата и при подготовката и разгръщането на средите ще се спазват последните актуални препоръки на OWASP (Open Web Application Security Project);
- Ще бъде изграден модул за проследимост на действия и събития в Системата. За всяко действие (добавяне, изтриване, модификация, четене) ще регистрира и съдържа информация със следните атрибути:
    - Уникален номер;
    - Точно време на възникване на събитието;
    - Вид (номенклатура от идентификатори за вид събитие);
    - Данни за информационна система, където е възникнало събитието;

Чл.36 а, ал. 3 от ЗОП

- Име или идентификатор на компонент в информационната система, регистрирал събитието;
- Приоритет;
- Описание на събитието;
- Данни за събитието.

- Астрономическото време за удостоверяване настъпването на факти с правно или техническо значение се отчита с точност до година, дата, час, минута, секунда и при технологична необходимост - милисекунда, изписани в съответствие със стандарта БДС ISO 8601:2006;
- Ще бъдат проведени тестове за проникване (penetration tests), с които да се идентифицират и коригират слаби места в сигурността на Системата.

6.3.6.6. Използваемост

6.3.6.6.1. Общи изисквания за използваемост и достъпност

Технологичната платформа на база на която ще бъде разработена системата отговаря на утвърдените стандарти за достъпност (accessibility) и ползваемост (usability). При дизайна на потребителския интерфейс се използва модула за теми на ASP.Net MVC (theming engine), което ще позволи да се имплементира унифицирана рамка, или основна страница (Master page). Всички други страници от системата взимат общата рамка на основната страница, където е зададена основната HTML структура и референциите към CSS файловете и другите ресурсни файлове. По този начин се гарантира, че всяка една от страниците ще има сходен дизайн и ще бъде валидирана като документ с ниво „strict" според препоръките на W3C/WAI.

По отношение на потребителския интерфейс, софтуерното решение ще се придържа към следните принципи:

- При проектирането и разработката на софтуерните компоненти и потребителските интерфейси ще се спазват стандартите за достъпност на потребителския интерфейс за хора с увреждания WCAG 2.0, съответстващ на ISO/IEC 40500:2012;
- Всички ресурси ще са достъпни чрез GET заявка на уникален адрес (URL). Не се допуска използване на POST за достигане до формуляр за подаване не заявление, за генериране на справка и други;
- Функционалностите на потребителския интерфейс на Системата ще бъдат независими от използваните от потребителите интернет браузъри и устройства, при условие че последните са версии в период на поддръжка от съответните производители.
- Всички текстови елементи от потребителския интерфейс ще бъдат визуализирани с шрифтове, които са подходящи за изобразяване на екран и които осигуряват максимална съвместимост и еднакво възпроизвеждане под различни клиентски

Чл.36 а, ал. 3 от ЗОП

операционни системи и браузъри. Няма да бъдат използвани серифни шрифтове (Serif).

- Полета, опции от менюта и командни бутони, които не са разрешени конкретно за ролята на влезлия в системата потребител, няма да са достъпни за този потребител. Това няма да отменя необходимостта от ограничаване на достъпа до бизнес логиката на приложението чрез декларативен или програмен подход.
- Всяка екранна форма ще има наименование, което да се изписва в горната част на екранната форма. Наименованията ще подсказват на потребителя какво е предназначението на формата.
- Всички търсения ще са нечувствителни към малки и главни букви.
- Полетата за пароли задължително ще различават малки и главни букви.
- Полетата за потребителски имена ще позволяват използване на имейл адреси като потребителско име, включително да допускат всички символи, регламентирани в RFC 1123, за наименуването на хостове;
- Главните и малките букви на въвежданите данни се запазват непроменени, не се допуска Системата да променя капитализацията на данните, въвеждани от потребителите.
- Системата ще позволява въвеждане на данни, съдържащи както български, така и символи на официалните езици на ЕС.
- Наименованията на полетата ще са достатъчно описателни, като максимално се доближават до характера на съдържащите се в тях данни.
- Системата ще поддържа прекъсване на потребителски сесии при липса на активност. Времето ще може да се променя от администратора на системата без промяна в изходния код. Настройките за време за прекъсване на неактивни сесии ще включват и възможността администраторите да дефинират стилизирана страница с информативно съобщение, към която Системата да пренасочва автоматично браузърите на потребителите в случай на прекъсната сесия;
- Дългите списъци с резултати ще се разделят на номерирани страници с подходящи навигационни елементи за преминаване към предишна, следваща, първа и последна страница, към конкретна страница. Навигационните елементи трябва да са логически обособени и свързани със съответния списък и да се визуализират в началото и в края на HTML контейнера, съдържащ списъка;
- За големите йерархически категоризации ще се предвиди възможност за навигация по нива или чрез отложено зареждане (lazy load).

6.3.6.6.2. Интернационализация

Изпълнителят, вземайки предвид изискванията посочени от Възложителя в техническата спецификация, предвижда:

- Системата ще може да съхранява и едновременно да визуализира данни и съдържание, което е въведено/генерирано на различни езици, а именно – двуезична на български и на английски език;

Чл.36 а, ал. 3 от ЗОП

- Всички софтуерни компоненти на Системата, използваните софтуерни библиотеки и развойни комплекти, приложните сървъри и сървърите за управление на бази данни, елементите от потребителския интерфейс, програмно-приложните интерфейси, уебуслугите и др. ще се поддържат стандартно и ще са конфигурирани изрично за спазване на минимум Unicode 5.2 стандарт при съхранението и обработката на текстови данни, съответно трябва да се използва само UTF-8 кодиране на текстовите данни.
- Всички публично достъпни потребителски интерфейси ще поддържат многоезичност - български и английски език.
- Публичната част на Системата ще бъде разработена и ще включва набори с текстове на минимум два официални езика в ЕС, а именно български и английски език. Преводите на английски език ще бъдат осъществени професионално, като няма да се допуска използването на средства за машинен превод без ръчна проверка и корекции от професионални преводачи.
- Версиите на съдържанието на съответните езици ще включват всички текстове, които се визуализират във всички елементи на потребителския интерфейс, справките, генерираните от системата електронни документи, съобщения, нотификации, имейл съобщения, номенклатурите и таксономиите и др. Данните, които се съхраняват в системата само на български език, се изписват/визуализират на български език;
- Системата ще позволява превод на всички многоезични текстове с подходящ потребителски интерфейс, достъпен за администратори на Системата, без промени в изходния код. Модулът за превод на текстове, използвани в Системата, ще поддържа и контекстни референции, които да позволяват на администраторите да тестват и да проверяват бързо и лесно направените преводи и тяхната съгласуваност в реалните екрани, страници и документи;
- Публичната част на Системата ще позволява превключване между работните езици на потребителския интерфейс в реално време от профила на потребителя и от подходящ, видим и лесно достъпен навигационен елемент в горната част на всяка страница, който включва не само текст, но и подходяща интернационална икона за съответния език;
- При визуализация на числа ще се използва разделител за хиляди (интервал).
- При визуализация на дати и точно време в елементи от потребителския интерфейс в генерирани справки или в електронни документи всички формати за дата и час щеса съобразени с избрания от потребителя език/локация в настройките на неговия профил:
    - За България стандартният формат е „DD.MM.YYYY HH:MM:SS", като наличието на време към датата е в зависимост от вида на визуализираната информация и бизнес-смисъла от показването на точно време;
    - Системата трябва да поддържа и всички формати съгласно ISO БДС 8601:2006.

6.3.6.6.3. Изисквания за използваемост на потребителския интерфейс

Потребителският интерфейс ще бъде изграден при спазване на следните изисквания:

Чл.36 а, ал. 3 от ЗОП

- Ще бъде реализирана възможност за добавяне и редактиране от страна на администраторите на Системата, без да са необходими промени в изходния код, на контекстна помощна информация за:
- Ще бъде разработена контекстна помощна информация за всички процеси, екрани и електронни форми, включително ясни указания за попълване и разяснения за особеностите при попълване на различните групи полета или на отделни полета;
- Контекстната помощна информация, указанията към потребителите и информативните текстове за всяка електронна административна услуга няма да съдържа акроними, имена и референции към нормативни документи, които са въведени като обикновен текст (plain-text). Всички акроними, референции към нормативни документи, формуляри, изисквания и др. ще бъдат разработени като хипервръзки към съответните актуални версии на нормативни документи и/или към съответния речник/списък с акроними и термини;
- Достъпът на потребителя до контекстната помощна информация ще бъде реализиран по унифициран и консистентен начин чрез подходящи навигационни елементи, като например чрез подходящо разположени микро бутони с икони, разположени до/пред/след етикета на съответния елемент, за който се отнася контекстната помощ, или чрез обработка на "Mouse Hover/Mouse Over" събития;
- Потребителският интерфейс ще бъде достъпен за хора с увреждания съгласно изискванията на чл. 48, ал. 5 от ЗОП.

6.3.7. Системен журнал

Изграждането решение ще осигурява проследимост на действията на всеки потребител (одит), както и версия на предишното състояние на данните, които той е променил в резултат на своите действия - системен журнал.

Атрибутите, които ще се запазват при всеки запис, ще включват като минимум следните данни:

- дата/час на действието;
- модул на системата, в който се извършва действието;
- действие;
- обект, над който е извършено действието;
- допълнителна информация;
- IP адрес и браузър на потребителя.

Размерът на журнала на потребителските действия нараства по време на работа на всяка система, което налага по-различното му третиране от гледна точка на организация на базата данни:

Чл.36 а, ал. 3 от ЗОП

- по време на работа на Системата потребителският журнал ще се записва в специализиран компонент, който поддържа много бързо добавяне на записи; този подход се налага, за да не се забавя излишно работата на Системата;
- специална фонова задача ще акумулира записаните данни и да ги организира в отделна специално предвидена за целта база данни, отделна от работната база данни на Системата;
- данните в специализираната база данни ще се архивират и изчистват, като в специализираната база данни ще бъде достъпна информация за не повече от 2 месеца назад; при необходимост от информация за предишен период администраторът на Системата първо ще възстанови архивните данни;
- ще бъде предоставен достъп до системния журнал на органите на реда чрез потребителски или програмен интерфейс; за достъпа ще се изисква електронна идентификация.

6.3.8. Дизайн на бази данни и взаимодействие с тях

При реализацията на решението, ще се използва стандартна релационна база данни, Microsoft SQL Server 2016 x64 Standad Edition, с която разполага НСИ..

При използване на база данни (релационна) ще бъдат следвани добрите практики за дизайн и взаимоедйствие с базата данни, в т.ч.:

- дизайнът на схемата на базата данни ще бъде с максимално ниво на нормализация, освен ако това не би навредило сериозно на производителността;
- базата данни ще може да оперира в клъстър; в определени случаи ще бъде използван т.нар. sharding;
- имената на таблиците и колоните ще следват унифицирана конвенция;
- ще бъдат създадени индекси по определени колони, така че да се оптимизират най-често използваните заявки; създаването на индекс ще е мотивирано и подкрепено със замервания;
- връзките между таблици ще бъдат дефинирани чрез foreign key;
- периодично ще бъде правен анализ на заявките, включително чрез EXPLAIN (при SQL бази данни), и ще бъдат предприети мерки за оптимизиране на бавните такива;
- задължително ще се използват транзакции, като нивото на изолация ще бъде мотивирано в предадената документация;
- при операции върху много записи (batch) ще се избягват дългопродължаващи транзакции;
- заявките ще бъдат ограничени в броя записи, които връщат;
- при използване на ORM или на друг слой на абстракция между приложението и базата данни, ще се минимизира броят на излишните заявки (т.нар. n+1 selects проблем);
- при използване на нерелационна база данни ще се използват по-бързи и компактни протоколи за комуникация, ако такива са достъпни.

Чл.36 а, ал. 3 от ЗОП

7. ПОДХОД ЗА ИЗПЪЛНЕНИЕ НА ИЗИСКВАНИЯТА КЪМ ПОРЪЧКАТА

7.1. Предложение за извършване на дейностите по анализ и проектиране

При извършване на дейностите по анализ и проектиране Изпълнителят ще следва Методологията за усъвършенстване на работните процеси за предоставяне на административни услуги и Наръчника за прилагане на методологията, приета с Решение № 578 на Министерския съвет от 30 септември 2013 г.;

7.1.1. Методология за анализ на процесите и моделиране

Реализацията на софтуерни системи с нарастваща сложност изисква прилагането на общи стандарти и ефективни методологии за проектиране и разработка. От появата си през 1997г. до сега Унифицираният език за моделиране – UML™ – се превърна фактически в стандартен „графичен език за визуализиране, специфициране, конструиране и документиране на елементите на една софтуерно-интензивна система". Сред предимствата на един UML подход са:

- Плавен преход между отделните етапи при разработката на софтуерния продукт – от формулирането на изискванията до крайната реализация, чрез използване на междинни модели за анализ и проектиране на архитектурата и поведението на системата, които позволяват всяко изискване да бъде адресирано на подходящото ниво на абстракция;
- Възможност за ефикасно разширяване на съществуващи модели с цел включване на допълнителни функционални изисквания в една бързо променяща се бизнес среда – проектът може да расте запазвайки ефективна и ясна архитектура;
- Стандартен графичен език за изразяване и дискусия на идеи, изисквания и проектантски решения, който улеснява комуникацията, особено при паралелна разработка от различни екипи (например при outsorcing на разработката);
- Един UML-базиран процес за разработка позволява да се настрои количеството и качеството на проектната документация според изискванията на конкретния проект (време, обем, бюджет, нужда от бъдещо разширяване).

UML моделирането ще бъде основно средство в процеса на анализа и проектирането на системата. С негова помощ ще се постигнат няколко основни цели:

- формално описание на единнатаа система и нейните компоненти (подсистеми);
- описание на начина на имплементация на работата на функционалността;
- провеждане на експерименти с модела (симулация) и база за взимане на решения за оптимизиране на системата.

UML осигурява независимост на създадените модели от техническата реализация в информационните и комуникационни технологии. Прилагането на стандарта в този проект

Чл.36 а, ал. 3 от ЗОП

ще бъде гъвкаво и съобразено със спецификата на анализираните процеси, по време на проектиране и създаване на моделите ще се запазва смисловото съдържание на моделираните обекти.

UML дефинира правила за изграждане на различни типове диаграми, които служат за графично представяне на различни аспекти на софтуерната система. Общият UML модел на който ще бъде разработен за софтуерната система се разглежда като изграден от два взаимно допълващи се модела:

Структурен модел (Structural model), който показва структурата на системата и подсистемите, използвайки обекти, атрибути, операции и връзки;

Функционален модел (Functional model), наричан още динамичен модел (Dynamic model), който показва функционалността на системата, обособена в конкретните и модули и измененията на системата във времето.

Всеки елемент от модела се описва с дадена UML диаграма - частично графично представяне на един от двата системни модела. В нотацията на UML съществуват 14 официални типа диаграми:

- **Клас диаграма** (Class diagram) – описва класове от обекти и връзките между тях;
- **Обектна диаграма** (Object diagram) – представя цялостен или частичен изглед на даден обект на системата;
- **Диаграма на пакети** (Package diagram) – представя йерархична структура на модул или компонент на системата;
- **Диаграми на компоненти** (Component diagram) – описва структурата и връзките между компонентите;
- **Диаграма на съставна структура** (Composite structure diagram) – представя декомпозиция на клас и комуникацията му с другите класове;
- **Диаграма за разгръщане/диаграма на внедряването** (Deployment diagram) – представя връзката между софтуерната и хардуерната реализация на системата;
- **Профилна диаграма** (Profile Diagram) – използва се за описване на класовете и пакетите;
- **Случай на употреба** (Use Case Diagram) – представя начина на взаимодействие между потребител и системата, употребява се за специфициране на функционалните изисквания към разработваната софтуерна система и изготвяне на тестове;
- **Диаграма на дейност** (Activity diagram) – описва последователни или паралелни контролни потоци;
- **Диаграма на на състоянията** (State diagram) – описва промяната на състоянието на определен обект или система по време на неговия жизнен цикъл в резултат на възникващи събития;

Чл.36 а, ал. 3 от ЗОП

- **Диаграма на последователност** (Sequence diagram) – служи за представяне на взаимодействието между обектите, като се набляга върху последователността на дейностите;
- **Диаграма на комуникация** (Communication diagram) – описва взаимодействията между обектите, като особено внимание се отделя на връзките между тях;
- **Диаграма за преглед на взаимодействието** (Interaction overview diagram) – представлява комбинация от диаграми на последователност и комуникационни диаграми;
- **Времева диаграма** (Timing Diagram) – представя взаимодействие между обекти, като се набляга върху синхронизацията на обектите.

BPMN

Business Process Model and Notation (BPMN) е другият стандарт, който ще се използва за визуално моделиране на бизнес процеси под формата на диаграма – **Business Process Diagram (BPD)**. BPMN надгражда Unified Modeling Language (UML), като целта е да се разшири техническата насоченост на описанието в UML, създавайки универсален език за описание на бизнес процеси, който да е еднакво четим и удобен за бизнес потребителите (мениджъри, бизнес анализатори и др.)и в същото време да дава възможност за описание на сложна семантика на бизнес процесите.

Поради спецификата на проекта и широкия набор от функционалности, която трябва да се реализира, използването на неформалния метод би довело до създаване на описания с твърде голям обем, чието систематично интерпретиране би било сериозно затруднение. Подобна сложна система е много по-подходяща за описание чрез диаграми и модели на данните, отколкото с текстово описание на изискванията. По тази причина най-подходящият подход към дейността е използването на полуформален подход, чрез комбинирането на документ със спецификация на изискванията, с модел от UML диаграми, представящи систематично изискванията към архитектурата на системата и отделните потребителски случаи (use cases).

7.1.1.1. Методи, стъпки и инструменти за реализация на дейностите по анализ и проектиране

Дейностите свързани с анализа и проектирането ще бъдат от ключово значение за успеха на разработката, тъй като Системата съдържа множество обвързани компоненти, които представляват сериозно предизвикателство пред използването на гъвкави подходи от екстремното програмиране базирани на прототипиране и на изграждане на системата итеративно на малки „парчета“. Потребителите на системата трудно могат да оценят изградения модел, ако той не е до голяма степен завършен, както и ако не е имплементирана съответна аналитична функционалност, която да покаже как потребителя консумира информацията. По този начин възможността за изграждане чрез итеративно подобряване на прототипи (throw away prototypes), не би била разумна от гледна точка на нужния ресурс и време за проектиране, разработка, зареждане с данни и тестване на тези прототипи. В тази

Чл.36 а, ал. 3 от ЗОП

връзка, избраният подход акцентира върху правилното специфициране на изискванията и максимално близка работа с екипа на Възложителя през целия процес на проектиране и разработка, така че да се постигне голяма степен на завършеност на реализацията преди пилотното използване, а след това да се премине към отстраняване на грешки и доразвиване на основната функционалност.

Предлаганата методология за разработка на процеса – RUP, поставя спецификацията на потребителските случаи (use case) в основата на процеса на анализ на изискванията ипроектиране, разработване и тестване на системата. Модела на потребителските случаи е гръбнака на формалното описание на системата, той се изготвя в процеса на спецификация на изискванията, като дори може да се каже че процеса на спецификация на изискванията бива инициран чрез процеса на спецификация на потребителските случаи, тъй като дефинирането на use case сценарии е естествен начин за извличане на изискванията към системата. **Инструментите** за изпълнение на дейностите по моделиране ще бъдат **Microsoft Visio** – софтуерен продукт за създаване на диаграми и проектиране, тясно интегрирана с **Microsoft Visual Studio** – средата за разработка на .Net приложения. Microsoft Visio е утвърден продукт, който съдържа нужната фунцкионалност за генериране на всички видове UML диаграми нужни за анализа и проектирането на системата – Use Case диаграми, клас диаграми, описания на моделите на архитектурата и т.н..

Първоначалния анализ на наличните източници на информация и срещи с екипа на Възложителят ще определят обхвата на модела на потребителските случаи, който следва да бъде правилно дефиниран в спецификацията за да се постигне пълното постигане на целите на проекта. Продуктът от тази дейност ще бъде изготвянето на спецификацията на изискванията както и на use cases, които да послужат като основа за проектирането и тестването на системата. Работата по спецификация на изискванията ще бъде надградена в процеса на изготвяне на модела на потребителските случаи (use case model), а по-късно и в етапа на проектиране, където ще се специфицира подхода за реализация на специфицираните изисквания.

При изпълнение на дейността по спецификация на изисквания ще бъде възприет итеративен подход към подготовката на детайлната спецификация. При този подход екипа на Изпълнителя ще извършва в итерации основните стъпки от процеса на формализиране на изискванията:

- Извличане на изискванията на база на изследване на идентифицираните източници на информация, както и на база на резултатите от проведените срещи и дискусии с екипа на Възложителя;
- Анализ и дефиниране на изискванията, в процес на обсъждане с Възложителя;
- Документиране на изискванията в проект на Спецификация на изискванията;
- Валидиране на изискванията, документирани в проекта на Спецификацията на изискванията, завършващо с обратна връзка от екипа на Възложителя.

На база на обратната връзка от Възложителя, в края на итерацията от стъпки, екипът от анализа или инициира наново процеса на извличане и анализ на изисквания, или се приема процеса на специфициране на изискванията за приключен.

Итеративния подход и съответстващите му дейности и резултати са описани графично на следната диаграма:

Чл.36 а, ал. 3 от ЗОП

Фигура 15 Диаграма на итеративния подход и съответстващите му дейности и резултати

7.1.2. Метод за спецификация на потребителските случаи

Потребителските случаи (use case) се смятат за стандартна нотация при изготвянето на изискванията в обектно-ориентирано моделиране. Потребителските случаи на употреба могат да подобрят комуникацията между заинтересованите страни, експерти по бизнес анализ, разработчици и други специалисти участващи в разработването на системата.

Разглеждат се три различни нива на абстракция на изискванията (бизнес, потребителски и системни) и това как потребителските случаи могат да бъдат определящ фактор при разработването на детайлните изисквания за разработване на системата.

- ✓ **Класификация на изискванията**

Изискванията към Система като тази, определят неговото функционално поведение, както и типовете информация, до която трябва да се осигури достъп, как тя се трансформира и организира.

Определянето на изискванията дават възможност на участниците в проекта да уточнят ясно целта, насоката и да изяснят очакванията от информационните цели. Тъй като често участниците – бъдещи потребителите на системата дават доста обобщени бизнес изисквания, а екипът разработващ системата има нужда от точни, ясни и недвусмислено очертани изисквания се налага допълнително изясняване, с цел те да се трансформират в

Чл.36 а, ал. 3 от ЗОП

детайлни, пълни и подлежащи на тестване спецификации, на базата на които да бъде разработена система максимално отговаряща на нуждите на Възложителя.

✓ **Бизнес изисквания**

Бизнес изискванията представят цели на високо равнище, които следва да се реализират чрез внедряването на системата. Те са описани в Техническата спецификация, която описва визията и обхвата на проекта. Бизнес изискванията идентифицират първичните ползи, които системата ще донесе на Поръчителя и потребителите. Те представляват най-високото ниво абстракция във веригата на изискванията. Те описват целите, възможностите и основните потребителски типове, които дават обобщена представа, за това как би следвало да функционира системата.

✓ **Потребителски изисквания**

Потребителските изисквания описват задачите и действията, които даден потребител трябва да може да извършва с помощта на системата. Тези изисквания трябва да се уточнят с хората, които на практика ще използват системата. Тези потребители са в състояние да опишат не само задачите, които е нужно да извършват в системата, но и нефункционалните й характеристики, които са важни за доброто приемане и лесната работа със системата.

Потребителските изисквания следва да са в синхрон с бизнес изискванията. Те могат да бъдат уточнени най-пълноценно посредством описанието на потребителските сценарии. Фокусират се върху конкретните нужди при използване на системата и съответно са доста по-значими от традиционният подход за извличане на изисквания, посредством директно питане на потребителите какво биха искали да прави системата.

✓ **Детайлни системни изисквания**

Те представляват системните изисквания на много детайлно ниво, което подпомага пълното, детайлно специфициране на изискванията, позволяваща използването на така специфицираните изисквания при изграждане на компонентите от разработчиците.

Тези изисквания трябва да съответстват на потребителските и бизнес изисквания и да съдържат точни и ясни критерии, които да позволяват верификация.

Функционалните изисквания определят функционалностите, които разработващият екип следва да реализира при изграждането на системата, така че системата да удовлетворява бизнес изискванията. Бизнес изискванията улавят очакваното поведение на системата, което може са бъде изразено чрез услуги, задачи и функции, от които системата се нуждае, за да отговори на целта, заложена в проекта.

Информационните изисквания дефинират информационните нужди на Поръчителя и заинтересованите страни. Те описват типовете информация и данни, които системата следва да предостави, или до които да даде достъп. Те специфицират предоставените данни, посредством качеството, което те следва да имат, източникът от който идват, как би следвало да бъдат обработвани, как следва да бъдат комбинирани за анализ и кои аналитични методи следва да бъдат използвани.

Другите изисквания, извън функционалните и информационните, могат да бъдат специфицирани, за да се опишат допълнителни аспекти на системата, като изисквания към интерфейса и средата (законова, културна, политическа).

✓ **Качествени характеристики на изискванията**

Чл.36 а, ал. 3 от ЗОП

Добавянето на качествени характеристика към описанието на функционалните, информационните и други изисквания, позволява оценяването им в определени измерения, които са важни или за потребителите или за екипа, разработващ системата.

Характеристиките на изискванията са свойства или качества, които системата трябва да притежава. Те могат да съдържат стандарти, регулации и условия, с които системата трябва да бъде съобразена: описания на външния интерфейс, изисквания за производителността, дизайна и ограничения при изпълнението; качествени характеристики.

- ✓ **Идентифициране на изисквания посредством потребителски случаи (use cases)**

Анализаторите от дълги години използват потребителските случаи на употреба (use cases), за да опишат начините, по които потребителя взаимодейства със системата. Целта е да се извлекат релевантни и точни изисквания. Използването на потребителски случаи възниква с напредъка на обектно-ориентирания подход към програмирането, подходът който е заложен в основата на подхода за проектиране и разработка на предлаганото решение.

- ✓ **Модел на потребителските случаи и процес на изготвяне на детайлната спецификация**

Описанието на потребителските случаи често не дава достатъчно пълна информация на екипа разработващ системата за функционалностите, които те следва да разработят, както и за това каква и коя информация следва да бъде достъпна. Събирането на всички изисквания от потребителските случаи може да доведе до формирането на твърде тромави и комплексни потребителски случаи. От друга страна, спирането на развитието на изискванията по време на стадия на събиране на потребителските изисквания ще доведе до много пропуски в информацията, от която екипът разработващ системата ще има нужда по време на стадия на конструирането ѝ. За да се понижи тази несигурност всеки потребителски случай трябва да бъде разработен заедно с пълните си системни изисквания.

Всеки потребителски случай на употреба води до определянето на няколко системни изисквания, които ще позволят на специфицирането на дадена функционалност в разработеното приложение и съответно няколко потребителски случая могат да формират спецификацията на една и съща функционалност или компонент.

Специфицирането на функционалностите, компоненти и изискванията към тяхното качество може да бъде описано чрез детайлни системни изисквания, които могат да бъдат асоциирани с потребителските случаи по няколко начина.

Подходът, който ще бъде избран зависи от това дали екипът, разработващ системата трябва да създаде дизайна, конструкцията и тестовете, базирайки се на потребителски случаи, на детайлни системни изисквания или на комбинация от двете. Избирайки подходящия подход е важно, да се избягва дублирането на информация от различни източници, което би направило спазването на изискванията много по-трудно.

- ✓ **Итеративно и инкрементално разработване на детайлната спецификация**

Екипът на Изпълнителя предлага итеративния и инкрементален подход, който е в основата на RUP да бъде приложен по същия начин към дейността по изготвяне на детайлната спецификация. Итеративният подход към изготвянето на детайлната спецификация включва пресяването на потребителските случаи посредством няколко итерации. Итеративните стъпки към събирането на изисквания от потребителски случаи е

Чл.36 а, ал. 3 от ЗОП

строго индивидуално според ситуацията. Предварително няма как да се определи точен брой итерации, които ще бъдат необходими, за да се покрият изискванията във всички ситуации. Но може да се каже, че спецификацията на итеративните изисквания винаги преминава през същите логически стъпки във всички ситуации.

7.1.2.1. Подход към осигуряване на качеството на продукта от дейностите по спецификация на системния проект

При осигуряването на качеството на извършените дейности ще бъде използван като референтен модел препоръките на IEEE - Reccomendation for SRS. Използвайки IEEE Standard 830, Recommended Practice for Software Requirements Specifications, ръководството на екипа ще съблюдава правилата и критериите за качество при изготвяне на спецификацията. Целта на въпросните действия по осигуряване на качеството е да се гарантира, че изготвения документ ще предложи стабилна и ясна рамка за проектирането и изграждането на технологичната реализация. Така специфицираните изисквания ще отговарят на изискванията за качество на спецификацията на изискванията:

- ✓ **Точност**

Този атрибут не е формално проверим в процеса на качествен контрол, но итеративния модел на валидиране на изисквания позволява на крайния потребител – екипът на Възложителя да провери документираните изисквания, по отношение на това как те представят описаната визия за изискванията към решението.

- ✓ **Недвусмисленост**

Недвусмислеността на изискванията ще бъде проверена чрез преглед на изискванията от трети лица, тъй като основните проблеми при изготвянето на изискванията идват в различното тълкуване на едно и също изискване от разработчиците и бъдещите потребители.

- ✓ **Пълнота**

Тук ще се приложи сравнение със стандартите за съдържание на спецификацията, така че да се сравни дали всички задължителни елементи присъстват и са опасни в нужната пълнота. При итеративния процес на подготовка, всички непълни или отсъстващите елементи ще бъдат ясно индикирани и коментирани, така че в крайния вариант да не се допусне одобрение без наличие на пълния набор от елементи.

- ✓ **Непротиворечивост**

Чрез използване на речник на термините и дефиниции за основните предположения ще се избегне конфликт между отделни части от документа за спецификация на изискванията, както и между спецификацията и други документи свързани с изпълнението на проекта.

- ✓ **Приоритизирани по важност**

Процеса на приоритизиране на изискванията следва да бъде извършен от екипа на Възложителя, така че от една страна да се обсъди аргументацията на приоритетите, така че екипът по изпълнение да възприеме логиката и да научи съответните ключови предположения, а от друга страна да се насочи вниманието и на екипа по разработка.

Чл.36 а, ал. 3 от ЗОП

✓ **Проверими**

Работата по проверка на този индикатор за качество ще бъде изключително полезна в рамките на подготовката на тестовия план. Проверката се изразява в дефиниране на ясен критерий за проверка на всяко изискване.

✓ **Променими**

С цел лесната итеративна работа по подобрение на изискванията те ще бъдат документирани в съответствие със структурата на спецификация при спазване на ясни правила за идентифициране и реферирание на отделните изисквания, осигуряващи че всяко изискване се включва еднократно в спецификацията, а се референцира от другите части на документа, ако има зависимост с друго изискване. По този начин промените в документа не нарушават неговата консистентност.

✓ **Проследими**

С цел проследимост на развитието на изискванията, всяко изискване ще бъде идентифицирано с уникален номер, като се поддържа и информация за неговата идентификация в по-ранни версии на документа.

7.1.3. Методика за проектиране

Проектирането на компонентите ще бъде извършено от екипа на Изпълнителя съобразно съвременни, доказани и препоръчвани практики в областта на софтуерния дизайн, които са доказали своята приложимост при разработката на комплексни софтуерни системи.

В съответствие с RUP, процесът на проектиране ще бъде извършен като итеративен процес на дефиниране и подобряване на архитектурата, на дизайна на компонентите, интерфейсите и останалите характеристики на компонентите. В основата на дейността ще се състои в това изискванията към системата да се анализират с цел да се изработи визия за нейната вътрешна структура, която след това да се развие в цялостна архитектура, която да послужи за основа на разработката. Така разработената архитектура ще описва как системата се декомпозира и организира на отделни компоненти и интерфейсите между тези компоненти. След това детайлно се проектира и функционалността и интерфейса на всеки от компонентите, така че да се постигне необходимото ниво на детайлност на това описание до степен, която да позволи на разработчиците да извършат самото изграждане на компонента. Описаният подход, избран от екипа на Изпълнителя, се характеризира от следните основни дейности:

- Дизайн на софтуерната архитектура (дизайн на високо ниво) – описва се структурата и организацията на системата на високо ниво. Идентифицират се отделните компоненти, от които тя се състои.
- Дизайн на компонентите/модулите – всеки компонент/модул се описва с ниво на детайлност, което е достатъчно за неговото конструиране.

Резултатът от дейността по проектиране ще бъде логическия дизайн и организация е съвкупност от модели и артефакти, които съдържат/описват взетите важни решения в процеса на тяхното изготвяне.

Чл.36 а, ал. 3 от ЗОП

Прилагането на гъвкава и модулна архитектура, структурирана по начин, който едновременно осигурява консистентност на данните и максимална защита от грешки, а от друга страна позволява лесно надграждане и разширяване на системата, без този процес да бъде съпътстван от регресивни грешки поради скрита свързаност на модулите, е изключително сложна задача. За постигне на нейните цели следва да се спазват основните принципи на обектно ориентирания дизайн.

7.1.3.1. Принципи за дизайн

В основата на добрият софтуерен дизайн е спазването на определени принципи за дизайн, които са доказали своята ефективност и ефикасност при множество успешни софтуерни системи:

- ✓ **Абстракция** – абстракцията може да бъде определена като процес на "забравяне" на информация, така че дадено множество от различни по своята същност неща да могат да бъдат третирани като еднакви. Абстракцията по същество представлява опростяване на проблемната област, като цената която се плаща е загуба на информация;
- ✓ **Свързаност и кохезия** – свързаността/обвързаността (copupling) дефинира силата на връзката между отделните модули, докато кохезията (cohesion) дефинира доколко вътрешните елементи на даден модул са взаимно (логически) свързани и служат за изпълнението на неговите функции. При наличие на висока свързаност между модулите, ако трябва да се направи изменение в даден модул е твърде вероятно това да наложи промени и в свързаните модули. Обратно при ниска свързаност промените обикновено се ограничават до модула, който реализира измененатa функционалност. Високата кохезия предполага, че логически свързаните елементи са групирани заедно. Това спомага да се намали сложността, тъй като се елиминират сложните последователности от взаимодействия между различните модули. В идеалния случай модулите, от които се състои дадена софтуерна система трябва да имат ниска свързаност и висока кохезия.
- ✓ **Декомпозиция и модулност** – големите и сложни софтуерни системи е добре да се декомпозират на отделни под-системи и модули, като целта е да се разположат различните функционалности и отговорности в различни компоненти. По този начин се постига разделяне на отделни компоненти, чиято сложност е управляема.
- ✓ **Енкапсулация** – енкапсулацията (encapsulation/information hiding) представлява групиране и пакетиране на елементите и вътрешните детайли на дадена абстракция, правейки ги недостъпни отвън. Целта е да се постигне потенциал за промяна: вътрешните механизми на компонента могат да бъдат променяни и подобрявани без това да оказва влияние на останалите компоненти. Обикновено се енкапсулират онези аспекти на компонента, които има вероятност да бъдат променяни в бъдеще.
- ✓ **Разделяне на интерфейсите от имплементацията** – принципът е свързан с принципа на скриване на информацията. Интеракцията с дадена абстракция се осъществява само чрез строго дефинирани публично достъпни интерфейси,

Чл.36 а, ал. 3 от ЗОП

вътрешната структура остава скрита.

- ✓ **Достатъчност, пълнота и простота на дизайна** – постигането на достатъчност, пълнота и простота означава, че софтуерния компонент обхваща всички важни характеристики на абстракцията, която реализира и нищо повече.

7.1.3.2. Шаблони за дизайн (Design patterns)

При обектно ориентирания дизайн стремежът е системата да се планира като съвкупност от взаимодействащи обекти. Всеки обект представя даден елемент от системата, която се моделира и се характеризира със своето състояние и поведение. Обектът представлява абстракция групираща в себе си данни и процедури. Интеракцията с обекта става посредством добре дефиниран интерфейс на обекта. Ключова роля при обектно ориентирания дизайн играят понятията наследяване и полиморфизъм.

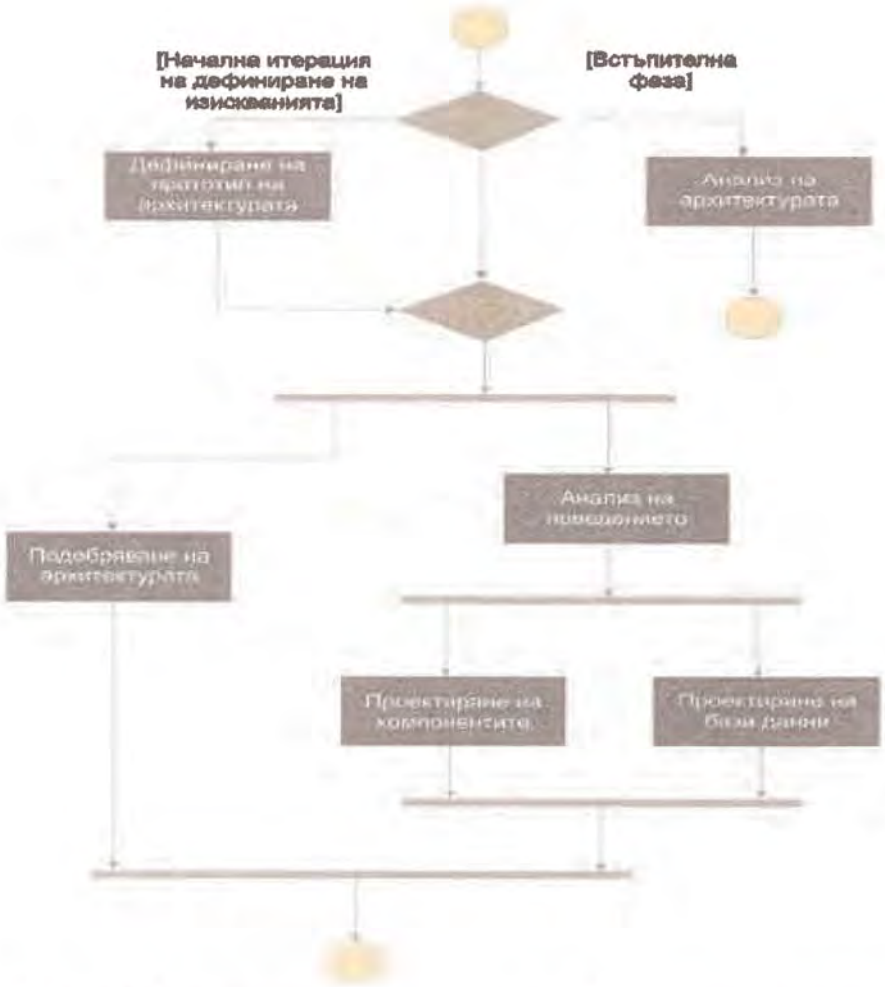
Подходът при обектно ориентирания дизайн е проблемния домейн да се декомпозира на отделни обекти следвайки принципите на дизайн от общото към частното и свързване на частите съобразно техните отговорности. Обектно-ориентирания анализ и дизайн по същество е процес на последователни действия на опростяване 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 (design patterns), които са доказали своята ефективност при много различни ситуации. Докато различните архитектурни стилове за софтуерен дизайн представляват шаблони от високо ниво (определят макроархитектурата), съществуват шаблони за дизайн описващи детайлите на по-ниско, локално ниво (определят микроархитуктурата). Такива видове са: Creational patterns (builder, factory, prototype, singleton), Structural patterns (adapter, bridge, composite, decorator, façade, flyweight, proxy), Behavioral patterns (command, interpreter, iterator, mediator, memento, observer, state, strategy, template, visitor).

7.1.3.3. Приложение на итеративния подход в обектно – ориентирано проектиране

Основните методи в процеса на разработка на софтуер, в предлаганата от Изпълнителя методология – процес за разработка на софтуер RUP, се базират изключително на итеративния обектно-ориентиран подход на анализ и дизайн на системата. В началните итерации артефактите ще бъдат предимно документи, описващи изискванията и съдържащи аналитични и технически UML модели. Следващите итерации ще създадат софтуерни версии, които реализират специфицираните изисквания. Последните итерации ще се съсредоточат върху тестове, отстраняване на програмни грешки, внедряване на софтуера и евентуално очертаване на бъдещото й развитие.

На диаграмата е показана архитектурата на процесите за обектно ориентиран анализ и проектиране:

Чл.36 а, ал. 3 от ЗОП



Фигура 16 Архитектурата на процесите за обектно ориентиран анализ и проектиране

В началото фокусът се насочва върху изграждането на първоначална архитектура на компонентите (Define a Candidate Architecture), за да може да бъде дадена отправна точка за главната работа по анализ. В случай, че вече съществува архитектура, която е била изготвена на предишни итерации, в предишни проекти или зададена като изискване, фокусът на работата се измества върху нейното актуализиране (Refine the Architecture). Следващата стъпка е да се анализира очакваното поведение на системата и да се специфицират конкретните елементи, които изграждат това поведение (Analyze Behavior).

След като първоначалните елементи бъдат идентифицирани, следва да бъдат допълнително детайлизирани в етапите на дизайн на компонентите и базата данни. Дизайна на компоненти (Design Components) изготвя множество компоненти, които доставят изискваното поведение на системата. Паралелно с това, анализ на начините за съхранението им в базата данни бива извършен в дизайна на базата данни (Design the Database).

Реализацията на тази концепция и постигането на обектно-ориентиран анализ и дизайн на системата се извършва на базата на готовите дефинирани изисквания под формата

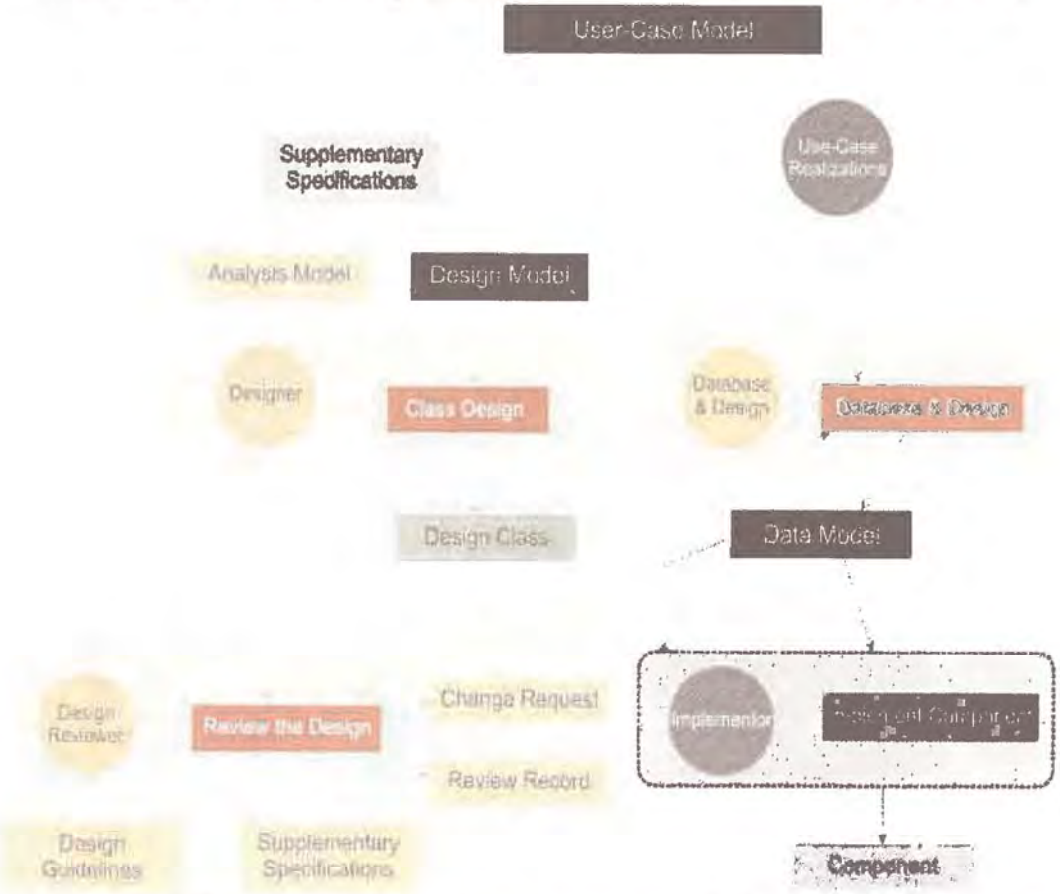
Чл.36 а, ал. 3 от ЗОП

на Модел на потребителските случаи (Use Cases), от които се изгражда тяхната реализация под формата на Клас диаграми (Class diagrams) и Диаграми на последователността и взаимодействието (Sequence diagrams и Collaboration diagrams, съответно). Това е имплементационния модел на системата, който включва още Дизайн модел и Модел на данните.

Дизайн моделът е обектен модел, който описва реализацията на потребителски случаи и служи за извеждане на Модела на имплементацията и неговия програмен код.

Моделът на данните е подмножество на имплементационния модел, което описва логическия и физически вид на постоянните (персистентни) данни в системата. Той включва и видовете поведения в базата данни, например записани процедури, активатори, ограничения и др.

На диаграмата е показана процесната архитектура за дизайн на компонентите:



Фигура 17 Диаграма на процесната архитектура за дизайн на компонентите

Използването на модели ще позволи лесна промяна и бъдещо разширение на съществуващите услуги и добавяне на нови електронни услуги.

Чл.36 а, ал. 3 от ЗОП

7.1.3.4. Методика за проектиране на потребителските интерфейси

Потребителският интерфейс ще бъде изграден чрез използването на стандартите HTML 5 и CSS 3. Всички уеб страници ще бъдат изградени така, че структурата и съдържанието на страницата (HTML) да бъдат в максимална степен отделени от тяхното визуално представяне (CSS). Това ще позволи при необходимост лесната модификация на визуалните елементи.

При изграждането на потребителският интерфейс ще бъде използван подхода Responsive Web Design (RWD), който ще осигури гъвкавост и възможност за оптимално представяне на данните върху различните браузъри. За реализирането на богат, динамичен и удобен за работа потребителски интерфейс, който не изисква често презареждане на съдържанието от сървъра, презентационния слой ще бъде реализиран като Single Web Application. Ще бъдат използвани библиотеки за динамично уеб съдържание jQuery (http://jquery.com/), KnockoutJS (http://knockoutjs.com) или AngularJS (http://angularjs.org/). Конкретната библиотека, която ще бъде използвана за динамично уеб съдържание ще бъде избрана съобразно конкретните изисквания към потребителския интерфейс дефинирани в резултат на анализа.

При проектирането на потребителския интерфейс ще бъде използван подхода залегнал в стандарта Human centered design processes for interactive systems (ISO 13407) и препоръките на W3C в Web Content Accessibility Guidelines 2.0 (ISO/IEC 40500:2012).

Потребителският интерфейс ще спазва следните добри практики:

- ✓ Интерфейсът предоставян от всеки модул ще бъде ориентиран към реализираните от него функции. Когато потребителят има нужда от достъп до информация в друг модул ще бъде предвидена възможност за пряк достъп до нея без да е необходима навигация през менютата на системата
- ✓ Опциите в менютата и достъпната функционалност в интерфейса на системата ще зависят от ролята и правата за достъп на потребителя, екранните форми няма да бъдат утежнени с излишни функции
- ✓ Потребителят ще бъде информиран от системата за успеха или неуспеха на изпълняваните от него операции
- ✓ Логически свързаните полета ще бъдат групирани;
- ✓ Всяко поле ще бъде означено с етикет подсказващ неговата функция;
- ✓ Полетата ще бъдат разположени на екрана, така че да се ограничи дължината на вертикалния скрол. Хоризонтален скрол ще се допуска по изключение;
- ✓ Всяка форма извършваща промяна в данните ще притежава бутон "Отказ", при натискането на който ще се отменя действието
- ✓ Съобщенията за грешка във въвежданите данни ще съдържат ясна информация за вида на грешката и очаквания формат на данните.

Чл.36 а, ал. 3 от ЗОП

7.1.4. Инструменти за описание на аналитичните дейности

Business Process Model & Notation (BPMN) представлява стандартизиран и съвременен вариант за дефиниране и анализ на бизнес процеси. BPMN се състои от стандартна нотация, която е интуитивна за разбиране от различните роли в дадена организация – мениджмънт, бизнес анализатори, технически специалисти и програмисти. Основната цел на BPMN е да подпомага управлението на бизнес процеси (business process management), като предоставя универсален език за комуникация между ролите създаващи дизайн на бизнес процеси и техните изпълнители. След преценка на добрите практики във функционалното моделиране и на спецификите на настоящия проект, включително традициите в агенцията, измежду богатото разнообразие от изразни средства в UML, „Смарт Системс 2010“ ЕООД избра следните диаграми, които ще се използват за функционалните модели на новата система в своята техническа и експлоатационна документация:

UML диаграми

Целта на този точка е да се направи въведение в унифицирания език за моделиране UML. UML е стандартен (ISO/IEC 19501:2005), език за моделиране с общо приложение в областта на софтуерното инженерство, създаден от Grady Booch, Ivar Jacobson и James Rumbaugh в Rational Software през 1990те, последващо придобит от Object Management Group (OMG) в 1997, където се управлява и досега.

От 2000 година UML се приема от International Organization for Standardization (ISO) като индустриален стандарт за моделиране на софтуерни системи.

Диаграма на дейностите

„Смарт Системс 2010“ ЕООД ще използва диаграми на дейностите, за да моделира последователността от действия, които описват определена функционалност на системата.

Диаграмата на дейности е техника за описване на процедурна логика, бизнес процес и работен поток.

Тя е като блок-схема, но поддържа паралелно поведение. Може да има разклонения както при блоксхемите. Показва логиката на поведение. Състоянията са дейности. Дейност е изпълнението на задача, като това може да бъде физическа дейност или изпълнение на код. Диаграмата на дейности показва последователност от действия.

Диаграмата на дейностите е по-удобна за описание на процеси на бизнес ниво, поради две причини:

- Показва всички възможни развития на процеса (успешен и не успешен завършек)
- Описва логиката на процесите, без да описва средствата или градивните елементи.
- Начало

Всяка диаграма на дейности има едно начало, от което започва последователността от действия.

Чл.36 а, ал. 3 от ЗОП

Фигура 18 Начало на диаграма за дейности

- Край

Всяка диаграма на дейности има един, край където завършва последователността от действия.

Фигура 19 Край на диаграма за дейности

- Действие

Действията са свързани помежду си с преходи.

Фигура 20 Действие в диаграма за дейности

- Преходи

Преходите се означават със стрелки и показват посоката от предишно съм следващо действие. Понякога съдържат текст.

Фигура 21 Преходи в диаграма за дейности

- Процес

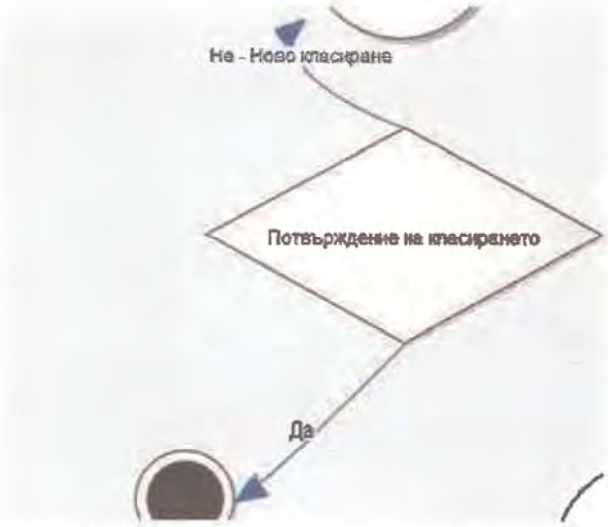
Процесите се визуализират с кръг.

Фигура 22 Процес в диаграма за дейности

- Процес за проверка

Чл.36 а, ал. 3 от ЗОП

Осъществява се проверка за валидация на формата. Ако е валидна се изпраща формата и с това приключва процеса, а ако не е валидна се връща процеса в попълване на формата.



Фигура 23 Процес за проверка

- Обекти

Фигура 24 Обекти в диаграма за дейности

Use Case Diagram (Диаграма на потребителските случай)

В процеса на проектиране на софтуерния продукт, диаграмата на потребителските случаи е първата диаграма, която се създава от проектантите, когато се започне проект. Тази диаграма позволява да се опишат на най-високо ниво целите на потребителя, които системата трябва да изпълнява. Тези цели не е необходимо да са задачи или действия, а може да са по общи изисквания към функционалността на системата. С други думи това е техника за определяне на функционалните изисквания на една система. Те описват типичните взаимодействия между потребителите и системата, предоставят описание на начина, по който тя се използва.

Примерите за използване (Use-cases) са основани на сценарий техника в UML, която идентифицира актьорите в едно взаимодействие и която описва самото взаимодействие. Множество от примери на използване би трябвало да опише всички взаимодействия със системата.

Сценариите са примери от реалния живот, как системата може да се използва.

Те трябва да включват:

- Описание на началната ситуация

Чл.36 а, ал. 3 от ЗОП

Чл.36 а, ал. 3 от ЗОП

- Описание на нормалния поток от събития
- Описание на това, което не е както трябва
- Информация за други дейности, извършвани в същия момент
- Описание на състоянието, в което сценарият завършва

Компоненти при Use case диаграми

- Actor (Актьор) – роля, която един потребител играе по отношение на системата. Може да бъде клиент, представител по поддръжка на клиенти, мениджър продажби, продуктов аналитик. Актьорите изпълняват случаите на употреба. Един актьор може да изпълнява много случаи на употреба и един случай на употреба може да се изпълнява от много актьори. Актьора клиент може да представлява множество хора. Един човек може да играе ролята на няколко актьора. Актьора може да не е човек, а друга компютърна система, ако системата извършва услуга за нея.

- Associations – плътна връзка без стрелка. Associations са връзка между actors и use cases и означава, че актьора осъществява потребителския случай.

- Случай на употреба

7.2. Предложение за извършване на дейностите по разработка на системата

7.2.1. Методология за извършване на дейностите по разработка на системата

Чл.36 а, ал. 3 от ЗОП

Предлаганата методология за разработка се основава на принципите на RUP, като за източник за логиката на реализацията се базираме на софтуерната спецификация на изискванията и софтуерния дизайн на компонентите специфициран в системния проект, както и на изготвените в рамките на изпълнението му план за тестване и тестовите сценарии за контролиране на качеството на софтуера, част от паралелния нему процес по контрол на качеството на разработката.

Цялостната производствена процедура обхваща няколко паралелни процеса и може да бъде обобщена в следните стъпки:

- ✓ Стъпка 1 - Разработване на софтуерните модули (units);
- ✓ Стъпка 2 - Разработване на план за тестване и тестовите сценарии;
- ✓ Стъпка 3 - Разработка на модулните тестове (unit tests)
- ✓ Стъпка 4 - Проверка (тестване) на софтуерните модули;
- ✓ Стъпка 5 - Интегриране (build) на софтуерните модули;
- ✓ Стъпка 6 - Разработване и провеждане на интеграционни тестове (integration tests);
- ✓ Стъпка 7 - Зареждане на системата с начални данни;
- ✓ Стъпка 8 - Разработване и провеждане на системни тестове;
- ✓ Стъпка 9 - Произвеждане на крайния продукт и документация.

7.2.1.1. Методика за разработване на модела на данни

Предложената методика за разработване на модела на данните в системата обхваща всички етапи от жизнения цикъл на проектирането на базата данни, като отчита спецификите на отделните приложения с които трябва да се интегрира системата. Една от основните специфики на приложенията, която ще бъде взета под внимание е тяхната архитектура – някои от приложенията с които се интегрира системата са разпределени, а други са централизирани, в допълнение всяко от тях е реализирано на базата на различни технологии и платформи.

Основните етапи от жизнения цикъл на разработването на модела на данни са следните:

- Проучване и анализ на изискванията
- Изграждане на логически модел
- Проектиране на базата данни

По-надолу ще развием в детайли всеки един от тези етапи.

- ✓ **Проучване и анализ на изискванията**

Основната цел на този етап от разработването на модела данни е да се определят потребителските изисквания към базите данни. За целта е необходимо да се идентифицират компонентите и приложенията, взаимодействащи си със системата. Това са всички

Чл.36 а, ал. 3 от ЗОП

разработени приложения, които ще бъдат интегрирани със системата, както и възможните нови такива.

На този етап трябва ясно да се определят нуждите на потребителите, изискванията към системата и ограниченията върху решението.

Изследването на системата е особено важен етап в процеса на цялата разработка. Допуснатите пропуски на този етап рефлектират върху пълнотата на проекта. Най-общо е необходимо да се проведе проучване, както бе описано в предходната точка в следните насоки:

- Функционирането на съществуващата система и несъвършенствата в нея;
- Икономическата и организационна среда на системата и възможните промени в бъдеще;
- Проблемите, слабите места и ограниченията на системата, както и породилите ги причини;
- Основните процеси и начина на тяхното реализиране;
- Типовете данни и техния обем;
- Справките и отчетите, необходими за функциониране на системата;
- Потребителските изисквания и възможностите за тяхната реализация.

По време на проучването и анализа е важно да се осмислят взаимовръзките между отделните компоненти на системата. Указаните области на проучване не се изследват последователно. Те са дадени в този ред за по-голяма прегледност. При изследването е важно да се постигнат крайните резултати, т.е. да се опишат основните потребителски изисквания и да не се пропуснат някои важни детайли.

✓ **Изграждане на логически модел**

Когато се проектира една база данни, е необходимо да се вземат решения по отношение на създаването на най-удобния модел на дадена система от реалния свят. При създаване на модела на данните се извършва организиране на изискванията на системата в едно логическо представяне на базата данни. Моделът на данните се състои от обекти и техните атрибути и ограничения, дефиниции на релации между обектите и ограничения върху тези релации, следователно разработването на логически модел на данните се осъществява чрез определяне на обектите, техните атрибути и ограничения и техните релации. Създаденият по този начин модел позволява да се идентифицират съответните таблици, които трябва да бъдат създадени, колоните, съдържащи се в тях, релациите между таблиците.

- Определяне на обектите и техните колони – когато се определят обектите, е необходимо, от изискванията на системата да се дефинират основните логически подразделения на информацията. Докато се разглеждат изискванията на системата, се дефинират основните обекти и събития. В резултат на това се добавят таблици към дизайна на базата данни, които им съответстват.

След като са дефинирани всички таблици, които могат да бъдат определени в този момент, трябва да се дефинират колоните на тези таблици. Тази информация се взема

Чл.36 а. ал. 3 от ЗОП
Чл.36 а, ал. 3 от ЗОП

директно от изискванията на системата, в които е определено какви данни трябва да се поддържат за обектите и събитията.

- ✓ **Определяне на релациите между обектите** – по време на процеса на дефиниране на релациите между таблиците може да се открие, че е необходимо да се промени създадения до този момент дизайн. Започва се с избиране на една от основните таблици и определяне на обектите, които имат релации към тази таблица.

След като се установят релациите между таблиците, трябва да се дефинира типа на тези релации. Всяка линия се маркира в двата си края или с цифрата 1, или със символа ‚n'. Цифрата 1 се отнася до страната едно на релацията, а символът ‚n' се отнася до страната много на релацията. За да се определи типа на релацията, се разглеждат данните, които съдържат таблиците и видовете взаимоотношения между тях. В един нормализиран проект на базата данни обаче релациите много към много трябва да се модифицират, като се добави една свързваща таблица и между нея и във всяка първоначална таблица да се създадат релации едно към много. Аналогично се определят и типовете на останалите релации.

- ✓ **Определяне на ограниченията върху данните** - работните правила включват всички ограничения за една система, включително за целостта на данните и сигурността.

На този етап от процеса на изграждане на модела на данните трябва да се обърне внимание на спецификите и ограниченията върху данните. Например дадено поле може да не е задължително във формата, а да е позволено да се попълни на по-късен етап. В такъв случай полетата, които съдържат информация, която не е задължителна, да бъдат маркирани като опционални, т.е. да позволят съхраняване на стойност NULL в тези полета. От друга страна, ако разгледаме поле, което е задължително за попълване, колоната трябва задължително да има попълнена стойност и тя да съществува като стойност в колоната ID от съответната таблица.

7.2.1.2. Проектиране на базата данни

Теорията на проектиране на релационни бази данни се състои от следните основни понятия:

- Таблици и уникалност - Първичен ключ (primary key) уникален идентификатор на ред, който представлява колона или група от колони, използвани за разграничаване всеки отделен ред от останалите редове в таблицата. Може да бъде прост или съставен. Простият ключ е създаден от една колона, съдържаща уникални стойности за всеки ред от таблицата.

Всяка таблица може да има само един първичен ключ, дори когато няколко колони или комбинации от колони съдържат уникални стойности (наречени алтернативни ключове).

Изборът на първичен ключ трябва да се основава на принципите за:

- минималност (избират се толкова колони, колкото е необходимо);
- стабилност (избират се колони, които рядко биват променяни);
- простота (от колкото е възможно по-прост тип)

Чл.36 а, ал. 3 от ЗОП

Чл.36 а, ал. 3 от ЗОП

- Външни ключове и домейни - Външен ключ (foreign key) колона или група от колони в дадена таблица, представляваща връзка с друга таблица посредством нейния първичен (родителски) ключ.

Родителски ключ е ключът, към който сочи един външен ключ. Родителският ключ трябва да бъде уникален идентификатор, за да може да се определи към кой ред от таблицата на родителския ключ сочи външният ключ. Броят и типът на колоните, съставящи външния ключ, трябва да съответства на броя и типа на колоните на родителския ключ, но могат да се използват различни имена за тях. Не е задължително стойностите на външния ключ да бъдат уникални в своята-собствена таблица. Те трябва да бъдат в същата област от допустими стойности (домейн), на която принадлежат стойностите на родителския ключ.

Домейн съвкупност от стойности, които са допустими за дадена колона.

- Релационни връзки - Релация (relationship) се нарича връзка между таблици, която е базирана на първичен ключ от едната таблица и външен ключ от другата таблица.

Съществуват три типа релации между таблиците: релации едно към едно; релации едно към много; релации много към много.

Релации едно към едно - две таблици А и В са свързани с релация едно към едно, ако за всеки ред от таблицата А има най-много един съответстващ ред от таблицата В, но всеки ред от таблицата В може да има точно един съответстващ ред от таблица А.

Релации едно към много - две таблици А и В са свързани с релация едно към много, ако за всеки ред от таблицата А има нула или повече съответстващи редове от таблицата В, но всеки ред от таблицата В може да има точно един съответстващ ред от таблицата А.

Релации много към много - две таблици А и В са свързани с релация много към много, ако за всеки ред от таблица А има много съответстващи редове от таблицата В и обратно. Този тип релация се създава, като се дефинира трета таблица, наречена свързваща таблица (junction table), която съдържа първичните ключове от двете таблици като външни ключове; първичният ключ в свързващата таблица е съставен от двата външни ключа.

- Правила за запазване на целостта на данните - целостта на данните е важно понятие за проектиране на базите данни. Има четири вида цялост на данните:
  - цялост на обект – едно от изискванията на проектирането на релационна
  - база данни е възможността да се разграничат различните инстанции на даден обект. Това понятие е известно като цялост на обект и се реализира чрез създаване на първичен ключ. Според това правило за цялост, колоните съставящи първичния ключ, не могат да имат стойност NULL. Релационните бази данни поддържат специална стойност NULL, която указва неизвестните стойности (unknown);
  - цялост на област – свързана е с осигуряване на валидност на стойностите на колоните, т.е. да принадлежат на допустима област от стойности. Реализира се с определяне на типа на колоните, допускане на стойност NULL, ограничения, стойност по подразбиране, дефиниране на външен ключ;
  - цялост на връзка – запазва дефинираните отношения (релации) между таблиците, когато се въвеждат, променят или изтриват редове. Целостта на

Чл.36 а, ал. 3 от ЗОП

връзките гарантира, че съществува съгласуваност на стойностите на ключовете между таблиците – първичните и външните в съответните таблици. Реализира се с дефиниране на ограничението външен ключ. Когато се реализира цялост на връзките, не се допуска да бъдат добавени редове в една таблица, която е страната "много" на релацията, ако в първичната таблица, която е страната "едно" на релацията, липсва съответен ред. Също така не се допуска да се променят стойностите на колоните на първичния ключ в една таблица, която е страната "едно" на релацията, ако в свързаната таблица има поне един съответен ред. Не се допуска да се изтриват редове от една първична таблица, ако има свързани редове в таблицата с външните ключове;

- дефинирана от потребителя цялост – дава възможност за определянето на специфични бизнес правила, които не могат да се отнесат към някоя от другите категории цялост. Реализира се чрез създаване на ограничения, съхранени процедури и тригери.

7.2.1.3. Разработване на софтуерните модули (units)

Софтуерните модули се разработват от програмистите на базата на софтуерните спецификации и дизайн, при спазване на вътрешните конвенции за кодиране.

Планът на проекта е основата за наблюдение на дейностите и за предприемане на коригиращи действия. Развитието на проекта се контролира главно чрез сравняване на реално получения продукт и усилията и средствата, вложени в изпълнението на задачата с времевата диаграма (schedule) на плана по зададените вътрешни срокове или контролни точки от структурирането на задачите. Когато изпълнението се отклонява съществено от плана, трябва да се предприемат подходящи коригиращи действия. Едно отклонение е значително, когато, ако се остави нерешено, ще възпрепятства постигането на целите на проекта. Коригиращите действия включват:

- Промяна на начина на работа;
- Добавяне на ресурси;
- Смяна на процесите;
- Преразглеждане на рисковете;
- Промяна на изискванията;
- Преразглеждане на оценките и плановете;
- Предоговаряне на споразуменията.

7.2.1.4. Процедура за управление на програмния код

Предлаганата процедура за управление на програмния код ще се прилага за всички резултати от изпълнението на настоящия проект - версиите на софтуерения продукт и свързаните документи представящи резултатите на изпълнението на дейностите по проекта. Функционалността нужна за прилагане на процедурата се осигурява от система за контрол на версиите (SVN) и от архивираща (back-up) система.

Чл.36 а, ал. 3 от ЗОП

Изходният код (Source Code) разработван по проекта, може бъде публично достъпен онлайн като Софтуер с отворен код от първия ден на разработка, чрез използване на система за контрол на версиите.

Разработките по проекта ще бъдат задължително подаване към система за контрол на версиите. Освен кодът, цялата документация и отчетни материали ще бъдат качвани в хранилището.

Ще бъде предвидено използването на система за контрол на версиите и цялата информация за главното копие на хранилището, прието за оригинален и централен източник на съдържанието, да бъде достъпна публично, онлайн, в реално време.

Изпълнителят ще поддържа еднакви версии на софтуера за продуктивната и тестовата среда. Новите функционалности ще се инсталират на продуктивния вариант на системата само след като са преминали успешно изпитанията в тестовата среда.

Изпълнителят ще осигури достъп от минимум един акаунт до софтуерното хранилище (software repository) и последния актуален сорс код на Системата. Тези акаунти ще бъдат използвани от Възложителя за контрол на версиите, допълнително синхронизиране и архивиране на сорс кода на компонентите.

Предимствата на системата за контрол на весиите е това, че предоставя възможност за подробно разглеждане на направените промени, които са довели до създаването на нова версия, за сравняване на версиите една с друга и за връщане на по-стара версия като актуална.

В основата на всяка една система за контрол на версиите, както е и при Subversion, стои хранилището на информация (repository). При постъпване на нова версия на файл в хранилището, старата му версия се съхранява с дата и час или с други думи, съхраняват се всички версии на даден файл в хронологичен ред. Системата съхранява също и автора на всяка версия - по тази причина при свързване с хранилището всеки потребител се идентифицира с име и парола. Потребителите имат различни права - или да редактират файла (т.е. да запазват нови версии в хранилището) или само да четат данни. Тъй като повечето софтуерни продукти могат да работят само с по една версия на даден файл, а не с всички наведнъж, потребителят получава работно копие (working copy) на нужните му файлове, което съдържа изисканата от него версия от хранилището (в типичния случай - най-актуалната) и работи с него.

Системата за контрол на версиите е ценна не само за съхраняването на програмен код, в нея успешно могат да се съхраняват и документите изготвени при разработката на даден проект – файлове с последователнлните версии на планове, спецификации, документация на системата. Поддържането на такива версии в хранилището има предимство пред съхраняването им във файлови сървъри, защото се запазва точната верига на промени по един документ, а не само съхранението му към момента на копиране в папката, като по този начин може да се възстанови версия на документ към даден момент във времето.

Управление на конфигурацията

Управлението на конфигурацията е една от фундаменталните дейности от софтуерното инженерство. Ролята на управлението на конфигурацията е да управлява множеството от документи, обекти, софтуер и хардуер с цел получаване на краен продукт. Управлението на конфигурации и промени засяга и двете основни дейности на проекта.

Чл.36 а, ал. 3 от ЗОП

Чл.36 а, ал. 3 от ЗОП

Инструментите за конфигурация на управлението предлагат възможности за:

- **Контрол на версиите** – поддържа история на промените, направени на компонент и се развива във времето. Позволява също и достъп до определена версия, не просто последната;
- **Паралелна разработка** – няколко разработчика могат да правят различни промени на един и същ компонент по едно и също време. Тези промени по – късно се обединяват и възникналите конфликти се разрешават ръчно или автоматизирано.

Управлението на конфигурациите включва три основни дейности:

- Определяне елементите на конфигурацията;
- Контрол над съдържанието на елементите;
- Проследяване на статуса на конфигурацията.

Средства за контрол на версиите и хранилище с публичен достъп

Изпълнителят предлага да инсталира в средата на Възложителя система за контрол на версиите Субвършън (Subversion SVN). SVN управлява ключовата информация за разработката на системата, процесите на поддръжка и инсталиране и база за потенциално преизползваеми артефакти. Хранилището за код на SVN може да бъде конфигурирано за предоставяне на публичен достъп до програмния код.

SVN дава възможност за:

- Поддържане версии на директориите и принадлежащите им файлове;
- Реална история на версиите – възможни са копирания и преименувания на файловете. Всеки нов файл започва със собствена чиста история, няма наследяване при преименуване;
- Атомарни предавания - при набор от модификации или всички се записват, или всички се отказват. Това позволява разработчиците да изграждат и предават промени като логически обединения и премахва проблемите от частичен запис;
- Версии на метаданните - всеки файл и директория има собствен набор от свойства, асоциирани с тях. Подържат се и версии на тези свойства;
- Избор на нива от мрежата - SVN стандартно използва HTTP протокол. Може да се модифицира и за използване по SSH протокол;
- Консистентно управление на данните - SVN показва разликите във файловете използвайки бинарен алгоритъм, който работи еднакво както за текстови, така и за бинарни файлове. И двата типа се пазят в базата с данни в компресиран вид;
- Ефективно създаване разклонения и тагове;
- Използва механизъм на директно копиране и създаване на твърди връзки.

Чл.36 а, ал. 3 от ЗОП

Управление на промените

Управлението на промени предлага процедури за контрол над исканията, които се смятат за отклонение от договорената основна линия на проекта. Тези процедури се прилагат към всички видове искания за промени, засягащи документи от планирането на проекта, потребителски изисквания или контрола на грешките.

Исканията за промени се повдигат и от двете участващи страни и са адресирани към другата страна за оценка и одобрение. С цел избягване на забавяне, което може да доведе до промени във времевите граници на проекта, период от 10 работни дни от регистрацията на искането може да бъде приет за отправна точка при оценката и вземането на решение.

Всеки член на проекта и от двете страни може да повдигне искане за промяна, но то трябва да е съгласувано с отговорниците по качеството.

Основни стъпки на процедурата са:

- Попълване на форма за искане на промяна;
- Анализиране на искането;
- Оценяване на необходимите разходи;
- Прилагане на искането;
- Поддържане на история на исканията.

7.2.1.5. Инструменти за следене на прогреса на развитие на системата и разпределение на задачите

В рамките на проекта ще бъдат изградена среда за управление на задачите и следене на прогреса, интегрирана със среда за регисриране на проблеми с разработената функционалност – в периодите на тестване, а и на последващата гаранционна поддръжка. Цялостното проследяване, регистриране и управление на задачи, регистрирането на грешки, проблеми/несъответствия ще се осъществява посредством система за управление на задачи на Изпълнителя, а именно **Redmine**. Системата ще работи в режим 24/7. Системата ще бъде разделена на два модула:

- Модул за регистриране на задачи към екипа за разработка – модула представлява проект в системата Redmine, в който ще имат достъп само участниците в екипа по реализация. Задачите в този проект се регистрират от ръководителя на екип и са регистрирани или на база специфицирана задача от плана на проекта (категория Feature), или на регистрирана грешка или проблем от тестване или експлоатация на системата (категория Bug). Във втория случай се регистрира връзката за съответстващия таск във втория модул.
- Модул за регистриране на грешки/проблеми/несъответствия – в този модул могат да се регистрират тестери от екипа на Изълнителя, потребители и ИТ служители от страна на Възложителя, които регистрират открити грешки и несъответствия. Заявките се насочват към бизнес анализатора, който извършва първоначалния анализ и дефинира задачи в проекта за задачи към разработчиците. Ръководителя на екипа

Чл.36 а, ал. 3 от ЗОП