Софийски Университет “Св. Климент Охридски”

**Факултет по Математика и Информатика**

**Катедра “Информационни Технологии”**

# Д И П Л О М Н А  Р А Б О Т А

Тема: WEB Базирана система за проверка на лабораторни резултати през Интернет

**Дипломант:** Ивелин Валентинов Андреев

**Специалност:** Информатика

**Специализация:** Био- и медицинска информатика

**Факултетен номер:** M21766

**Научен ръководител:** доц. Антоний Попов

**Консултант:** Динко Ламбов – Гамаконсулт

**София 2007 г.**

# Съдържание

# Увод

Съвременните болнични заведения и лаборатории функционират под постоянния натиск на държавните органи и най-вече обществото непрекъснато да повишават ефективността си. Това дава основание да смятаме, че в скоро време за всички лабораторни информационни системи(ЛИС) ще бъде почти задължително да разполагат с интерфейс или да бъдат интегрирани с вече съществуващи или планирани болнични информационни системи(БИС) и да участват в изграждането на единно електронно досие на пациент.

Чрез електронното медицинско досие се съхранява, актуализира, а при нужда и предоставя медицинска информация на лекаря лекуващ пациента. Същевременно тази жизнено важната информация може да бъде мигновено достъпна от всяка точка на земята с помощта на Интернет. Възможността за достъп по всяко време на денонощието и от всеки, който е оторизиран, до личното досие на пациент, скъпоценното време и финансови разходи, които се спестяват са факти, които сами по себе си не се нуждаят от допълнителен коментар, за да послужат в подкрепа на тезата.

Често срещан недостатък, проявил се в световната практика в процеса на употреба на ЛИС и интеграцията им с вътреболничните такива, е съвместната работа и комуникацията помежду им. Често разработката на двата типа решения се извършва от независими софтуерни компании, което налага обмяна на данни чрез външни интерфейси. Потенциалните грешки при този подход логично се превръщат в източник на безпокойство, както за медицинските работници и специалистите в обработката на данни, така и за обикновените пациенти (1).

Целта на настоящата дипломна работа е да се създаде уеб-базиран интерфейс към база данни за лабораторни изследвания, който да се интегрира във вече съществуваща лабораторна информационна система и да предоставя на потребителите възможност за отдалечен достъп към резултатите от изследвания на пациенти. В допълнение, настоящата разработка има за цел и да разгледа световния опит в развитието и приложението на системите за лабораторни изследвания и чрез представяне на пример и задълбочен анализ да даде насока за прилагане на някои от добрите практики в развитието на електронно здравеопазване в България.

Разработката налага тематичната подялба на документа в няколко логически обособени глави:

**Глава 1. Обзор на проблемната област -** отговаря на въпросите какво е ЛИС и прави въведение в терминологията на проблемната област; описва някои от най-характерните особености на ЛИС и разглежда примери на реализация.

**Глава 2. Описание на избора на средства и технологии за реализация** – разглеждат се технологиите и подходите избрани за разработка на приложението. Прави се оценка на избора в контекста на задачата, която се решава.

**Глава 3. Реализация на системата –** прави се детайлен преглед на конкретните действия при имплементирането на програмното решение. Обръща се внимание на модулния подход при разработката, класовете, които влизат в състава на модулите и структурата на базата данни.

**Глава 4. Сигурност –** разглеждат се изискванията за сигурност на системата и се прави оценка на предпазните мерки предприети в тази насока.

**Глава 5. Тестване –** анализира се дали е постигната функционалността на приложението. Тестват се скоростта на работа на системата и се коментират отзиви от потребители.

**Глава 6. Заключение и посоки на развитие на системата –** След изложението се прави обобщение на постигнатото и се набелязват насоки за развитие на системата.

# 1. Обзор на проблемната област

Лабораторните информационни системи са клас софтуер, който се занимава с събирането, организирането и съхраняването на данни, резултат на медико-лабораторни процеси (2). Често тези системи разполагат с интерфейси към апаратура или към болнични информационни системи, а подходящият избор на такава е от изключително значение за доброто функциониране на всяка лаборатория.

Поради важността си ЛИС често са включени като част от цялостно решение и до голяма степен определят качеството на грижата за пациента. Системите намират приложение, както в среда на  хоспитализация така и за амбулаторни изследвания., като функциите им включват, но далеч не се изчерпват със:

- Регистрация на пациент;
- Заявка за изследване;

- Провеждане на изследване;
- Описване на резултатите;
- Рапортуване;
- Попълване на досие с изследвания за пациент;
- Попълване на досие с изследвания за лекар.

Приложението разработено за целта на дипломната работа има за цел да покрие всички изисквания към софтуерен продукт с комерсиално предназначение и да бъде в последствие интегрирано със системата "Гама Мултилаб", разработена от фирма Гамаконсулт и внедрена в някои от най-престижните болници в страната: УМБАЛ "Света Екатерина" - София, МБАЛ "Света Марина" – Варна, МБАЛ "Царица Йоана" – София, МБАЛ "Света Анна" – София, ВМА – София, МБАЛ "Доверие" – София, МБАЛ "Токуда" – София и др.

## *1.1. Гама Мултилаб*

Системата "Гама Мултилаб" е разработена с цел да осъществи автоматизирано управление на дейности, свързани с процеса на регистриране, контрол и установяване на разходите на отделните клинични услуги. По този начин системата допринася за постигане на автоматизирано съхранение на резултатите, прилагането им в електронно досие на пациента и оптимално използване на ресурсите в параклиничните болнични звена (3).

Модулът регистрира извършваните изследвания и процедури, в лабораторните звена на болницата: клинична лаборатория, микробиологична лаборатория, патолого-анатомия, както и ако има обособени звена като имунологична лаборатория, токсико-химия, серология, микология, паразитология и вирусология.

Данните, които се регистрират, стават част от медицинското досие на пациента., като връзката е двустранна и при подходяща оторизация, от лабораторния модул има възможност за достъп до данни за приема, поставената диагноза, провежданият курс на лечение, както и данни от предишни посещения на пациента в болничното заведение.

Заявките за изследване могат по избор на лечебното заведение да се получават по електронен път и на хартиен носител. Пробите от пациента се описват на мястото, на което се вземат (в болничните отделения, кабинети или самата лаборатория), след което се поставя бар код на контейнера съдържащ пробата. Автоматичното изчитане и отнасяне на пробата към пациента придава допълнителна сигурност за точността на резултатите, а връзката на

системата с лабораторната апаратура намалява възможността за неточности и облекчава работата на звената с голям обем на дейност.

Автоматизирано е изготвянето на искове към НЗОК за заплащане на извършените изследвания по договор.

## *1.2. Електронно здравеопазване(E-Health)*

В днешния динамичен свят става все по-наложително организациите да бъдат отворени за достъп 24 часа в денонощието, 7 дни в седмицата. Това с особена сила важи за заведенията и фирмите предлагащи здравна помощ. Ето защо бавно, но уверено в нашето ежедневие неумолимо си пробива път едно до скоро съвсем непознато понятие - е-здраве. Електронното здравеопазване (E-Health) е в полза на всички нас - както здравните заведения и осигурителните институции и държавата, така и на всеки човек, като консуматор на този продукт - здравната помощ (4).

Жизнено важна услуга, която всеки се нуждае - това е необходимостта да бъде обслужен бързо и навременно където и да се намира, както при профилактичен преглед , така и в случай на спешна, неотложна помощ. За тази цел медицинските лица имат нужда от пълните данни на пациента - а именно:

- Име и информация за връзка с личния лекар
- Предварителна здравна информация
- Алергии и реакции
- Предписани лечения и изпълнението им
- Резултати от тестове, диагнози и състояния
- Данни за прегледи и манипулации
- Инвалидност и заболявания
- Данни за хирургическа намеса, лечения и терапии
- Информация за диети
- Вид здравна осигуровката, плащания по нея, удръжки

Съгласно съвременните разбирания електронното здравеопазване представлява комплекс от мерки, базирани на организационна, технологична и правна рамка, които обхващат целия аспект на функциониране на здравната система. С въвеждането му се очаква

значително да се подобри качеството на оказваните здравни услуги, а наличният финансов ресурс да се използва по-ефективно. С електронното здравеопазване ще се постигне и по-голяма прозрачност в системата и така ще е послужи и като инструмент за борба с корупцията.

Електронното здравеопазване е бързо развиваща се област, в която чрез използване на съвременни информационни и комуникационни технологии си взаимодействат медицинската информатика, общественото здравеопазване, предлагането на здравни услуги и информация (5). С помощта на технологиите лекарите ще могат да получават повече информация, както за новите тенденции в медицината, така и за своите пациенти, а гражданите - да имат достъп до повече информация за своите права като здравноосигурени и за възможностите на лечебните заведения да извършват различни медицински услуги.

Основните принципи на електронното здравеопазване, които е необходимо да залегнат в основата на всяко информационно решение са:

- **Ефективност** - да се повиши качеството и обема на предлаганите услуги в здравеопазването, като при това се запазят или намалят разходите;
- **Качествено здравно обслужване**;
- **Медицина, базирана на доказателства -** електронното здравеопазване трябва да подпомогне вземането на медицински решения;
- **Равно-поставен достъп –** предоставяне на възможност на всички граждани от различни социални слоеве за достъп до здравни услуги и медицинска информация чрез Интернет;
- **Оперативна съвместимост -** на информационните системи;
- **Еднакво качество -** на здравните грижи.

От направените от правителствените органи задълбочени изследвания и анализи става явна необходимостта от приемане на национална стратегия за внедряване на електронното здравеопазване в България и предприемане на конкретни действия с цел нейното изпълняване в следващите пет години.

Все по явна е необходимостта от свързаност на национално ниво и от изграждане на инфраструктура, която да свързва здравните заведения. Необходимо е утвърждаване на национални стандарти и политики за сигурност и конфиденциалност при обмен и съхранение на медицинската информация. Изграждането на теле-медицински приложения и системи

трябва да бъде съобразено с правилата за оперативна съвместимост. Наложително става и изграждането на ефективни здравни портали и центрове за съхранение на медицински данни. Накрая, но не на последно място - утвърждаване на единна лицензионна политика в системата на здравеопазването за използваните софтуерни решения и продукти.

Стратегия предвижда изграждането на електронни досиета и електронни здравни карти, каквито да има всеки български гражданин, и които да са част от интегрираната информационна система на здравеопазването. Гражданинът, нуждаещ се от здравна помощ, представя електронна карта, чрез която става възможен достъпът до необходимите здравни услуги, а информацията се отразява от здравните специалисти в интегрираната електронна система на здравеопазването. На тази база се формират отчети към различните институции и след необходимите проверки извършените дейности се заплащат. Чрез отчетите, които съдържат медицински и финансова информация, Министерството на здравеопазването и Националната здравноосигурителна каса получават в реално време необходимата информация, което позволява по-ефективно управление на здравната система.

В контекста на възприетите принципи в националната стратегия, страната ни си поставя за цел развитие в няколко направления на електронното здравеопазване(6):

- **Системи и услуги** - предназначени за сектора по здравеопазване подпомагат напредъка при медицинската научно-изследователска дейност, ефективното управление и разпространение на медицински знания;
- **Ускорено внедряване на ИКТ** (информационните и комуникационните технологии) **-** след извършване на обстоен анализ на работните процеси в здравеопазването
- **Защита на данните -** Въвеждането на функции по съхранение и обмен по електронен път на регистри, здравни досиета, епикризи и др., съдържащи лични данни и здравна информация предполага адекватна законова защита на тази информация. За да се защитят по адекватен начин правата и интересите на гражданите, свързани с опазването на здравната информация е нужно установяването на ясни правила за достъп и съхранение на такива данни и в електронен вид.
- **По-добро здравно образование** - превенция чрез информираност, възможност за активно участие на пациентите във вземането на решения относно тяхното здраве;
- **Съдействие между професионалистите** – бърз и лесен достъп до информация, диагностика и извършване на сложни интервенции от разстояние, както и достъп до специализирани ресурси за образование и обучение;

- **Нормативна база -** За изграждането на модерна здравна система, базирана върху използването на информационни и комуникационни технологии изисква нормативна уредба, която да гарантира правата и интересите на гражданите и на всички други участници в системата на здравеопазването. Националната правна уредба следва да бъде съобразена с европейските изисквания в областта на електронното здравеопазване.
- **Защита на интелектуалната и индустриалната собственост -** Използването на информационни и комуникационни технологии в здравния сектор е свързано с необходимостта от гарантиране защитата на авторските и сродни права върху здравни разработки, статии и информация, търговски марки на лекарства, патенти и др.

## *1.3. Примери на лабораторни информационни системи*

### 1.3.1. iLab Online

Един от конкурентите в България на системата "Мултилаб", което да позволява отдалечен достъп през глобалната мрежа Интернет до лабораторни резултати, системата е "iLab Online". Лабораторната информационна система (ЛИС) iLab е първото решение от подобен характер представено на българския пазар и е достъпно за крайни клиенти от края на 2000 г. "iLab Online" се интегрирана с ЛИС "iLab"(7) и се използва в редица медицински центрове и болници, както и в едни от най-модерно оборудваните лаборатории у нас Цибалаб(8).

"iLab" е интегриран софтуерен продукт, автоматизиращ всички процеси в клиничната лаборатория и обхваща както лабораторната работа, така и финансовите потоци. Системата управлява лабораторната апаратура, в различна степен и според възможностите на конкретния апарат като поддържа едно- и двупосочни връзки с почти всички разпространени в България апарати. Тя е пригодена за работа и с НЗОК и предлага модул за достъп до резултати на пациенти през Интернет.

Преди първоначалното си пускане, "iLab" е била разработена в тясно сътрудничество с експерти от Медико-диагностична лаборатория (МДЛ) Цибалаб. До сега системата е претърпяла закономерно развитие от първоначалния си вариант, като през това време са били изчиствани бизнес процесите и се е опростявала логиката на системата с цел тя да стане по-интуитивна и достъпна за крайния потребител, като същевременно е останал непроменен стремежът да се изпълняват точно функционалните изисквания.

ЛИС “iLab” е пълно-функционална интегрирана лабораторна информационна система., която постоянно търпи развитие и добавя нови и нови функции. Предвижда се в бъдеще да бъдат разработени компоненти, предназначени за други разработчици на софтуер, които да спомагат вграждането на системата в други болнични и лабораторни информационни системи. Към днешна в ЛИС “iLab” е реализирано следното:

- Управление на финансовите потоци, разнообразни финансови справки и управление на фискален принтер;
- Неограничен брой видове пациенти (частни, каса, по договор и т.н.);
- Нива на достъп според вида потребител;
- Типизация на резултатите;
- Баркод етикиране на материалите;
- Едно- и двупосочни връзки с лабораторна апаратура;
- Работа с НЗОК;
- Достъп до резултати на пациенти през Интернет.

Предимствата на “iLab Online” като първа по рода си система на българския пазар са очевидни, защото реализира по един доказал себе си във времето механизъм на отдалечен достъп до данни от пациентите.

Основни недостатъци на системата са липсата на графики и визуално представяне на резултатите в частта за отдалечен достъп през Интернет. Основата на уеб-интерфейса е поставена през далечната 2001 г. И той е реализиран с вече остарели технологии (ASP), които не позволяват скалируемост, междуплатформеност, бърза и лесна поддръжка и др. Едновременно с това подходът за връзка с базата данни за извличане на информация за резултати е базиран на конкатениране на символни низове, което отваря възможности за злонамерени потребители (кракери) да придобият достъп до данните в системата. Настоящият избор на система за управление на бази от данни(СУБД) е Microsoft SQL Server 2000 и също се нуждае от модернизация с по-новия и бърз Microsoft SQL Server 2005. Това би намерило отражение при работа с големи масиви, каквито са в случая таблиците със стойности за показатели от изследвания, където броят на записите се измерва със стотици хиляди и милиони. Вероятно това е довело до появата на друг недостатък, който не е за пренебрегване - а именно, че до крайните потребители се предоставя достъп до данни с давност максимално шест месеца, което не дава възможност за дългосрочно проследяване на даден показател.

### 1.3.2. SoftWeb

Друг пример за лабораторна информационна система с достъп през Интернет е продуктът “SoftWeb” на американската компания “SCC Soft Computer”(9). Компанията е пионер на пазара още от далечната 1979 г., когато дори в глобален мащаб продукти за областта на електронното здравеопазване са били истинска революция.

“SCC Soft Computer” предлага пълна гама напредничави решения, реализирани с последни и най-съвременни технологии, притежава най-големия екип в света за разработка на софтуер в областта на ЛИС и в течение на времето си е извоювала име с отлична репутация, като дори може да се твърди, че де факто компанията определя стандартите в индустрията.

“SoftWeb” е решение предназначено да обслужва предимно нуждите на лекарски кабинети. Това уеб-базирано приложение позволява както въвеждане на заявки, така и получаване на данни за резултати в реално време през Интернет. Чрез своя интерфейс и реализация “SoftWeb” предлага практични и мощни решения за лекарския кабинет, които освен това са удобни и интуитивни. Основните функционалности включват:

- Извличане на информация за пациенти по зададени критерии, типове изследвания, местоположение и др., като чрез такива гъвкави критерии на търсене се филтрира излишната информация и се представя поглед върху данните от различни точки;
- Оцветяване на състоянията, които се отклоняват от границите на нормалните стойности за съответния показател на изследване. Това предупреждение в реално време, което лекарят получава за състоянието на пациента увеличава сигурността и намалява времето за констатиране на патологични състояния;
- Текстови и графични изгледи на клинични резултати, които подпомагат анализа и установяването на корелационни зависимости и поставяне на диагноза;
- Настолен интерфейс позволява бърз достъп и лесен през добре познатата среда на Windows;
- Уеб-интерфейс позволява бърз достъп от болницата, от дома или от лекарския офис чрез използване на сигурен канал за достъп през Интернет;
- Обобщаващи рапорти, с възможност за едновременно показване на резултати от различни отделения. Рапортите могат да бъдат изпратени и по електронна поща, само с натискане на един бутон;
- Възможност за разпечатване на резултати, дори и отдалечено, което елиминира необходимостта от индивидуални заявки за получаване на справки от клиентите.

“SoftWeb” е бърза, надеждна и съвременна система, която подпомага работата в лекарския офис. С нейното използване значително се намаляват усилията за много рутинни дейности, които лекарят ежедневно изпълнява. Приложението спомага организирането на пациентите, дава възможност за множество справки в графичен и текстов формат, а интуитивният му интерфейс го прави разбираемо и лесно достъпно.

Разработката за целите на настоящата дипломна работа е значително повлияна в заимстване на идеи и подходи от изложената по-горе система, която е приета като еталон и положителен пример, наложени от световен лидер в областта на ЛИС.

## *2. Описание на избора на средства и технологии за реализация*

### 2.1.1. Функционални и нефункционални изисквания към приложението

- Да се реализира уеб-сайт, който да се интегрира със съществуващата система “Мултилаб” разработена от фирма Гамаконсулт, и да предоставя възможност за достъп до информацията за даден пациент през Интернет;
- Да се осигури сигурност на информацията. Всеки потребител да разполага с уникални име и парола, което да позволява достъп само до резултатите за този потребител, кодирани със съответната парола;
- Възможност за достъп до минали резултати на пациента – само чрез идентификация със съответната парола;
- Възможност за разпечатване на резултати от сайта.

### 2.1.2. Предимства и недостатъци на достъпа до резултати през Интернет

Достъп навсякъде и по всяко време, дори от мобилни и портативни устройства.; междуплатформеност и независимост от операционната система на клиентския компютър; леснота на инсталация и поддръжка, защото приложението се поставя на едно единствено място – сървъра, вместо да се обслужват хиляди компютри. Това са основните предимства на програмирането за Интернет

Уеб-приложението позволява свързаност и отдалечен достъп до данни. Това от своя страна означава, че без излишна загуба на време, без да е необходима инсталация на клиентската машина, потребителят може бързо и лесно да добие необходимата информация. Единственото изискване е почти символично - да има наличен уеб-браузър.

Приложението "Мултилаб Уеб" е оптимизирано за екран със стандартна разделителна способност 1024*768 пиксела и уеб-браузър Internet Explorer 6+. Въпреки това, целта е била то да бъде достъпно за максимален брой хора, поради което за изпълнението на всички функционални изисквания са използвани подходи, които не нарушават съвместимостта с други широко разпространени браузъри като Mozilla Firefox и Opera.

Като всеки подход, разработката на дадена система като уеб-приложение също има своите ограничения. Основен недостатък е липсата на възможност за несвързан режим на работа(Offline), когато няма връзка с Интернет. Наборът от средства за постигане на изискванията към системата не е голям, което довежда до трудно и скъпо реализиране на напълно обикновени действия за настолните приложения. Също така друго неприятно ограничение е извършването на входно-изходни операции с хардуера на клиентската машина. Накрая и не на последно място трябва да отбележим, малкия процент граждани на България, които имат достъп до компютър и Интернет.

### 2.1.3. Технологии и средства за реализация

Софтуерният продукт, разработен за целите на настоящата дипломна работа е реализиран с модерни, широко достъпни и доказали себе си технологии и средства. Това са:

**Основни средства**

- Развойна среда: Microsoft Visual Studio .NET 2005 v.8.0 Professional
- Платформа: .NET Framework 2.0
- Програмен език: Visual C#
- Потребителски интерфейс: ASP.NET 2.0
- Уеб сървър: Internet Information Server (IIS) 6
- Система за управление на бази данни (СУБД): MS SQL Server 2005 Express
- Среда за работа със СУБД: Microsoft SQL Server Management Studio Express
- Контрол на версиите и програмния код: Microsoft Visual SourceSafe 6.0

- UML моделиращо средство: Sparx Systems, Enterprise Architect 6.5
- Оптимизационно средство: AutomatedQA, AQtime 5
- Средство за следене на клиентския компютър: Web development Helper 0.6

**Други**

- Библиотека за работа с 2D графика: ZedGraph (10)
- Допълнение(Plug-in) към развойната среда: VS 2005 Web Application Project
- Програмно средство за генериране на PDF рапорт: Crystal Reports 2005

### 2.1.4. Избор на платформа

За разработката на приложението е използвана платформата ".NET Framework 2.0", програмният език "Visual C#" (11) и най-добрата развойна среда за езика към момента – "Microsoft Visual Studio .NET 2005".

.NET Framework е съвременна платформа за разработка и изпълнение на приложения, която предоставя програмен модел, стандартна библиотека от класове(Framework Class Library) и среда за контролирано изпълнение на програмен код(Common Language Runtime). Тя поддържа различни езици за програмиране и позволява тяхната съвместна работа(12).

.NET приложенията се пишат на езици от високо ниво (C#, VB.NET, Managed C++ и др.) и се компилират до междинен език от ниско ниво, наречен IL (Intermediate Language). По време на изпълнение IL програмите, или т. нар. управляван код(managed code), се компилират до инструкции за текущата хардуерна архитектура, съобразени с текущата операционна система, и след това се изпълняват от микропроцесора.

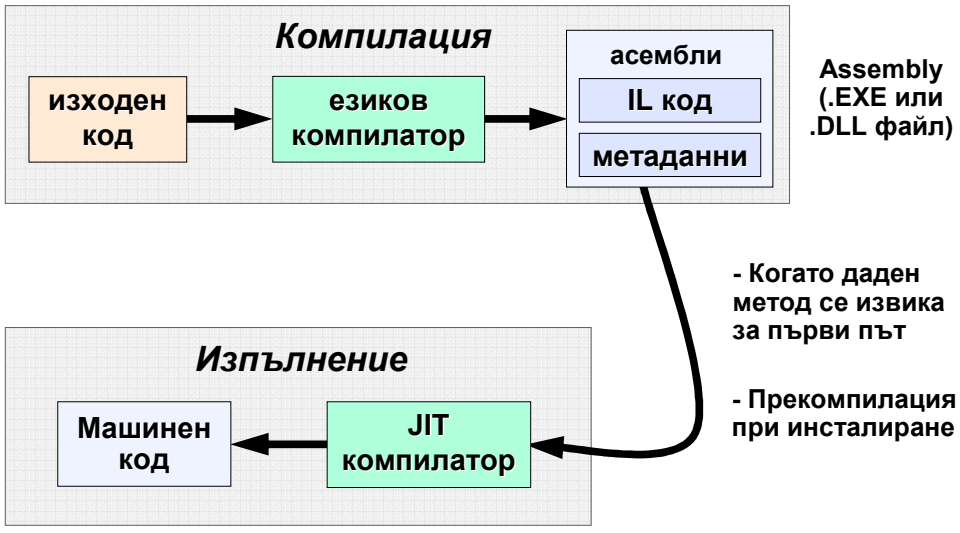
Можем да разделим .NET Framework на два основни компонента:

- **Common Language Runtime (CLR)** – средата, в която се изпълнява управляваният код на .NET приложенията. Представлява виртуална машина, която контролирано изпълнява .NET кода и осигурява различни услуги, като управление на сигурността, управление на паметта и др. Задачите които CLR изпълнява са:
  - Изпълнение на IL кода;
  - Управление на паметта и ресурсите;
  - Управление на сигурността и осигуряване безопасността на типовете;;

- Управление на изключенията, управление на конкурентността;
- Взаимодействие с неуправляван код;
- Подпомагане процесите на дебъгване и оптимизиране.

При компилация (фиг.1) на изходния код от някой от програмните езици за платформата, получаваме асембли. То представлява изпълним файл, съдържащ .NET управляван код и мета данни, които описват съдържанието на асемблито - имената на класовете и типовете, информация за членовете на класовете (методи, полета, свойства и други). При изпълнение на дадено асембли, CLR го зарежда в паметта, анализира мета данните му, извършва проверки на кода – дали е коректен спрямо IL стандарта, дали има необходимите права за изпълнение и др. След това с помощта на JIT (Just-In-Time) компилатор междинният език се компилира до машинен код за текущия процесор. Компилираният вече код след това се изпълнява директно от процесора (не се интепретира).

Предимството на JIT компилацията е, че може да оптимизира кода за текущата хардуерна платформа по най-добрия начин. При неуправлявания код това не е възможно, защото кодът се компилира така, че да работи върху всички процесори, без да използва пълните възможности на текущата хардуерна платформа. По тази причина в някои случаи управляваният код може да е дори по-бърз от неуправлявания, въпреки нуждата от JIT компилация, която отнема време.




**фиг. 1** Компилация на програмен код за .NET Framework

- **Framework Class Library (FCL) -** стандартната библиотека на .NET Framework, в която се съдържат няколко хиляди дефиниции на типове, които предоставят богата функционалност при изграждането на .NET приложения: вход/изход, връзка с бази данни, работа с XML, изграждане на уеб-приложения, използване на уеб-услуги, изграждане на графичен потребителски интерфейс и др.

### 2.1.5. Избор на програмен език

Езикът Visual C# (или за краткост само C#) е съвременен, обектно-ориентиран, силно типизиран език, с широка поддръжка на идеите на компонентно-ориентирания подход за разработка.

C# е наследник на езика C++, но не наследява всичко, а само част от синтаксиса и някои негови силни страни (например полиморфизъм и предефиниране на оператори). C# поддържа синтаксис за дефиниране и използване на свойства и събития, които играят важна роля при дефинирането и използването на компоненти.

Въпреки, че .NET Framework поддържа много други езици, за реализацията е избран езикът C# по няколко причини:

- C# е препоръчваният език за програмиране за .NET Framework. Архитектите на езика са го проектирали специално за платформата .NET Framework, като още по време на дизайна са го съобразили с нейните особености. C# притежава максимално стегнат и ясен синтаксис, наследява простотата на Java, мощността на C++ и силните черти на Delphi.

- Езици като C++, Visual Basic и JScript не са проектирани специално за .NET Framework, а са адаптирани допълнително към него чрез редица изменения и добавки. В следствие на това те запазват някои синтактични особености, които не са удобни при работата с .NET. (14)

- Общността на C# разработчиците е по-добре развита, отколкото на разработчиците на другите .NET езици. Поради голямата популярност на езика C# за него има по-добра поддръжка от инструментите за разработка.

- В България C# е най-популярният от .NET езиците и се използва най-масово в българските софтуерни компании.

### 2.1.6. ASP.NET

ASP.NET е програмна платформа част от .NET Framework за разработка на уеб-приложения. Тя предлага съвкупност от класове, които работят съвместно, за да обслужват HTTP заявки. Също като класическото ASP (Active Server Pages), ASP.NET се изпълнява на уеб-сървър и предоставя възможност за разработка на интерактивни, динамични и персонализирани уеб-базирани приложения , както и за разработка и използване на уеб-услуги.

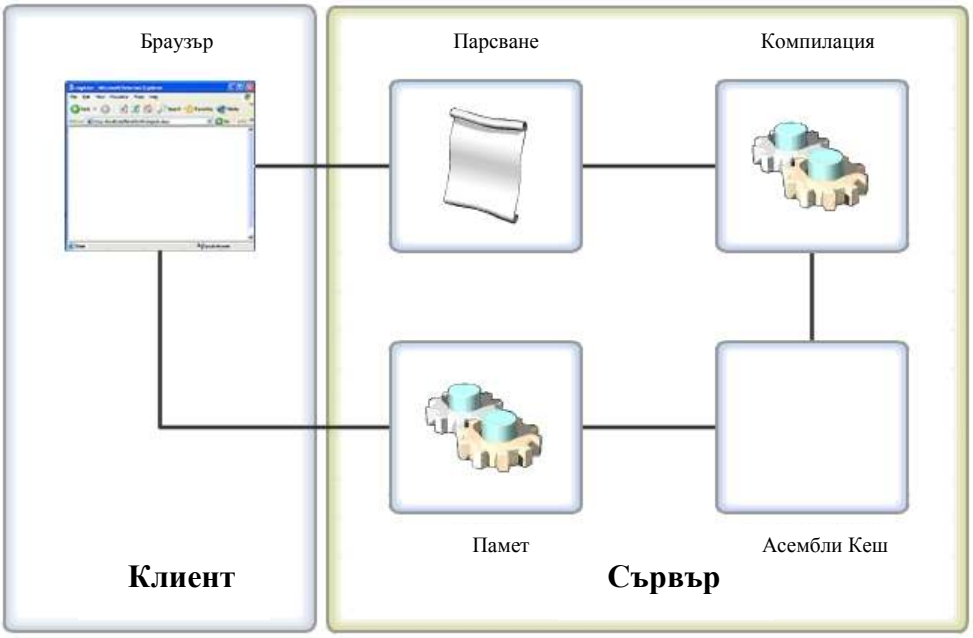
Разликите между ASP и ASP.NET са значителни. Основният компонент на ASP.NET е уеб-формата – абстракция на HTML страницата, която Интернет потребителите виждат в браузъра си. Ключова характеристика на ASP.NET е възможността за разделяне на кода описващ дизайна от кода, реализиращ логиката(code-behind), но нивото на абстракция, включва и богат избор от уеб-контроли, подобни на тези в Windows Forms, и намалява нуждата програмиста да работи с чист HTML код.

ASP.NET уеб-формите се изпълняват от страна на сървъра, като в резултат се генерира HTML код, който да се върне като отговор на клиентската заявката. Този код е пригоден за типа на клиентския браузър, което позволява улеснена разработка на уеб-форми, тъй като те практически могат да работят върху всяко устройство, което разполага с Интернет свързаност и уеб-браузър. За генериране на страницата могат да се извършват обработки, изискващи достъп до бази от данни и до ресурсите на самия сървър, генериращи допълнителни уеб-форми и други.

Моделът на изпълнение (execution model) на ASP.NET е следният (фиг.2):

- Клиентският браузър изпраща HTTP заявка на сървъра;
- Сървърът идентифицира страницата, за която е предназначена заявката и започва да я изпълнява;
- ASP.NET интерпретира кода на страницата;
- Ако кодът не е бил компилиран до асембли, ASP.NET извиква компилатора.

- Средата за изпълнение (CLR) зарежда в паметта и изпълнява междинния код;
- Страницата се изпълнява, като в отговор на клиентската заявка се генерира HTML код. Сървърът връща този резултат на клиента като HTTP отговор.




**фиг. 2** Модел на изпълнение на ASP.NET

### 2.1.7. Избор на СУБД

За физическата реализация на базата от данни на информационната система е избрана релационната система за управление на бази данни (СУБД) Microsoft SQL Server 2005 Express. Тя е бърз, многонишков (multi-threaded), многопотребителски(multi-user) SQL (Structured Query Language) сървър за бази данни, като едновременно с това е и безплатен (free software).

Това е наследникът на MSDE (Microsoft SQL Server Desktop Engine 2000). Той е свободен за сваляне през Интернет, свободен за ползване и дистрибуция, лесен за ползване и напълно съвместим с Microsoft SQL Server 2005, олекотен като инсталация и хардуерни изисквания и същевременно мощен. Напълно свободно и безплатно за неговата администрация и менажиране може да се използва SQL Server Management Studio Express.

SQL е стандартизиран език за извличане и актуализация на информацията, намираща се в бази от данни. Той позволява модифициране на структурата на базите данни; промяна на системните настройки за сигурността; добавяне на разрешения за достъп на даден потребител

до базите данни или таблици; извличане от базата данни на информация; обновяване на съдържанието на базата данни; обработка на транзакции. С помощта на заявки могат да бъдат извършвани широка гама от действия, като избор на записи от базата данни – select, добавяне на нови записи - insert, актуализация съдържанието на съществуващи записи – update и др.

SQL Server 2005 Express Edition е подходящ продукт за бързо изграждане и внедряване на приложения основани на бази данни, позволява лесно вграждане на локална база данни в приложения и поддържане на базови уеб-приложения. Той е безплатна, лесна за ползване, олекотена версия на SQL Server 2005. SQL Server Express е с възможност за безплатно изтегляне, свободно разпространение и е лесен за ползване от разработчиците. Постига се лесно управление на SQL Server Express с помощта на SQL Server Management Studio Express, нов инструмент, специално създаден да извършва основни административни задачи по базата с данни. по-лесно създаването, внедряването и управлението на приложения. Microsoft SQL Server 2005 е изграден на стабилната основа на Microsoft SQL Server 2000 предлага интегрирано решение за управление и анализ на данни, което помага на организациите да:

- Изградят и внедрят приложения, които са по-сигурни, по-надеждни и с по-големи възможности за скалируемост(scalability);
- Увеличат производителността на приложенията,като намаляват сложността при изграждането, инсталирането и управлението на базите данни;
- Подпомогнат програмистите с богата, гъвкава и модерна среда за разработка на по-сигурни приложения;
- Разпространяват данни между множество платформи, приложения и устройства

Предоставят мощни, интегрирани информационни средства, които спомагат за взимането на информирани бизнес решения и повишават производителността в цялата организация

### 2.1.8. Заключение

Подходът, при който за разработка на уеб-базирано приложение се използват разработените от световния лидер Майкрософт технологии - .NET Framework, ASP.NET 2.0 и Microsoft SQL Server 2005 позволяват постигането на един много добър баланс между компонентно ориентираната концепция за бърза разработка (RAD - Rapid Application Development) и постигане на отлични технически характеристики на продукта.

RAD е подход за разработка, при който приложенията се създават визуално, чрез сглобяване на готови компоненти и посредством помощни инструменти за автоматично генериране на голяма част от кода. В резултат приложенията се разработват много бързо, с малко ръчно писане на код и с намалени усилия от страна на програмиста.

При компонентно-ориентираната разработка всеки компонент решава някаква определена задача, която е част от проекта. Компонентите се поставят в приложението, след което се интегрират един с друг чрез настройка на техните свойства и събития. Свойствата на всеки компонент определят различни негови характеристики, а събитията служат за управление на действията, които са предизвикани от него.

Модерните технологии позволяват бърза и лесна инсталация, скалируемост и не на последно място – междуплатформеност. С технологиите, които ASP.NET 2.0 предоставя, е постигнат унифициран потребителски интерфейс с помощта на мастер страници (master pages), които установяват еднакво разположение на контролите в, а посредством Cascading Style Sheets са дефинирани цветове, големини и поведение на елементите от потребителския интерфейс.

Накрая, но не на последно място - обектният модел на ASP.NET 2.0 позволява имплементирането на многоезичен и настройваем интерфейс, при който всеки символен низ, който се вижда от крайния клиент и не може да бъде окачествен като данни, се превежда в реално време от ресурсен файл. С подмяна или корекция на ресурсен файл в широко разпространения XML формат се предоставя възможност за администраторите на сайта бързо и лесно да променят текстовото съдържание на сайта, без за това да се изискват специфични познания в програмирането.

# 3. Реализация на системата

## *3.1. Избор на архитектура на приложението*

Приложението е реализирано с модифицирана трислойна архитектура. Трислойният модел се счита за един от шаблоните на софтуерна архитектура. Тя представлява клиент-сървър концепция, при която потребителският интерфейс, бизнес логиката на приложението и работата с базата данни се извършват от независими модули(13).

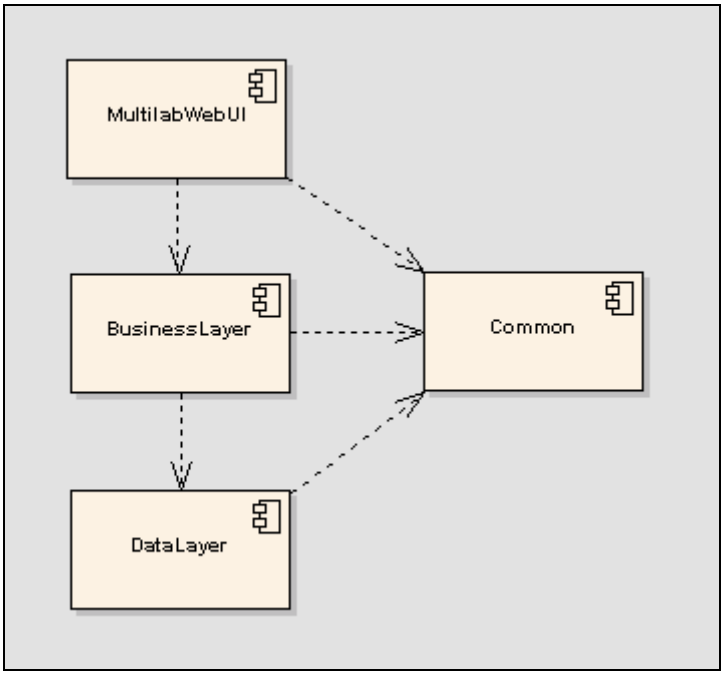
Освен другите предимства на модулния софтуер се счита, че трислойният модел позволява промяна на технологиите или друго обновление на всеки от модулите, без това да повлиява на останалите слоеве, като е необходимо да се спазят само утвърдените интерфейси между модулите. Допустимо е средният, логически, слой да бъде многослоен сам по себе си, което дава пример за N-слоен модел. Фундаментално правило при този тип архитектура е, че презентационният слой никога не комуникира директно със слоя за работа с базата данни, като това прави модела линеен. Това е фундаментална разлика в сравнение с друг популярен при разработката на уеб-приложения архитектурен шаблон - а именно модел-изглед-контролер (MVC - Model-View-Controller).

И двата модела имат за цел да отделят съдържанието от презентацията, но МИК е триъгълен, защото изгледът (View) изпраща заявки за обновление към контролера (Controller), който от своя страна обновява модела (Model), а изгледът сам по себе си черпи информация директно от модела. Хронологично трислойният модел е по-нов и води началото си от разпределените системи от началото на 90-те години, където всеки модул се е изпълнявал на различна платформа.

За целите на приложението стандартната трислойна архитектура е модифицирана до нелинейна четирислойна такава, като е добавен и общ четвърти слой, който съдържа функционалност, обща за останалите модули. По-долу са приложени UML диаграми на зависимостите между компонентите и тяхното разпределение.

## *3.2. Компоненти на приложението*

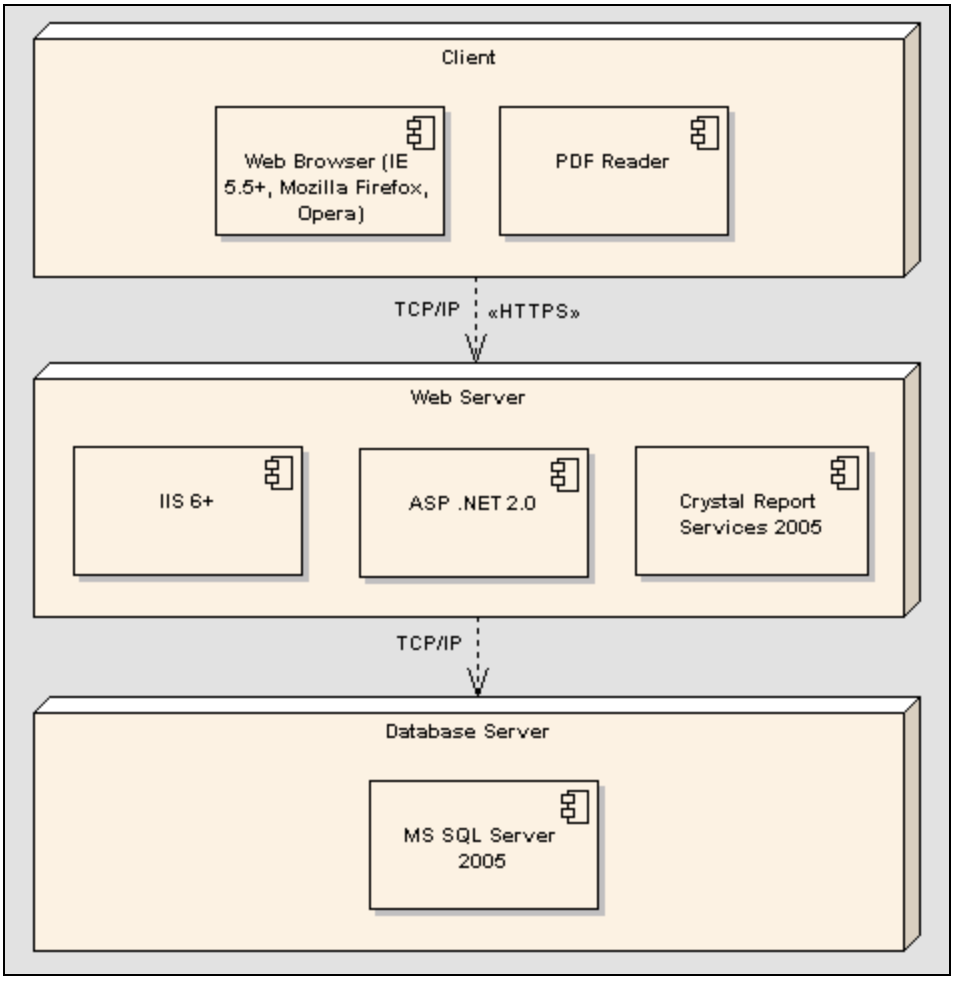
Ще илюстрираме системата с диаграма на компонентите (Component Diagram), която показва частите на софтуера, от които системата се изгражда (фиг. 3). Компонентните диаграми дефинират логически граници на системата, които ни позволяват да извършим семантично групираме на елементите в логически структури (14). Диаграмата има по-високо ниво на абстракция, сравнено с класова диаграма, но има за цел да покаже на високо ниво модулите и тяхната взаимосвързаност.




**фиг. 3** Компонентна диаграма (Component Diagram)

### *3.3. Разгръщане на приложението (Deployment)*

Диаграмата(15) има за цел да илюстрира къде и как се разгръща приложението при инсталация (фиг.4). Физическите местоположения (клиентски и сървърни компютри) и устройства се показват като възли, а вътрешната структура на възел също може да се покаже по-подробно с помощта на такива възли. Артефакти, подобно на външни компоненти и изпълними файлове се показват във възлите за да се даде по-подробна картина на конфигурацията по време на изпълнение.




**фиг. 4** Диаграма на разгръщане (Deployment Diagram)

## *3.4. Хардуерни и софтуерни изисквания за работата на системата*

### 3.4.1. Изисквания към клиента

- Процесор: Pentium II 300 MHz (препоръчително 1 GHz или по-бърз);
- Памет: 128 MB или повече (препоръчително 256 MB);
- Интернет браузър: Internet Explorer 5.5+(Препоръчително IE 7), Mozilla Firefox, Opera; възможност за приемане на authentication cookie от сайта; възможност на отваряне на pop-up прозорци от сайта;
- Дисплей: цветен (препоръчително 16bit цветност или по-висока), резолюция 800x600 (препоръчителна 1024x768);

- Интернет връзка: 56Kbps модем (препоръчително 512Kbps или повече);
- Други: софтуер за визуализация на файлове в PDF формат; идентификационен номер и парола за достъп.

### 3.4.2. Изисквания към уеб-сървъра

- Процесор: Pentium III 1 GHz (Препоръчително Intel Xeon 2 GHz или по-бърз);
- Памет: 512 MB или повече (препоръчително 1 GB или повече);
- Свободно дисково пространство: около 200 MB (за инсталация на .NET Framework 2.0 и допълнителните компоненти);
- Операционна система: Windows Server 2003; Windows XP Professional, Windows 2000 Professional или Server;
- .NET Framework: версия 2.0;
- Уеб-сървър: Internet Information Server 6.0+;
- Други: Crystal Reports 2005 Redistributable.

### 3.4.3. Изисквания към сървъра за бази данни

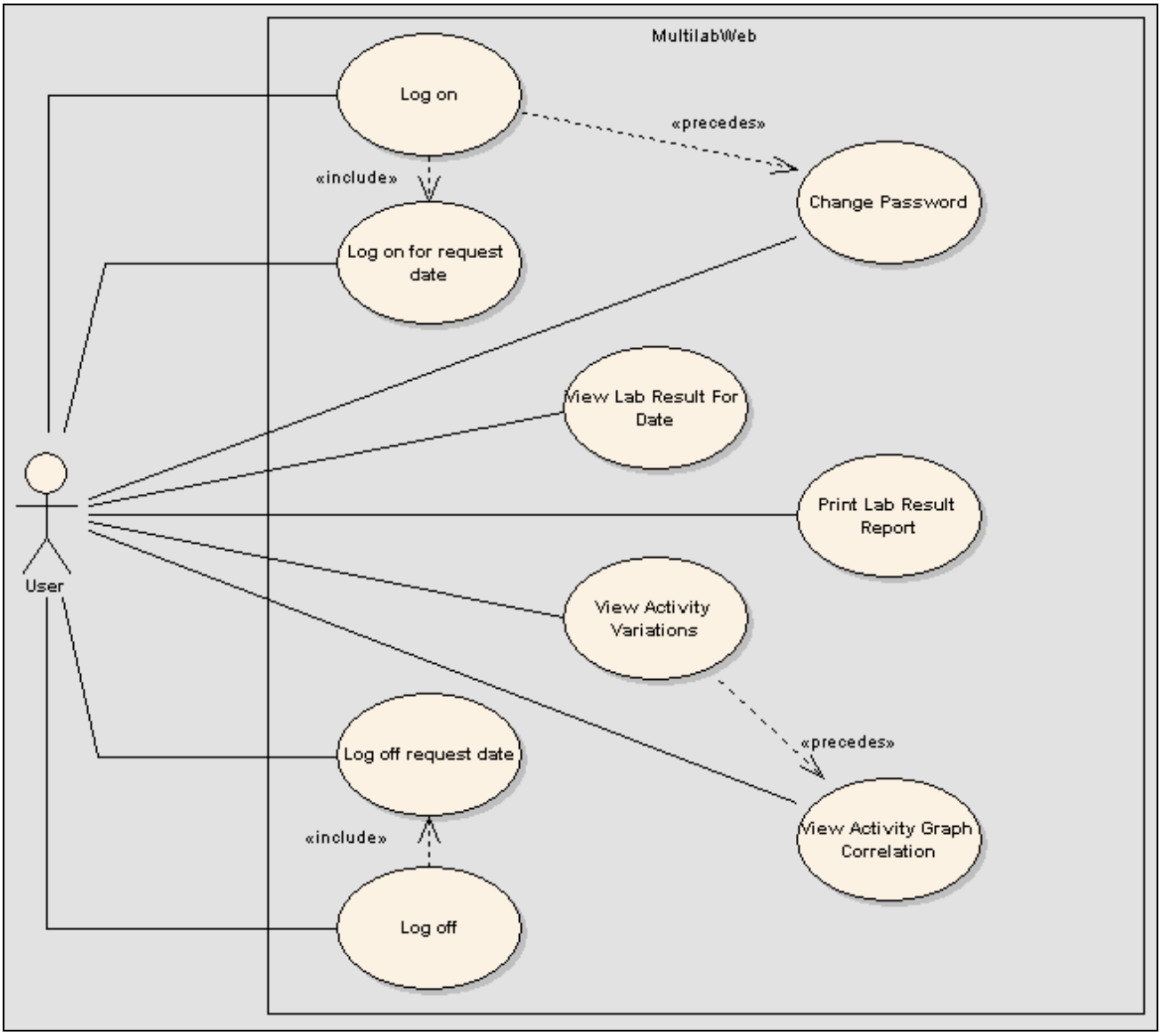
- Процесор: 600 MHz Pentium III-съвместим (препоръчително 1 GHz или по-бърз);
- Памет: 192 MB RAM (препоръчително 512 MB или повече);
- Свободно дисково пространство: около 350 MB за инсталацията, около 425 MB за помощните програми и 150 MB или повече за базата данни (препоръчително 1 GB или повече);
- Операционна система: Windows XP with Service Pack 2, Microsoft Windows 2000 Professional или Server със Service Pack 4, Windows Server 2003 със Service Pack 1;
- Други: .NET Framework 2.0.

## *3.5. Сценарии в системата (Use Cases)*

Диаграмата на сценариите (Use Case Diagram) показва основните сценарии и актьорските роли, които вземат участие в тяхната реализация. Най-общо диаграмата скицира

на високо ниво функционалните изисквания към системата, както и връзките между актьорите и системата (16).

Най-общо актьорите могат да бъдат както потребители на системата с техните допустими роли, така и външни системи или хардуер. Един потребителски сценарий (use-case) е последователност или набор от смислени действия, които семантично могат да бъдат обособени като работа, която се върши в системата. Целта на диаграмата е на високо ниво да покаже как изглеждат процесите в системата от страната на външен наблюдател(фиг. 5).




**фиг. 5** Диаграма на сценариите (Use Case Diagram)

Основните сценарии в системата, които по-долу ще опишем подробно и заедно със заснети екрани от реализацията са:

1. Вход в системата;
2. Оторизация за дата на изследване;
3. Отмяна на оторизацията за дата на изследване;
4. Промяна на парола;
5. Показване на резултати от изследване на базата на критерии за търсене;
6. Разпечатване на резултатите от изследване;
7. Показване на изменението на показател във времето;
8. Графично показване на измененията и отношението на показатели;
9. Изход от системата

### 3.5.1. Вход в системата

Входът в системата се извършва през менюто “Вход” и екрана (фиг. 6.1).

Предпоставки

- Потребителят има валидно потребителско име и активна парола.
- Потребителят не е автентикиран (authenticated) пред системата.

Стъпки

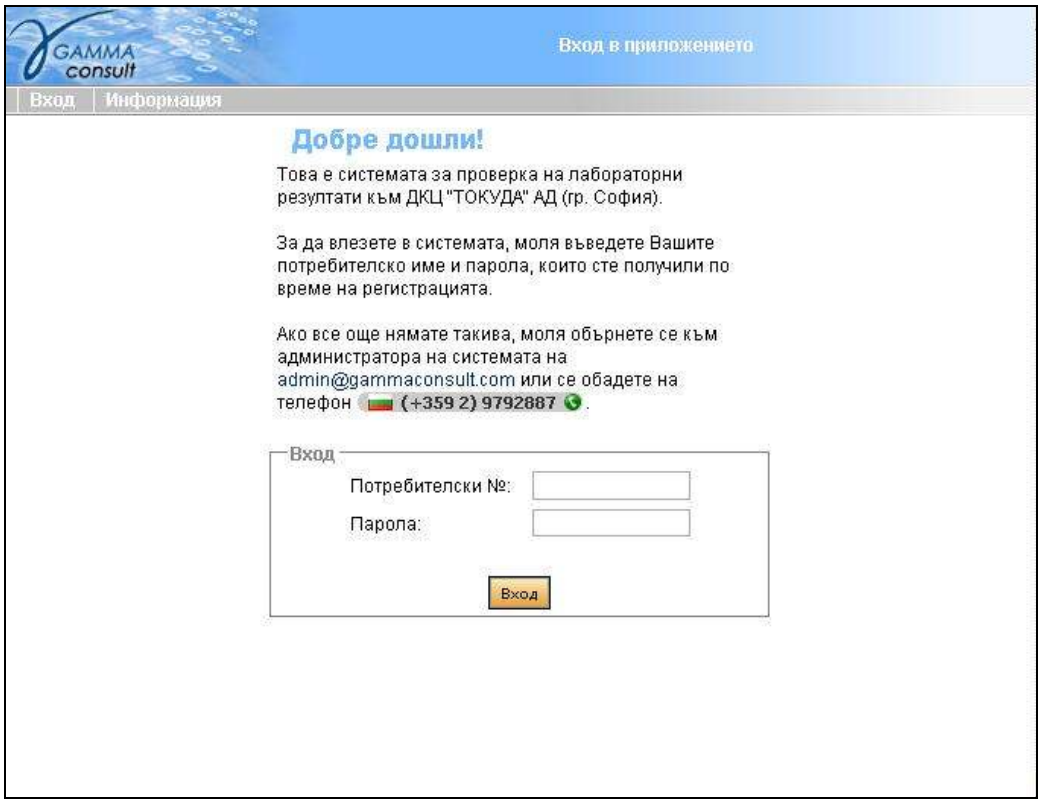
1. Потребителят отправя заявка към сървъра към страницата за вход или към произволна страница на системата от областта, предназначена за автентикирани потребители.
2. Системата извлича от базата информация за лечебното заведение или лаборатория, които притежават системата и показва страницата за вход и данни за заведението.
3. Потребителят въвежда потребителско име, което съвпада с личния му номер от пациентската карта.
4. Потребителят въвежда парола, която е получил при подаване на заявка за изследване или ако е променил паролата, въвежда новата парола.
5. Системата проверява за валидност двойката потребителски номер и парола в базата данни.
6. Системата пренасочва потребителя към страницата с лични данни за пациент.

1. При коректни потребителско име и парола потребителят бива автентикиран в системата и в обратния отговор(Response), който сървърът формира в

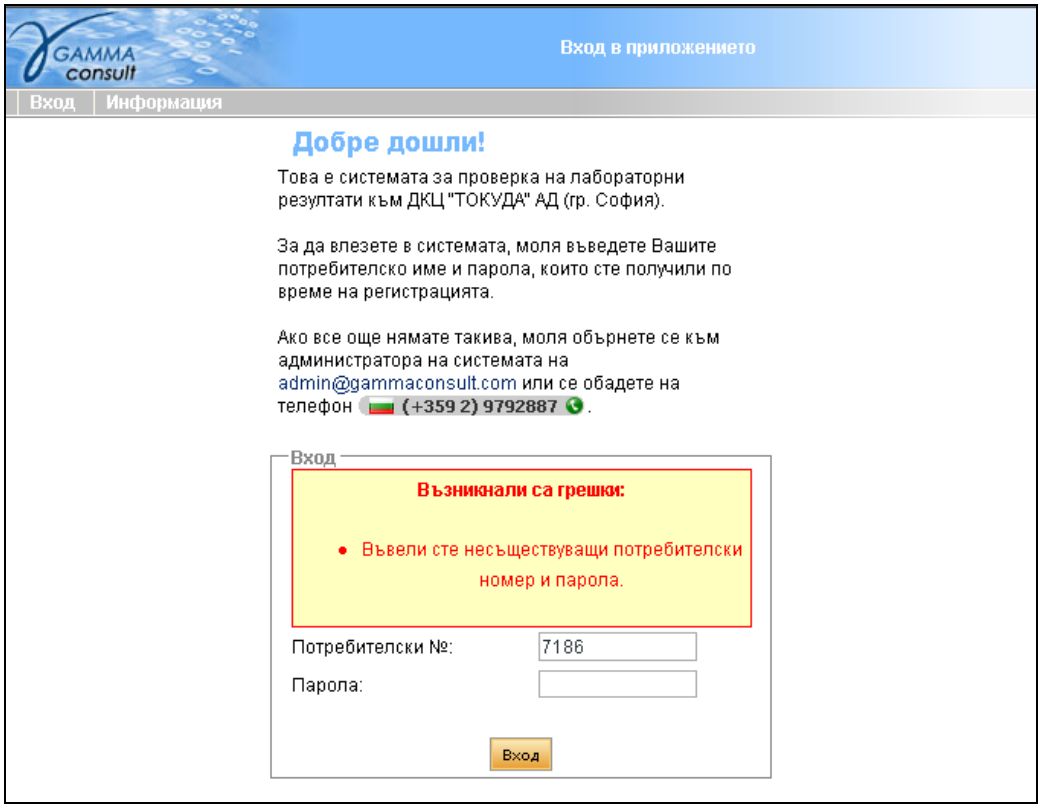
резултат на клиентската заявка, браузърът получава Forms Authentication Cookie с кодирано съобщение, съдържащо идентификационна информация

2. В случай на грешка показва съобщение (фиг. 6.2).

*Забележка*: Така въведената парола оторизира потребителя за всички изследвания, извършени в периода на валидност на паролата. Автентикацията е валидна в рамките на 20 минути от последното действие на клиента, което предизвиква заявка към сървъра. Продължителността зависи от продължителността на сесията и периода на валидност на криптираното съобщение в Authentication Cookie.




**фиг. 6.1** Екран за вход в системата




**фиг. 6.2** Екран за вход в системата. Показване на грешки

### 3.5.2. Оторизация за дата на изследване

Функционалността дава възможност на потребителя в рамките на текущата сесия да показва резултати от изследвания за дати различни от тези, за които е валидна паролата, с която той е автентикиран пред системата.

Сценарият описва реализацията на едно от ключовите функционални изисквания към осигуряване на сигурността на данните. Включването на всяка дата на изследване с отделна парола позволява допълнително ниво на сигурност за потребителите и намалява до минимум възможността трети лица да придобият достъп до поверителна информация за изследвания, тъй като за всяко от тях би могло да се изисква отделна парола.

Предпоставки

- Потребителят е автентикиран.
- Потребителят има изследвания за дати, за които все още не е оторизиран в рамките на текущата сесия.

Стъпки

1. Потребителят отправя заявка към сървъра за достъп до страницата с лични данни.
2. Сървърът прочита от базата информация за личните данни на пациента и за датите, за които има заявки за изследване и я визуализира. Датите на изследване се показват в таблична форма с икони, които подсказват дали потребителят е оторизиран за съответната дата (фиг. 7).
3. Потребителят избира дата на изследване и въвежда парола
    a. Глобална парола
        i. Потребителят въвежда вярна глобална парола (най-долният ред на таблицата с дати на изследване).
        ii. Системата оторизира потребителя за всички дати на изследване, за които паролата е валидна, в допълнение на датите, за които потребителят е бил вече оторизиран.
    b. Индивидуална парола
        i. Потребителят избира бутон "Добавяне" на реда с избрания запис и въвежда валидна парола.
        ii. Системата оторизира потребителя само за датата от избрания ред, в допълнение на датите, за които потребителят е бил вече оторизиран.

В случай на грешно въведена парола, системата показва съобщение за грешка.

### 3.5.3. Отмяна на оторизацията за дата на изследване

Функционалността позволява на потребителя да отмени получена оторизация за индивидуална дата на изследване (частичен изход).

Предпоставки

- Потребителят е автентикиран.
- Потребителят има изследвания за дати, за които е оторизиран в рамките на текущата сесия.

Стъпки

1. Потребителят отправя заявка към сървъра за достъп до страницата с лични данни.

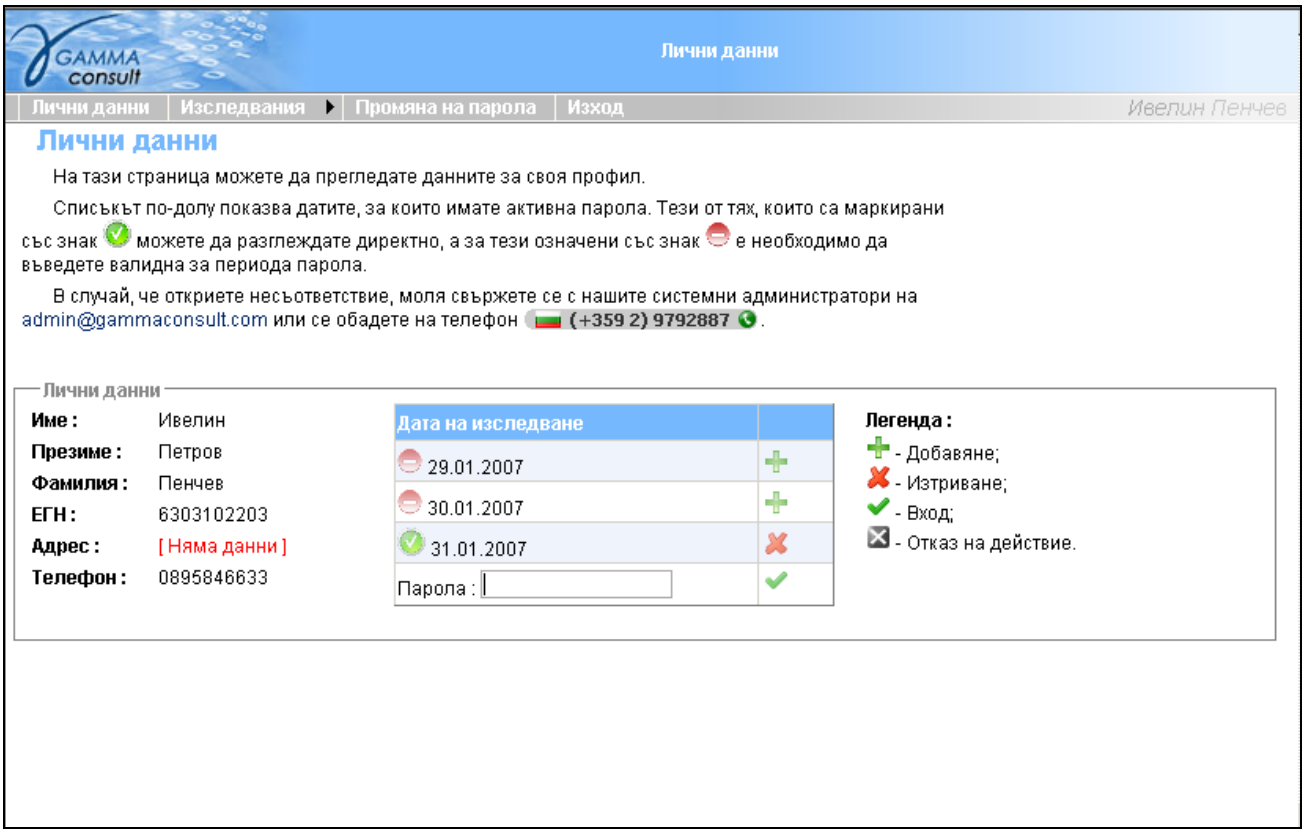
2. Сървърът генерира страницата в отговор на заявката на клиента, като показва текущия статус в таблицата с дати за изследвания (фиг. 7).

3. Потребителят избира ред от таблицата, за който е оторизиран и натиска бутон “Изтриване”.

4. Системата показва предупредително съобщение и изисква за потвърждаване на действието.

5. Потребителят потвърждава действието.

6. Системата премахва датата от списъка с дати, за които пациентът е оторизиран и променя статуса на реда от таблицата с дати за изследване.




**фиг. 7** Екран за лични данни и оторизиране за изследване

### 3.5.4. Промяна на парола

Функционалността позволява на потребителя да промени парола за изследване, като възможността е или да направи глобална или индивидуална промяна. При глобалната промяна потребителят не избира дата и всички пароли, които съвпадат с въведената за промяна, биват променени към новата. В случая на индивидуална парола, потребителят избира конкретен период на валидност на парола и само тази парола бива обновена.

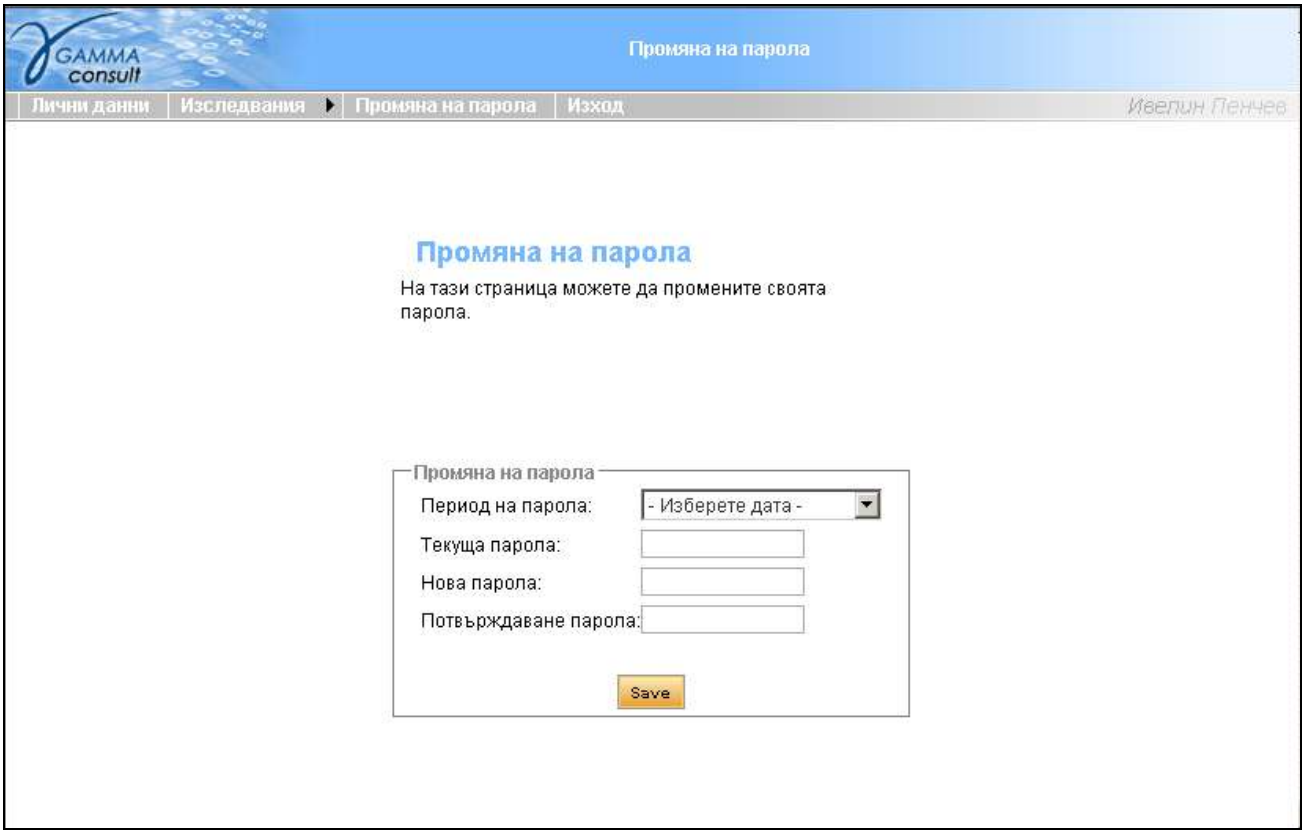
Предпоставки

- Потребителят е автентикиран.
- Потребителят вече е оторизиран в рамките на текущата сесия за дата, за което иска да промени парола (не е необходимо за тази дата да има изследване).

Стъпки

1. Потребителят отправя заявка към сървъра за достъп до страницата за промяна на парола.
2. Системата визуализира страницата, като зарежда в списъка с дати периодите на валидност на паролите, които потребителят е въвел в рамките на текущата сесия(фиг. 8).
3. Потребителят избира период на валидност и въвежда стара парола за периода, нова парола и потвърждение на новата парола.
4. Системата проверява съответствието на старата парола с периода на валидност, както и съответствието между новата парола и потвърждението ѝ и в случай на грешка показва съобщение. В случай на коректни данни, системата променя паролата и автоматично оторизира потребителя с новата парола.

*Забележка:* Системата извършва проверка и за дължината на въведената парола, като това може да се дефинира от конфигурационния файл на приложението (web.config).




**фиг. 8** Екран за промяна на потребителска парола

### 3.5.5. Показване на резултати от изследване на базата на критерии за търсене

Функционалността позволява на потребителя, по избрани критерии за търсене да филтрира изследванията сред тези, за чиято дата е автентикиран в рамките на текущата сесия. Формата има четири филтърни полета:

- дата на заявка: сървърът зарежда датите, за които има изследване и потребителят се е автентикирал за тях (в смисъла на сценарий 2);
- услуга: сървърът зарежда услугите, за които потребителят се е изследвал. Някои от услугите са самостоятелни, но други (примерно ПКК, автоматично, с 5 части диференциално броене) са съставени от показатели;
- дата от и дата до: дефинират периода, който представлява интерес за потребителя. Филтърът е полезен, когато периодите на валидност на парола са за по-голям период, от целевия.

Предпоставки

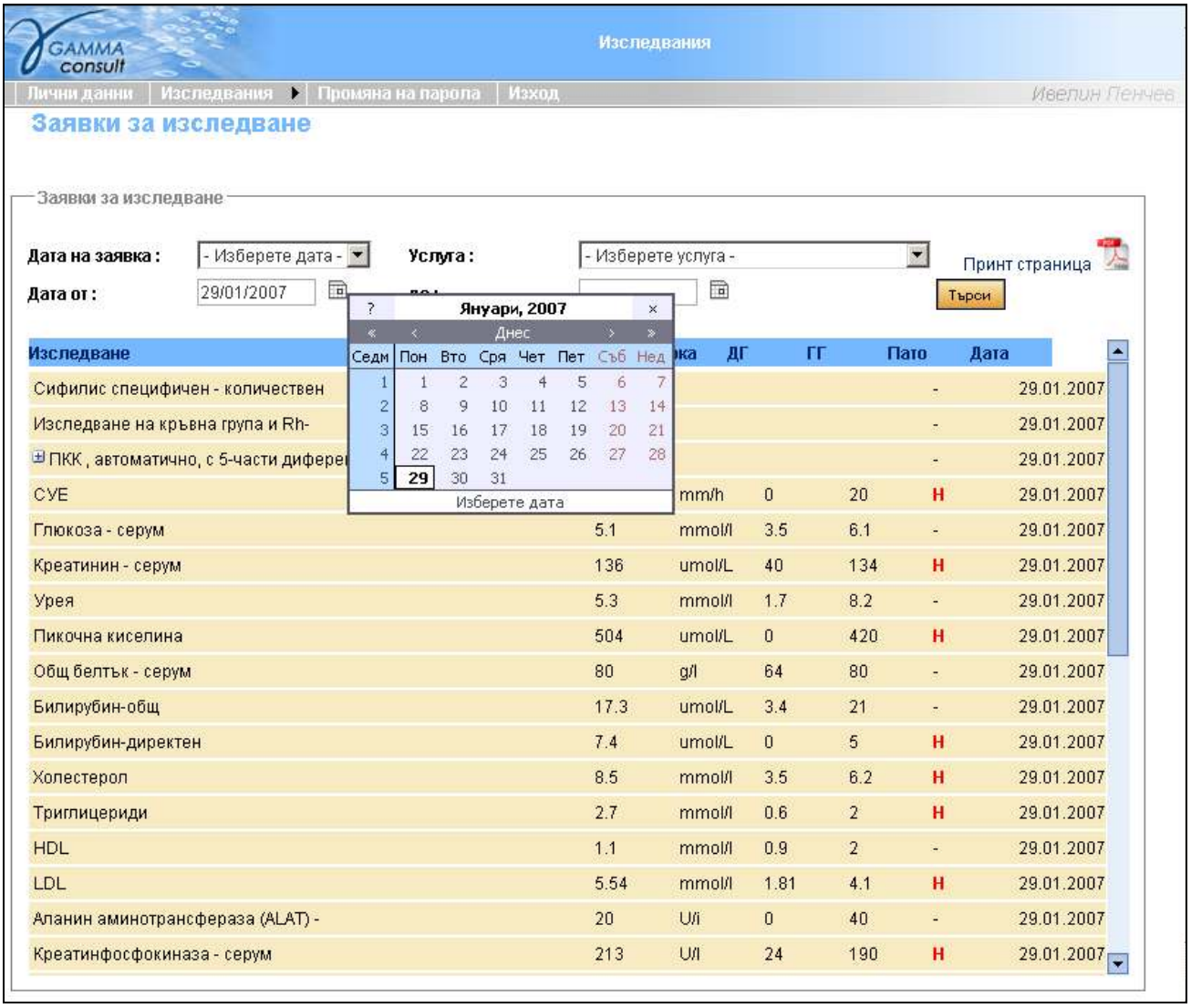
- Потребителят е автентикиран.
- Потребителят е оторизиран за поне една дата на изследване.

Стъпки

1. Потребителят отправя заявка към сървъра за достъп до страницата за показване на резултати от изследване.
2. Сървърът зарежда датите на изследване, за които потребителят е автентикиран и услугите, за които има данни за изследване и показва форматал.
3. Клиентът избира критерии за търсене (дата на заявка, услуга и период) и натиска бутона “Търси”.
4. Сървърът извлича резултатите от базата данни, извършва необходимата обработка, за да покаже йерархично услуги и показатели, където е необходимо и показва таблично резултата във формата (фиг. 9). Таблицата има следните колони:

- Изследване: име на услугата или показателя;
- Резултат: стойност на услугата или показателя;
- Мярка: мерна единица на резултата;
- Долна граница: приета стойност за долна граница на нормалните резултати;
- Горна граница: приета стойност за горна граница на нормалните резултати;
- Патологично състояние: със символи “H”,”L”и “-” се означават съответно завишено, занижено или нормално ниво на показател. Занижените и завишените стойности удобно се показват в червен цвят, с цел да бъдат по-лесно забележими;
- Дата: дата на изследването.

5. Клиентът разглежда резултатите и при необходимост избира подробен изглед на съставна услуга.
6. Сървърът показва показателите в състава на услугата като второ ниво на йерархия в таблицата под услугата.
7. Клиентът избира сортиране по колона в таблицата с резултати.
8. Сървърът извършва обработката на данните и показва резултата.




**фиг. 9** Екран за резултати от заявки за изследване

### 3.5.6. Разпечатване на резултатите от изследване

Функционалността предоставя възможност на потребителя да получи изглед на резултатите от изследване, готов за разпечатване на принтер. За технологичната реализация е избран форматът PDF, тъй като той предоставя възможност документа да се форматира подходящо, без да са известни настройките на клиента за големина на страницата за печат и разстоянията на съдържанието от ръба на листа.

Предпоставки

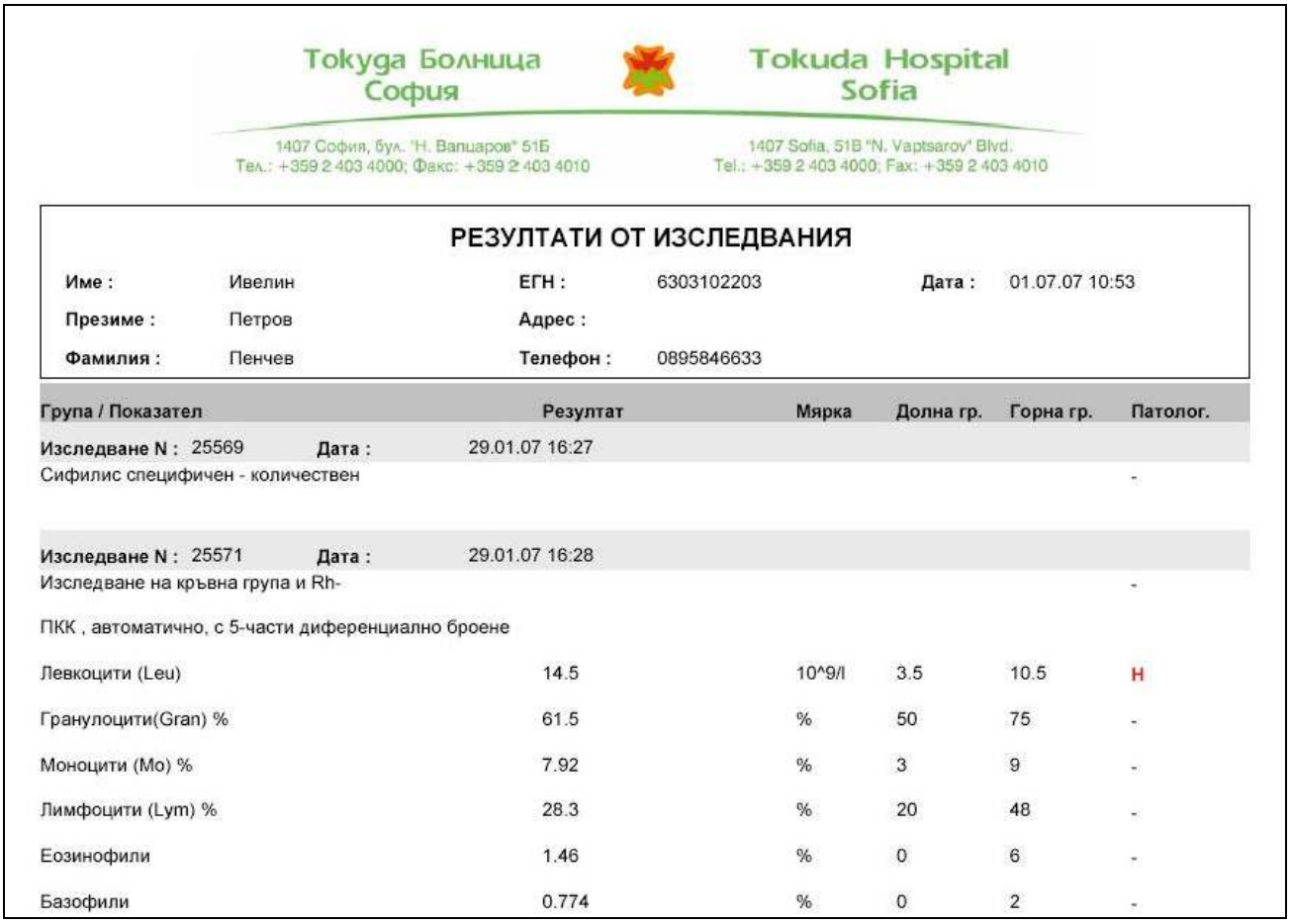
- Потребителят е автентикиран.
- Потребителят е оторизиран за поне една дата на изследване.

Нефункционални изисквания

- Потребителят разполага с инсталиран продукт, който отваря в режим за четене файлове в PDF формат.
- Потребителят разполага с инсталиран принтер за формат А4.

Стъпки

1. Потребителят отправя заявка към сървъра за достъп до страницата за показване на резултати от изследване.
2. Сървърът зарежда датите на изследване, за които потребителят е автентикиран и услугите, за които има данни за изследване за потребителя и показва формата.
3. Клиентът избира критерии за търсене (дата на заявка, услуга и период) и натиска бутона “Принт страница”.
4. Сървърът извлича резултатите от базата данни, извършва необходимата обработка, за да покаже йерархично услуги и показатели, където е необходимо, генерира таблично резултата в PDF формат и изпраща файла на клиента като файл за сваляне.
5. Клиентът получава покана да свали файла, приема поканата и след сваляне на файла го отваря с програма за четене на файлове в PDF формат (фиг. 10) и го разпечатва на принтера.



Tokyga Болница София

1407 София, бул. "Н. Вапцаров" 51Б
Тел.: +359 2 403 4000; Факс: +359 2 403 4010

Tokuda Hospital Sofia

1407 Sofia, 51B "N. Vaptsarov" Blvd.
Tel.: +359 2 403 4000; Fax: +359 2 403 4010

РЕЗУЛТАТИ ОТ ИЗСЛЕДВАНИЯ

| | | | | | |
|---|---|---|---|---|---|
| Име : | Ивелин | ЕГН : | 6303102203 | Дата : | 01.07.07 10:53 |
| Презиме : | Петров | Адрес : | | | |
| Фамилия : | Пенчев | Телефон : | 0895846633 | | |

| Група / Показател | Резултат | Мярка | Долна гр. | Горна гр. | Патолог. |
|---|---|---|---|---|---|
| Изследване N : 25569 Дата : | 29.01.07 16:27 | | | | |
| Сифилис специфичен - количествен | | | | | - |
| Изследване N : 25571 Дата : | 29.01.07 16:28 | | | | |
| Изследване на кръвна група и Rh- | | | | | - |
| ПКК , автоматично, с 5-части диференциално броене | | | | | |
| Левкоцити (Leu) | 14.5 | 10^9/l | 3.5 | 10.5 | H |
| Гранулоцити(Gran) % | 61.5 | % | 50 | 75 | - |
| Моноцити (Mo) % | 7.92 | % | 3 | 9 | - |
| Лимфоцити (Lym) % | 28.3 | % | 20 | 48 | - |
| Еозинофили | 1.46 | % | 0 | 6 | - |
| Базофили | 0.774 | % | 0 | 2 | - |

**фиг. 10** Страница за печат в PDF формат.

### 3.5.7. Показване на измененията на показател

Функционалността дава възможност на потребителя да проследи визуално изменението на показател в течение на времето. Формата има четири филтърни полета:

- дата на заявка: сървърът зарежда датите, за които има изследване и потребителят се е автентикирал за тях (в смисъла на сценарий 2);
- услуга: сървърът зарежда услугите, за които потребителят се е изследвал. Някои от услугите са самостоятелни, но други (примерно ПКК, автоматично, с 5 части диференциално броене) са съставени от показатели;
- дата от и дата до: дефинират периода, който представлява интерес за потребителя. Филтърът е полезен, когато периодите на валидност на парола са за по-голям период, от целевия.

Предпоставки

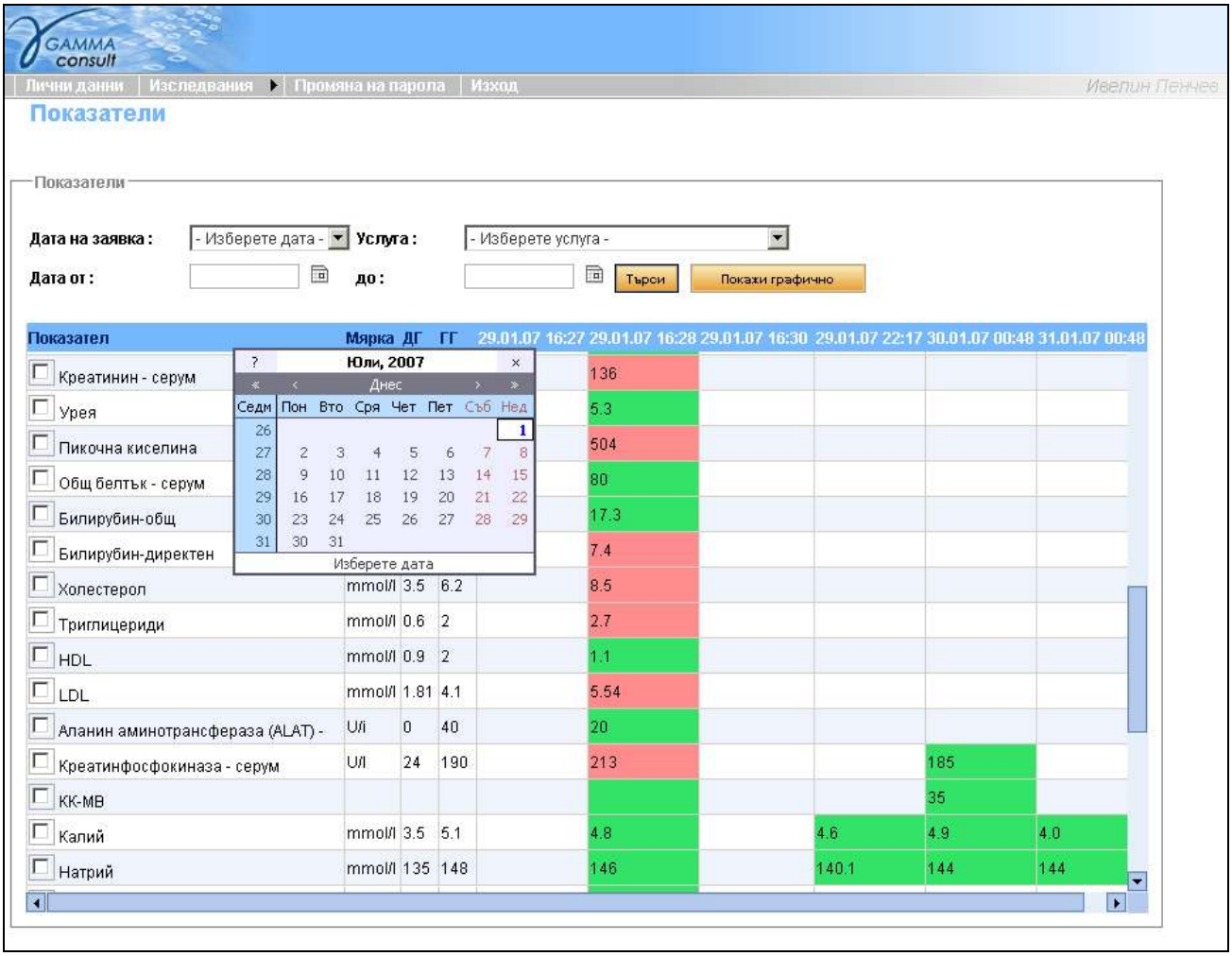
- Потребителят е автентикиран.
- Потребителят е оторизиран за поне една дата на изследване

Стъпки

1. Потребителят отправя заявка към сървъра за достъп до страницата за показване на стойностите на показатели.
2. Сървърът зарежда датите на изследване, за които потребителят е автентикиран и услугите, за които има данни за изследване за потребителя и показва формата за търсене.
3. Потребителят натиска бутон “Търси”.
4. Системата извлича резултатите от базата данни, извършва необходимата обработка и показва таблично показателите по редове, а по колони - датите на изследване, (за една дата е възможно да има повече от едно изследване) (фиг. 11). Таблицата има следните колони:

- Показател: име на показателя;
- Мярка: мерна единица на показателя;
- Долна граница: приета стойност за долна граница на нормалните резултати;
- Горна граница: приета стойност за горна граница на нормалните резултати;
- Дата и час на изследване на показателя: В тези колони има стойности, само когато показателят е изследван на съответната дата и час.

5. Потребителят разглежда табличното представяне на резултатите. За всеки показател клетките със стойности са оцветени в цветове: зелен, червен и син, съответно за нормални, завишени или занижени стойности на показател.
6. Потребителят избира сортиране по колона в таблицата с резултати.
7. Сървърът извършва обработката на данните и показва резултата.




**фиг. 11** Екран за показване стойности на показатели

### 3.5.8. Графично показване на изменения на показател и отношението между показатели

Функционалността дава възможност на потребителя да проследи графично изменението на няколко показателя в течение на времето, както и тяхното съотношение. Необходимо е показателите да бъдат измервани в една мерна единица.

Предпоставки

- Потребителят е изпълнил сценарий 7.

Нефункционални изисквания

- Потребителят е разрешил pop-up прозорци за сайта

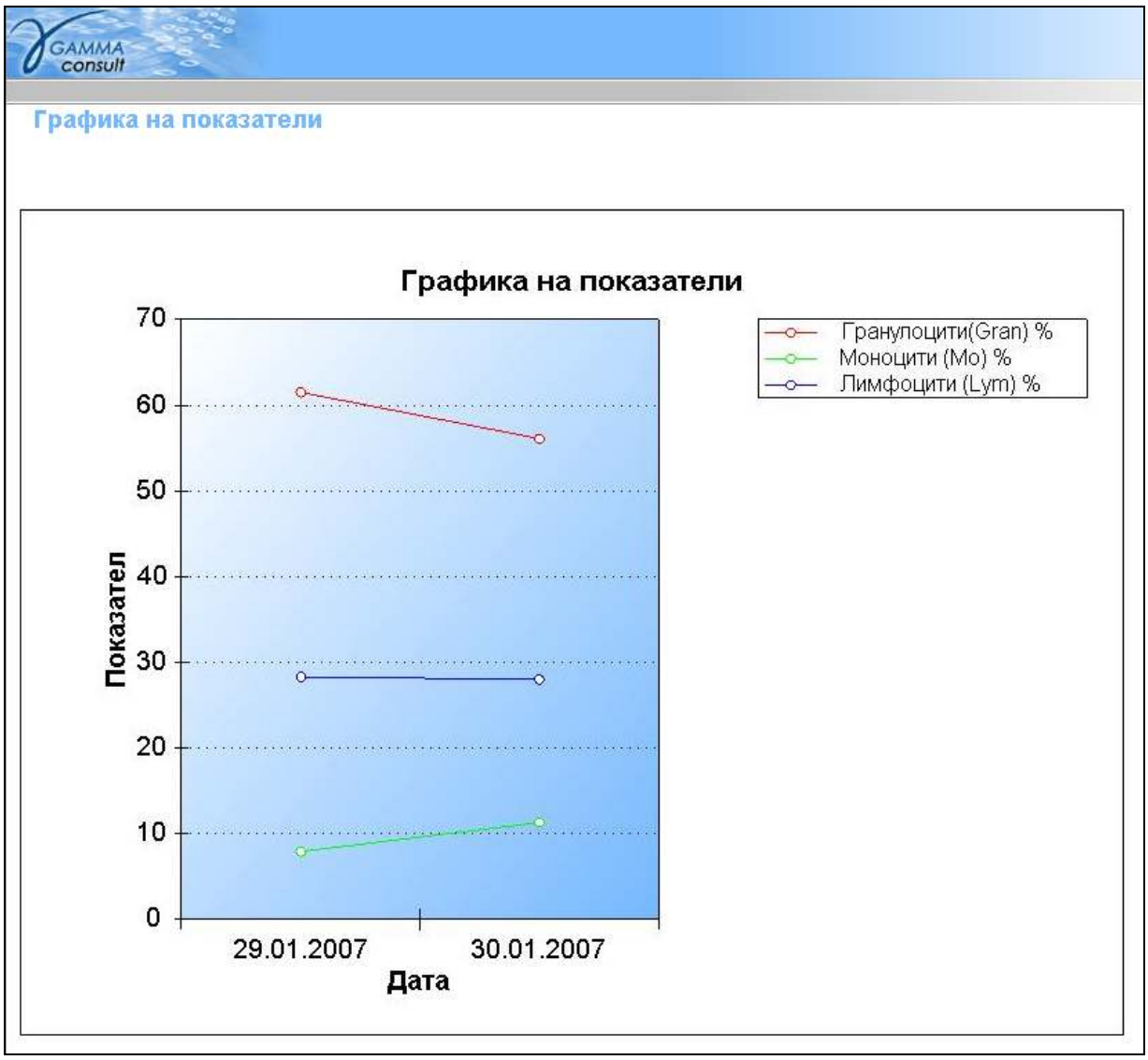
- Администраторите на сайта са дефинирали цветове за показателите (към момента цветовете са три – червен, зелен, син).

Стъпки

1. Потребителят избира показатели, като маркира кутийките(checkbox) на реда на показателя и натиска бутон “Покажи Графично”.

2. Сървърът валидира мерните единици на избраните показатели и дали техният брой не надвишава броя на цветовете дефинирани от администратора на сайта. Сървърът извлича стойностите за показателите избрани от потребителя и генерира двуизмерна графика, която изпраща на клиента.

3. Потребителят получава в pop-up прозорец графичен изглед на стойностите на избраните показатели (фиг. 12).




**фиг. 12** Екран за графика на показатели

### 3.5.9. Изход от системата

Функционалността позволява потребителят да приключи текущата сесия по сигурен начин, като декларира пред системата желание за това.

Предпоставки

- Потребителят е автентикиран пред системата.

Стъпки

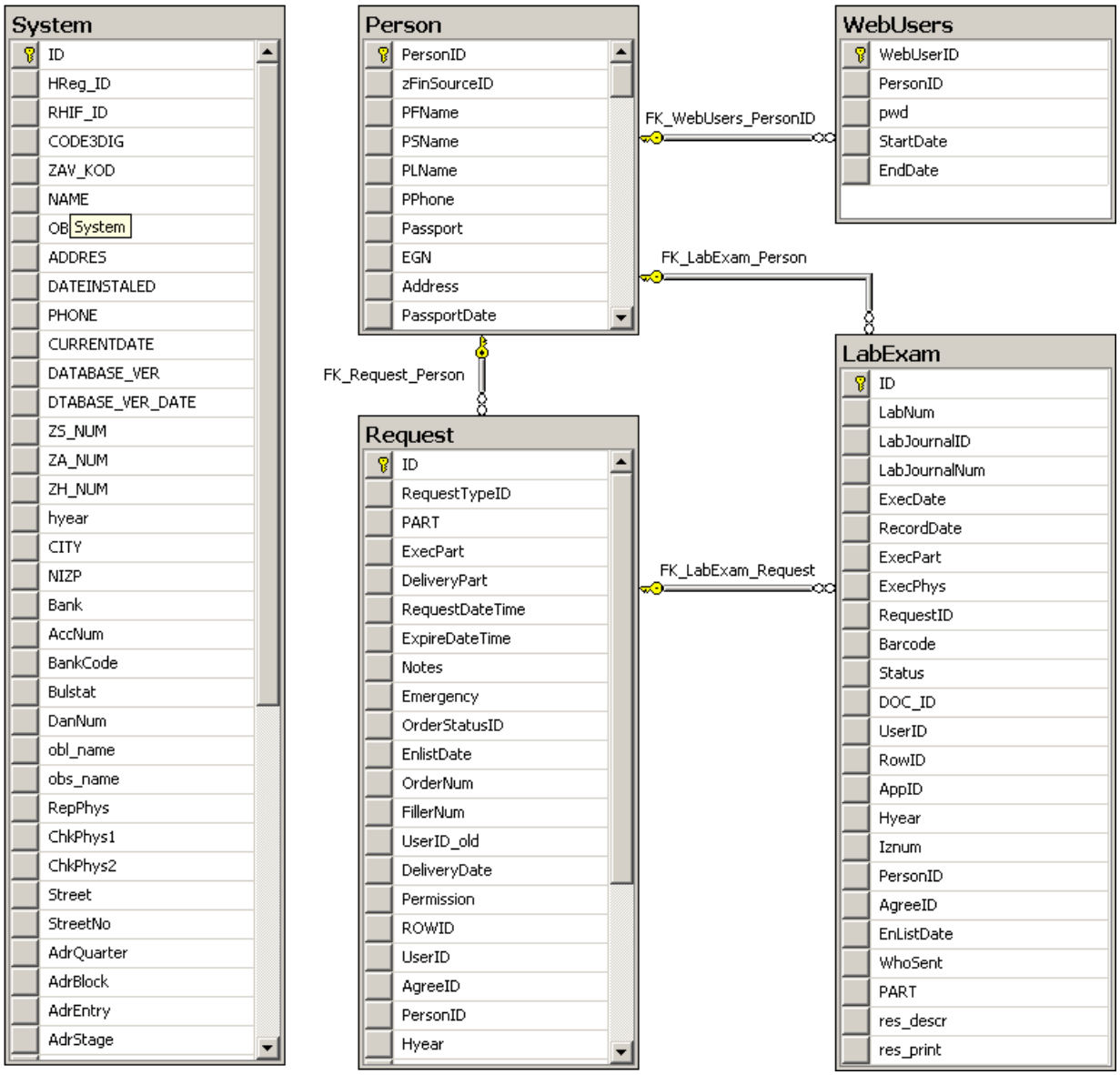
1. Потребителят избира опция “Изход” от главното меню.
2. Сървърът изчиства информацията за текущата сесия и изпраща празен билет (ticket) в Forms Authentication Cookie, което се връща в отговор на заявката на клиента. По този начин, при следващи заявки, потребителят няма да може да изпрати кодирани данни, които да го идентифицират пред системата и в бъдеще няма да бъде автентикиран.
3. Клиентът получава отговора на сървъра и бива пренасочен към страницата за вход в системата.

## *3.6. Релационен модел на базата данни*

Базата данни е неизменна и важна част от всяка система, която съхранява данни между потребителските сесии. Системата за управление на бази данни, както и моделът на базата към приложението към настоящата дипломна работа са утвърдени от фирма “ГамаКонсулт” с цел да се покрият изискванията, които фирмата има към своите софтуерни продукти. Представени са само онези таблици, връзките между тях, колоните, съхранени процедури и функции, които имат пряко отношение към темата на дипломното задание.

### 3.6.1. Таблици от общ характер и таблици с данни за пациенти

Таблиците и връзките между тях са визуализирани на диаграмата по долу (фиг. 13), след което следват техните описания. Ще наричаме дадена колона от таблица е значима, ако се използва в приложението.




**фиг. 13** Диаграма на таблиците с общо предназначение в базата данни.

1. **Таблица "System"** - съдържа информация за болничното заведение или лаборатория, в което оперира системата MultiLab. Системата обработва винаги един ред от тази таблица. По-важните от колоните са:

   a. NAME: Име на медицинското заведение;

   b. CITY: Град;

   c. ADDRES: Адрес;

   d. ZIP: Пощенски код;

   e. PHONE: Телефон;

   f. Bulstat: Булстат.

2. **Таблица "Person"** – съдържа личните данни на пациент. Всеки ред от таблицата съответства на точно един физически пациент, което се подсигурява с уникален индекс по колоната за ЕГН. Идентификационният номер на пациент е частен ключ на таблицата и се използва за потребителско име при вход в системата. По-специално за МБАЛ "Токуда" идентификационният номер е записан на пациентска карта, която се използва за идентификация и бързо търсене на данни от изследвания на пациент. По-важните от колоните са:

a. PersonID: Персонален номер. Колоната уникално идентифицира запис от таблицата и е основният външен ключ, чрез който се прави връзка между пациент и данните за негови изследвания;
b. PFName: Име;
c. PSName: Презиме;
d. PLName: Фамилия;
e. EGN: ЕГН;
f. Address: Адрес;
g. PPhone: Телефон.

3. **Таблица "WebUsers**" – съдържа паролите и периодите на валидност за потребителите на уеб-интерфейса. По-важните от колоните са:

a. WebUserID: частен ключ, използва се за уникално идентифициране на запис в случая на промяна на потребителска парола;
b. PersonID: Идентификационен номер на потребител. Външен ключ към таблицата с потребители. Идентифицира потребителя, за когото текущата парола се отнася;
c. Pwd: Парола на потребителя. Паролата се съхранява в базата данни хеширана със стандартните средства на Microsoft SQL Server 2005. Алгоритъмът който се използва е SHA-1 (Secure Hash Algorithm 1) (17). За всяко съобщение с големина по-малка от $2^{64}$ бита, алгоритъмът калкулира еднопосочен хеш който е 160 бита (20 символа, ако се представи в ASCII кодиране или 40 символа, ако се представи в шестнайсетична бройна система). Алгоритъмът се счита за сигурен при използване върху пароли. Първо чрез придобиване на достъп до хеша е невъзможно (освен чрез атака с пълно изчерпване) да се възстанови оригиналното съобщение. Второ алгоритъмът се счита за свободен от колизии

за по-малко от $2^{69}$ операции – т.е. не може да бъде намерено друго съобщение с по-малко от $2^{69}$ операции, което при хеширане да дава същия резултат;

d. StartDate: Началото на периода, за който текущата парола е валидна;
e. EndDate: Край на периода, за който паролата е валидна. Лесно би могло да се генерира относително "вечна" парола за приложението ако датите за начало и край на период се поставят достатъчно отдалечени една от друга във времето.

4. **Таблица "Request"** – съдържа данните за заявка за изследване – пациент, поръчител, изпълнител, дата на валидност и др. В лабораторията резултатите по заявки се описват в таблиците "LabExam", "LabTest" и "LabTestResult", които ще разгледаме в последствие. По-важните от колоните са:

a. RequestDateTime: Дата и час на заявка. Използва се за филтриране на заявките, чиито резултати пациентите могат да разглеждат;
b. PersonID: Идентификационен номер на пациент. Използва се за външен ключ към таблицата с пациенти.

5. **Таблица "LabExam" -** таблицата е подобна на посочената по-горе таблица "Request". Съдържа обобщена информация за заявката в процеса на провеждането на самото изследване. В таблицата най-общо се описват кой извършва изследването и кога, пациента за който се отнася, кога е приета за изпълнение и статуса. Един запис в тази таблица съответства на един запис в таблицата със заявки.

a. ExecDate: Дата и час на изпълняване;
b. RequestID: Номер на заявка. Външен ключ към таблицата със заявките, идентифициращ заявката към която се отнася текущото изследване;
c. PersonID: Идентификационен номер на потребител. Външен ключ към таблицата с потребители. Идентифицира еднозначно потребителя, за който се отнася даденото изследване;
d. Status: Статус на заявката за изследване.

### 3.6.2. Таблици за резултати от изследване

Таблиците и връзките между тях са визуализирани на диаграмата по долу (фиг. 14), след което следват техните описания. Ще наричаме дадена колона от таблица е значима, ако се използва в приложението.


**фиг. 14** Таблици с резултати за изследване

1. **Таблица “LabTest” -** съдържа списък от поръчаните изследвания в рамките на една заявка. По-важните от колоните са:

a. LabExamID: Уникален идентификатор на запис от таблица “LabExam”. Външен ключ еднозначно идентифициращ запис от таблицата “LabExam”;
b. ServiceID: Идентификационен номер на услуга. Външен ключ еднозначно идентифициращ запис от таблицата “Service”;
c. ExecDate: Дата и час на изпълняване;
d. Status: Статус на заявката за изследване.

2. **Таблица “Result”** – съдържа стойностите на показателите на изследване. По-важните от колоните са:

a. ID: Уникален идентификатор на запис.
b. LabTestID: Идентификатор на запис от таблица “LabTest”. Външен ключ еднозначно идентифициращ запис от таблицата “ LabTest ”.
c. ResOrder: Ред на сортиране. Колоната се използва за въвеждане на допълнителен критерий при сортиране на показателите при изпълнение на заявката за стойност на показатели при изследване.
d. LowLimit: Долна граница на нормалните за показателя стойности.
e. HighLimit Горна граница на нормалните за показателя стойности.
f. PatoStateID: Идентификатор на патологично състояние. При стойности извън нормалните се счита, че при пациента има проявена патология. Със стойности е означено следното: 0 – нормално състояние, 1 – измерените стойности за показателя са по-ниски от нормалните или 2 - измерените стойности за показателя са по-високи от нормалните.

3. **Таблица “Service” -** съдържа набор от всички услуги, които лабораторията предлага. По-важните от колоните са:

a. ID: Уникален идентификатор на запис;
b. Code: Код на услугата;
c. Name: Наименование на услугата;
d. SOrder: Ред на сортиране. Колоната се използва за въвеждане на допълнителен критерий при сортиране на показателите при изпълнение на заявката за стойност на показатели при изследване.

4. **Таблица “Activity” -** съдържа набора от показатели, които лабораторията изследва . По-важните от колоните са:

   a. ID: Идентификатор на запис;
   b. Code: Уникален код на показател;
   c. Name: Името на показателя, под което той се визуализира в приложението;
   d. NameOrig: - оригинално название. Колоната съдържа медицинското обозначение на показател.

5. **Таблица “ServiceActivity”** - служи за дефиниране на релация тип “много – към - много” между таблиците “Service” и “Activity”. По-важните от колоните са:

   a. ID: Уникален идентификатор на запис;
   b. ActivityID: Идентификатор на показател;
   c. ServiceID: Идентификатор на услуга;
   d. SOrder: Ред на сортиране.

### 3.6.3. Съхранени процедури(Stored Procedures) и потребителски функции (User Defined Functions)

Базата данни на уеб-интерфейса е еднаква с тази, която настолното приложение “Мултилаб” използва. Поради изключителната важност на данните и за да се осигури до максимална степен сигурната работа на системата, достъпът до базата данни на приложението се реализира само и единствено чрез съхранени процедури и потребителски функции. От страна на уеб-системата, връзката към базата данни се осъществява с потребител с ограничени права за достъп само до процедурите и функциите, които са необходими за работата й. По този начин сигурно се елиминира възможността в резултат на зложелателни потребители или грешка в приложението да бъдат загубени данни чрез интервенция върху базата данни през функции на уеб-интерфейса. Процедурите и функциите са следните:

1. **Процедура WebGetHospitalDetails**
   a. Аргументи – няма
   b. Описание – процедурата връща данни от последния запис на таблица “System”, където е записана информация за лабораторията или лечебното

заведение, при което е инсталирана лабораторната информационна система “Мултилаб”.

2. **Процедура WebGetPersonalDetails**
    a. Аргументи
        i. @PersonID – уникален идентификатор на пациент;
        ii. @pwd – валидна и активна парола на потребителя.
    b. Описание – процедурата връща личните данни на пациента с идентификатор съответстващ на подадения @PersonID. Необходимо е да има валидна и активна парола за периода на заявката.

3. **Процедура WebGetRequestDates**
    a. Аргументи
        i. @PersonID – уникален идентификатор на пациент;
    b. Описание – процедурата връща датите за които пациентът има входящи заявки. Необходимо е да има валидна и активна парола за периода на заявката.

4. **Процедура WebGetResults**
    a. Аргументи
        i. @PersonID – уникален идентификатор на пациент;
        ii. @pwd – валидна и активна парола на потребителя за периода на заявката;
        iii. @RequestDate – Дата на заявка;
        iv. @ServiceID – Идентификатор на услуга;
        v. @StartDate;
        vi. @EndDate.
    b. Описание – процедурата връща като резултат таблица с резултати, които отговарят на критериите за търсене.

5. **Процедура WebGetServices**
    a. Аргументи:
        i. @PersonID – уникален идентификатор на пациент.

b. Описание – Процедурата връща списък със стойности за услугите извършвани в лабораторията.

6. **Процедура WebUsers_Ins**
   a. Аргументи:
      i. @PersonID – уникален идентификатор на пациент;
      ii. @pwd – валидна парола на потребителя;
      iii. @StartDate (начало на периода, в който се търси изследване);
      iv. @EndDate (край на периода, в който се търси изследване).
   b. Описание – Процедурата добавя запис в таблицата “WebUsers”, който се използва в последствие

7. **Процедура WebUsers_UpdPWD**
   a. Аргументи
      i. @webUserID – идентификационен номер на запис от таблицата “WebUsers”;
      ii. @PersonID – Идентификатор на личността на пациента;
      iii. @old_pwd – Стара парола;
      iv. @new_pwd – Нова парола.
   b. Описание – Процедура се използва за смяна на една или повече пароли

8. **Потребителска функция WebUsers_CheckUpd**
   a. Аргументи
      i. @PersonID - Идентификатор на личността на пациента;
      ii. @new_pwd – Нова парола.
   b. Описание – функцията се използва за проверка на периодите, за които дадена парола е валидна.

## *3.7. Програмна част на реализацията*

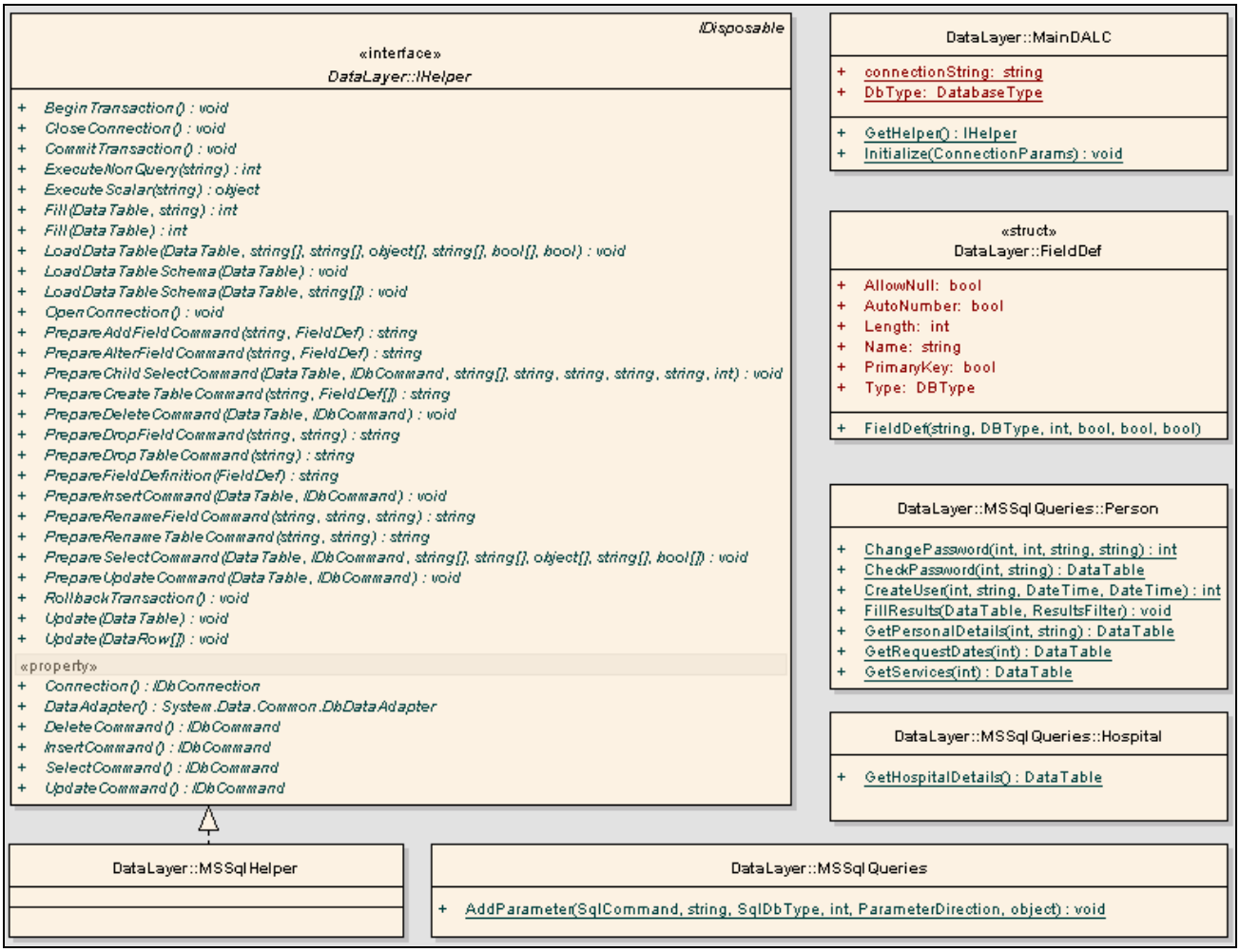
Ще опишем основните класове за всеки модул, тяхното предназначение и по-важните моменти от тяхната реализация. Реализацията е базирана на базови компоненти, които изграждат рамка(framework) на трислойното уеб-приложение. Разработката и обособяването

на подобни типови компоненти, позволяват преизползване на код и повишена скорост на разработка – фактори от изключително значение за ефективността на процеса на разработка.

Всеки от модулите в системата съдържа части, общи за голям кръг аналогични приложения, както и строго специфична функционалност, реализирана конкретно за нуждите на тази система.

### 3.7.1. Модул за връзка за базата данни (DataLayer)

Основните класове на модула за показани на фиг. 15.




**фиг. 15** Класова диаграма на модул “DataLayer”

- **IHelper:** Интерфейс, който по идеологията на модула трябва да бъде имплементиран от всички специфични имплементации, които предоставят достъп до

различни СУБД (SQL Server, MySQL, Oracle, Sybase, MS Access). В рамките на текущата система това е клас “MSSqlHelper”.

- **MSSqlHelper:** Имплементация на интерфейса “IHelper” за Microsoft SQL Server. По-важните методи и членове на класа са:
  - public MSSqlHelper(string connectionString) Създава нова инстанция на класа, като приема като параметър низ за връзка с базата данни (connection string);
  - public void PrepareSelectCommand(DataTable dataTable, IDbCommand sqlCommand, string[] selectColumns, string[] filterColumns, object[] filterValues, string[] sortColumns, bool[] sortAsc) - Подготвя команда за извличане (Select), чрез параметрите, които са подадени;
  - public void PrepareUpdateCommand(DataTable dataTable, IDbCommand sqlCommand) На основата на команда за извличане инициализира команда за обновление (Update);
  - public void PrepareInsertCommand(DataTable dataTable, IDbCommand sqlCommand) На основата на команда за извличане инициализира команда за добавяне (Insert);
  - public void PrepareDeleteCommand(DataTable dataTable, IDbCommand sqlCommand) На основата на команда за извличане инициализира команда за изтриване (Delete);
  - public int Fill(DataTable dataTable, string sql) Изпълнява SQL заявка и записва резултата в таблицата, получена като аргумент;
  - public int ExecuteNonQuery(string sql) Изпълнява SQL заявка и връща като резултат броя на повличниете записи;
  - public object ExecuteScalar(string sql) Изпълнява SQL заявка и връща като резултат стойността в първата колона на първия ред от резултата, ако има такъв;
  - public void Update(DataTable dataTable) Извършва обновление на записите от таблицата подадена като аргумент, в зависимост от тяхното състояние. Допустимите операции са обновление (Update), добавяне (Insert) или изтриване (Delete);
  - public void LoadDataTable(DataTable dataTable, string[] selectColumns, string[] filterColumns, object[] filterValues, string[] sortColumns, bool[] sortAsc, bool fillSchema) Зарежда таблица на базата на заявка към базата данни. При необходимост за колоните, които са избрани може да бъде заредена схемата на базата данни с информация за големини на полета, частни ключове, задължителни полета и др;

- public void LoadDataTableSchema(DataTable dataTable, string[] selectColumns) Зарежда схемата на базата данни за избраните колони;
- public void Dispose() – Имплементация на интерфейса "IDisposable". С включването в конструкция "using" на инстанциите на класа, автоматично се освобождават неменажираните ресурси (unmanaged resources), с което подсигуряваме освбождаването на връзките към базата данни;
- private void Init(string connectionString) – Инициализира член променливите (обектите за връзка и извличане на данни от базата);
- public void OpenConnection() – Отваря връзката към базата данни;
- public void CloseConnection() – Затваря връзката с базат данни;
- public void BeginTransaction() – Декларира начало на транзакция;
- public void RollbackTransaction() - Декларира край на транзакция;
- sqlDataAdapter – инстанция на клас "SqlDataAdapter";
- SelectCommand, InsertCommand, UpdateCommand, DeleteCommand– инстанции на клас SqlCommand, които се използват съответно за триене(Delete), добавяне(Insert), извличане(Select) или обновление(Update).

- **MainDALC:** Класът играе роля на фабрика за производство на точната имплементация на IHelper интерфейса, която да предостави средства за достъп и работа с базата данни. Класът се инициализира в началото на всяка заявка към сървъра от стойностите записани в конфигурационния файл (web.config). Всяка заявка се обработва в една нишка (thread), различна от нишката обработила предходна заявка, поради това член променливите на класа са статични и маркирани с "ThreadStatic" атрибут, което означава, че са статични за класа, но в рамките на текущата нишка. Това ни позволява без да създаваме инстанция от класа и да поддържаме референция към нея, само с еднократна инициализация да използваме функционалността, реализирана със статичните методи.
- **Person –** класът реализира изпълнението на съхранените процедури в базата данни, които са свързани с пациент. Методите са статични, тъй като не е необходимо да се съхранява някакво състояние за класа и всички необходими параметри за работата на методите се предават като аргументи. Методите включват:
  - public static DataTable GetPersonalDetails(int personID, string password) – Извлича личните данни за пациент;

- public static DataTable GetRequestDates(int personID) – Извлича датите, за които пациента има заявки за изследване;
- public static void FillResults(DataTable dtResults, ResultsFilter resultsFilter) - Извлича информацията за резултати от изследване за пациента, в зависимосто т критерии за търсене;
- public static DataTable GetServices(int personID) - Извлича услугите, за които потребителят има заявки;
- public static int CreateUser(int personID, string password, DateTime startDate, DateTime endDate) - Създава парола за вход в системата за зададен потребител и период на валидност;
- public static int ChangePassword(int webUserID, int personID, string oldPassword, string newPassword) - Извършва промяна на парола;
- public static DataTable CheckPassword(int personID, string password) - Извършва проверка за валидност на парола. Зарежда всички периоди на валидност, за които паролата е валидна.

- **Hospital** – класът реализира изпълнението на съхранените процедури в базата данни, които са свързани с болничното заведение или лаборатория. Методите са статични, тъй като не е необходимо да се съхранява някакво състояние за класа и всички необходими параметри за работата на методите се предават като аргументи. Методите включват:
  - public static DataTable GetHospitalDetails() – извлича данните за болничното заведение.

### 3.7.2. Общ модул (Common)

Модулът съдържа функционалност обща за останалите модули на системата. Основните класове на модула за показани на фиг. 16. Ще се спрем на по-важните класове и техните методи.

- **ApplicationLog** – реализира функционалността по форматиране и записване на изключения или други трасиращи съобщения, които помагат следенето на поведението на системата, откриването и коригирането на проблеми или други нетипични състояния и поведения.

- public static String FormatException(Exception ex, String catchInfo) Форматира изключение, като формира съобщение, което да е максимално информативно. Добавя информация за:

  - страница, където е възникнало изключението
  - адрес (IP:Port), от който е получена заявката
  - Cookie на заявката, което носи информация за идентификатора на сесия (Session ID)
  - Вътрешно изключение, ако има такова (допълнителна информация за причината за възникване на уловеното изключение)
  - Стек на извикванията – показва през кои методи е преминало изпълнението и на кой ред е възникнало изключението.
- public static void LogException(Exception exc, string info) Методът извършва запис на съобщение формирано от изключението чрез извикване на метода "FormatException". Извиква се в "catch" блок при улавяне на изключение.

- **Multilanguage** – Приложението разполага с механизъм, който в реално време извършва превеждане на интерфейса от конфигурационен файл в XML формат. Класът се използва за прочитане на XML файла и при търсене на стойности в речника.
- **Validation** – класът имплементира методи за валидация.
- **Encryption** – класът се използва за симетрично криптиране на стрингове. Алгоритъмът, който се използва е Рейндол (Rijndael) или известен още като "Advanced Encryption Standard (AES)". Основно приложение функционалността намира при криптиране на пароли и криптиране на низа на заявката (Query String). Криптирането на низа на заявката дава сигурност, че потребителят няма да може да манипулира низа и да го измени, като по този начин добие достъп до забранена функционалност – примерно редакция на запис от базата данни, който трябва да бъде отворен само за четене. Дължината на ключа в текущата реализация е 128 бита, което е признато от националната агенция за сигурност в САЩ за достатъчно за криптиране на информация с гриф "секретно"(18).
- **HashProvider –** класът се използва за хеширане на символни низове с използване на класа "MD5CryptoServiceProvider". Резултатът от изпълнението е шестнайсетичен символен низ с дължина 32 символа.

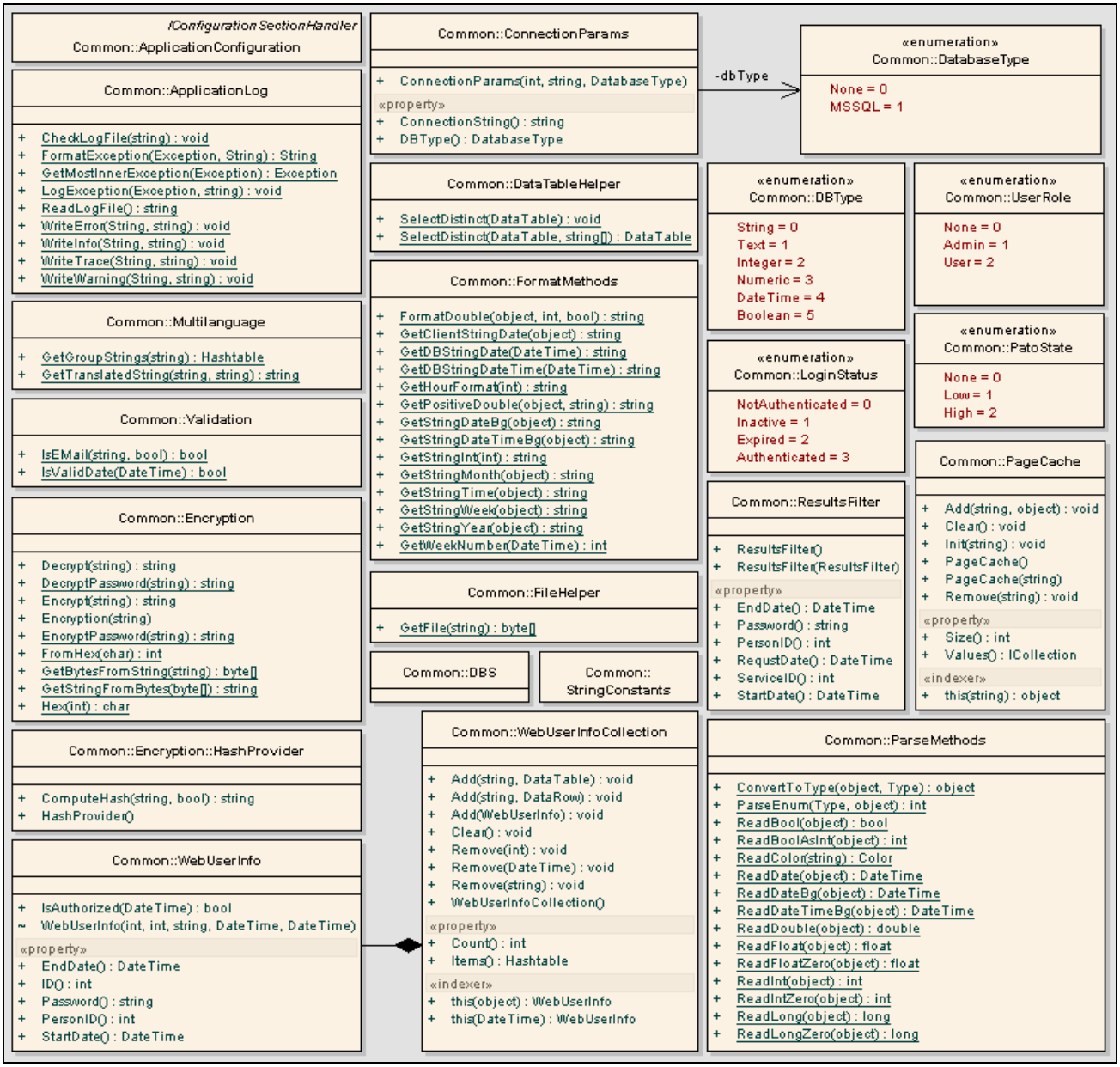
- **WebUserInfo** – класът се създава директно от записи от таблицата WebUsers и се използва за извличане на инфромация свързана с оторизацията на потребителя за дата на изследване.
- **WebUserInfoCollection** – Класът съдържа колекция от WebUserInfo обекти и методи за работа с тях. Използва се от обекта, който съхранява информацията за потребителя на текущата сесия, за да се подпомогне процеса на оторизация.
- **ConnectionParams** – Класът се използва за преносител на параметрите за връзка с базата данни. Използва се от клас MainDALC в модул DataLayer.
- **DataTableHelper** – Реализира функционалност, която предимно се изпълнява на сървъра, на който се намира СУБД и който не е стандартно реализиран в метода DataTable.Select(). Използваме, когато се налага обработка в паметта на данните прочетени от базата подобна на "SELECT DISTINCT".
- **FileHelper** – Съдържа методи за работа с файлове.

  - public static byte[] GetFile(string filePath) Прочита файл от диска и връща съдържанието като масив от байтове. Използва се за прочитане на логото на болничното заведение за целите на PDF принт страницата.

- **DBS** (съкратено от DB Scheme) – Класът съдържа изключително и само низови константи, описващи имената на таблиците от базата данни и техните колони, които представляват интерес за системата. Навсякъде в приложението колоните и таблиците се референцират по име чрез тези константи.
- **StringConstants** - Класът съдържа изключително и само низови константи, които представляват съобщения за грешки, имена на параметри, стоийности от XML ресурсния файла за многоезичността или имена на конфигурационни променливи от конфигурационния файл.
- **FormatMethods** – Класът съдържа методи за форматиране на обекти и прости стойности до символни низове(примерно формат на датата). Методите на този клас реализират унифицирано форматиране на базата на избраните езикови настройки за сайта.
- **ParseMethods** – Класът реализира механизми за конвертиране на стойности от обект или символен низ, когато сме сигурни в типа на данните, който очакваме. В случай на грешка при прочитане, резултатът е служебна стойност, който в последствие може да бъде сверена и да послужи за база за извеждане на съобщение за грешка. Механизмът е

изключително полезен за проверка на потребителския вход от клавиетурата и предпазва от възникване на изключения поради разминаване в типовете на данните.

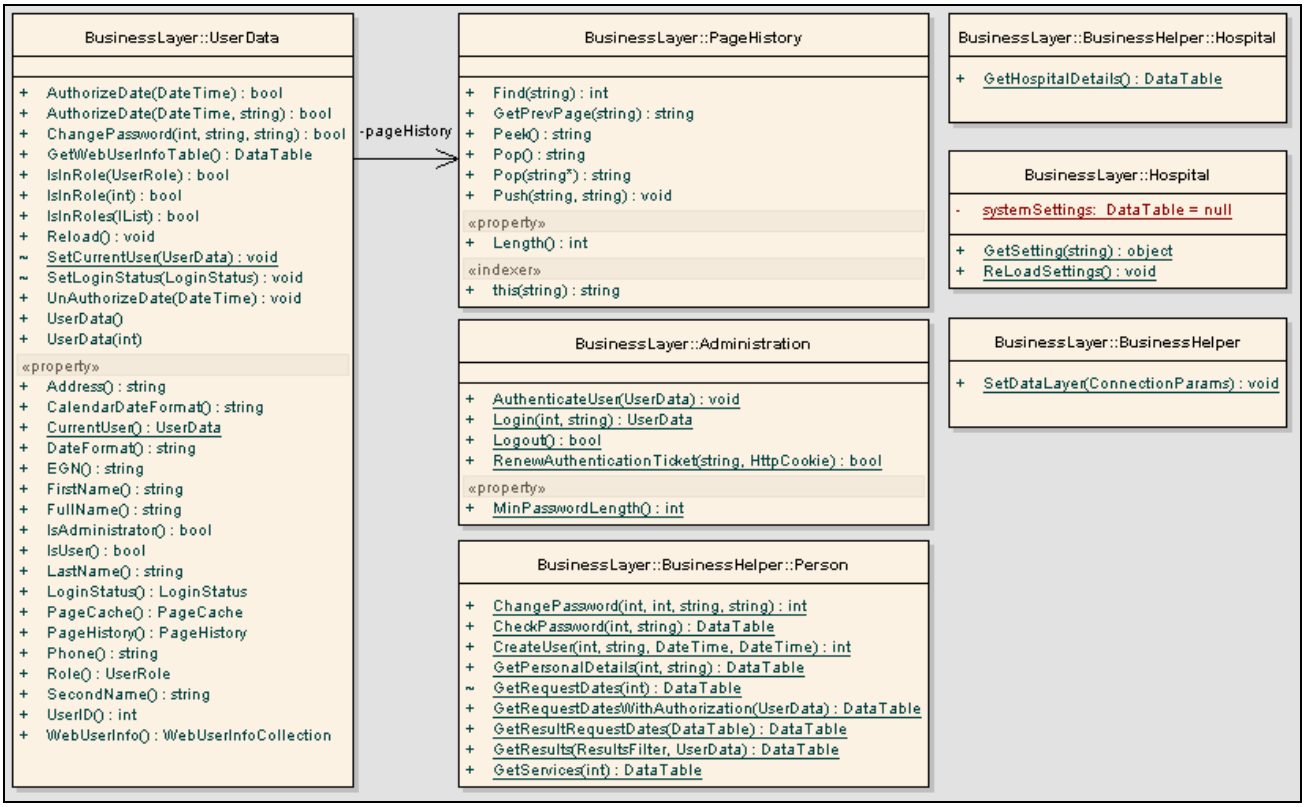
- **ResultsFilter** – Клас – контейнер за параметрите на формите за търсене.
- **PageCache** – Класът позволява съхраняването на сложни обекти между две или повече заявки към сървъра, като дефинира гъвкав механизъм за критерии за своевременно изчистване на контейнера, за да не се запълва с ненужна информация. Страницата, която поставя обект в контейнера, може да зададе страница-получател. При заявка в рамките на текущата сесия от друга страница, различна от получателя, контейнерът се изпразва.




**фиг. 16** Класова диаграма на общ модул Common

### 3.7.3. Модул за бизнес логика (BusinessLayer)

Модулът съдържа класове, които са ангажирани с биснес логиката на приложението. Основните класове на модула за показани на фиг. 17.




**фиг. 17** Класова диаграма на общ модул BusinessLayer

- **UserData** – Класът е един от основните в системата. За всеки потребител, в момента на автентикация, се създава инстанция от класа “UserData”, която се съхранява в потребителската сесия. Обектът се използва за управление на механизмите на автентикация, съхраняване на личните данни и извличане на информация за поведението на потребителя в системата. Повреме на процеса на автентикация, класът работи в тясна зависимост от класа “Administration”.
- **Administration** - Основен клас в системата. Реализаира процесите на автентикация и отмяната й. Всички методи на класа са статични.
- **BusinessHelper** – Класът се използва при всяка заявка към сървъра за инициализиране на параметрите за връзка с базата данни.

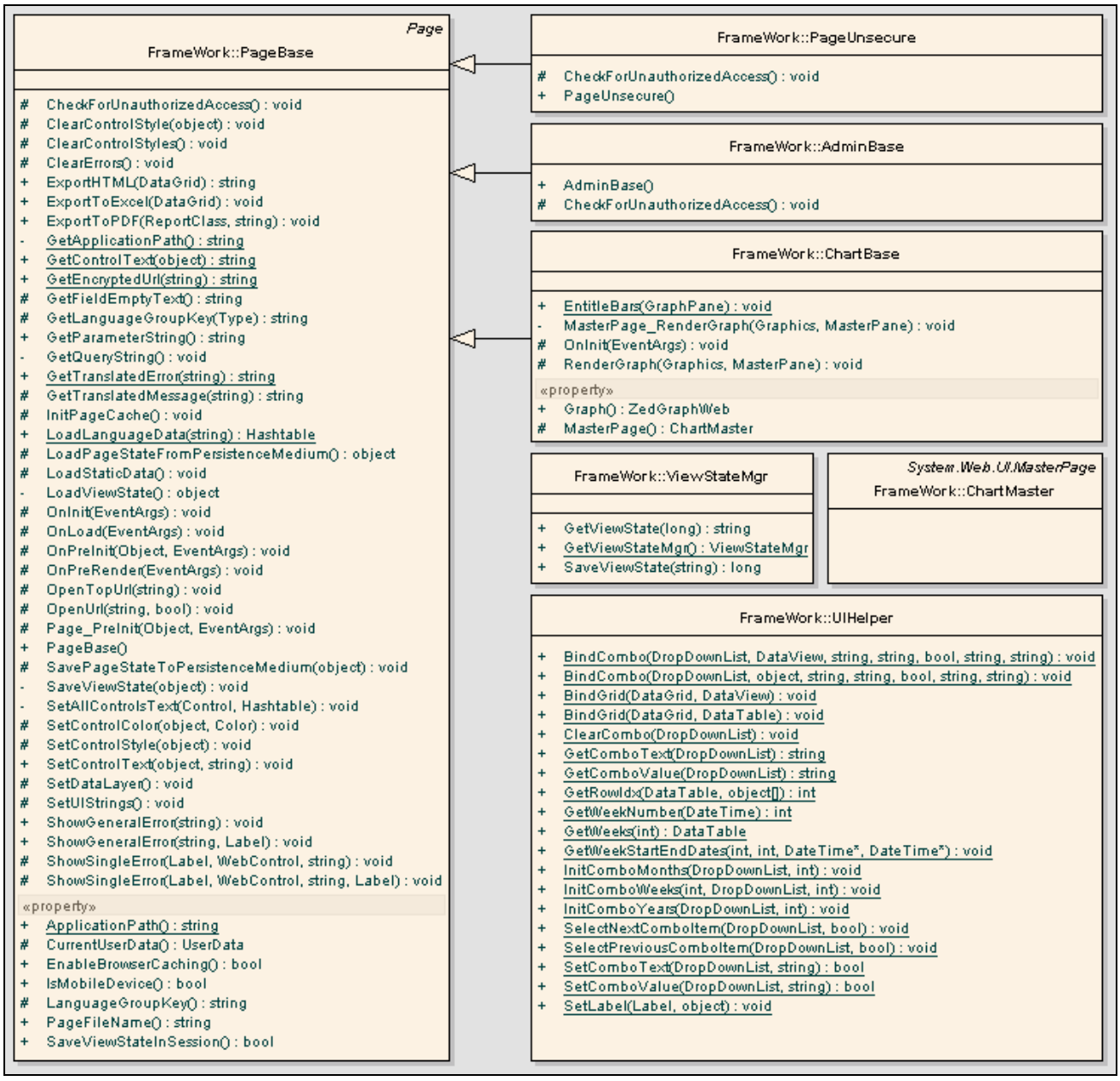
- **BusinessHelper.Person** – Класът е основен източник на данни за работа в системата за функционалностите свързани с данни за конкретен пациент. Чрез обръщения към модула за работа с базата данни, той извлича необходимата информация и я обработва допълнително, когато е необходимо. Всички методи на класа са статични.
- **BusinessHelper.Hospital** – Класът извлича данните за болничното заведение чрез обръщение към модула за работа с базата данни.
- **PageHistory** - Класът извършва трасиране на пътя на на потребителя в рамките на сайта. Реализацията наподобява цикличен стек с дълбочина десет записа. При всяка заявка към страница, в стека се записва низът на заявката заедно с параметрите (Query String), като това намира приложение във фукционалност за навигиране на потребителя до предходни страници, без да е необходимо на сървъра да се пази състояние за инициализационните параметри.
- **Hospital** – Класът се използва за кеширане на настройките от базата данни за болнично заведение или лаборатория. Извличането данните се извършва първоначално с помощта на BusinessHelper.Hospital, а в последствие от кешираните резултати. Считаме, че подходът е удачен, тъй като тези данни сравнително рядко ще бъдат променяни, а едновременно с това ще бъдат често изисквани от всички потребители на системата.

### 3.7.4. Модул на потрбителския интерфейс (MultilabWebUI)

Модулът реализира функционалността на потребителския интерфейс, която е свързана с рендирането на страниците, обработката на събития (Event Handling) и така реализира връзката с крайния потребител.

За унифицирания изглед и поведение на системата се използват общи базови класове, потребителски контроли (User Controls) и базови страници (Master Pages) за капсулиране на повтаряща се функционалност и формиране на разпредлението на дизайна (Design Layout). Използват се също стилове (Cascading Style Sheet) - за единни цветове и размери, многоезична поддръжка в реално време за интерфейса за гъвкавост при дефиниране на статичните елементи на интерфейса, скриптове за клиентска валидация и интеракция с клиента с цел подобряване на усещането на крайния потребител (примерно календар контролата), класове за съхраняване и обработка на състоянието на страниците(ViewState), помощни класове за работа с интерфейсни контроли (DataGrid, DropDownList) и др.

Ще обърнем внимание на по-важните от класовете в модула, предимно тези, които не са сторго специфични за текущото приложение. Основните класове на модула за показани на фиг. 18.




**фиг. 18** Класова диаграма на модул MultilabWebUI

- **PageBase** – Класът наследява директно от “System.Web.UI.Page” и е базов клас на всички страници в приложението. Това позволява всички промени или функционалности, които са общи за страниците да бъдат извършвани на едно централизирано място – базовия клас. Голяма част от методите на класа са виртуални, за да могат при необходимост да бъдат

предефинирани в класовете-наследници. Поради огромната важност на класа ще го разгледаме по-подробно, като се спрем на по-ключовите от методите или свойствата му:

- protected UserData CurrentUserData – Свойство (Property), предоставящо достъп на всяка от страниците до обект от тип "UserData", който съдържа информация за потребителя асоцииран с текущата сесия;
- public bool IsMobileDevice – Свойство, което идентифицира дали устройството, извършващо заявката, е мобилно или настолно;
- protected virtual void CheckForUnauthorizedAccess() – Методът се изпълнява по време на инициализацията на страницата, за да провери дали потребителят има право на достъп до нея. В случай, че такъв липсва, изпълнението на страницата се прекратява и потребителят се пренасочва към формата за вход;
- public static string GetEncryptedUrl(string Url) – методът криптира параметрите на символния низ на заявката към сървъра (Query String);
- protected virtual void SetDataLayer() – методът инициализира връзката с базата данни;
- public static Hashtable LoadLanguageData(string xmlFile) – методът прочита XML ресурсния файл с низовете за многоезичността и го кешира в паметта;
- protected virtual void ShowSingleError(Label lbl, WebControl txt, string errMsg) Методът показва съобщение за грешка, като едновременно с това маркира контрола и етикета й (caption), ако има такъв;
- protected string GetTranslatedMessage(string err) Методът превежда единично съобщение или грешка;
- public void ShowGeneralError(string errMsg) Методът се използва за показване на съобщение за грешка, което не е асоциирано с конкретно поле от потребителския интерфейс;
- private void SaveViewState(object viewState) Методът съхранява състоянието на страницата (ViewState) с сесията (в паметта на сървъра), с цел да се намали големината на страницата за клиента;
- private object LoadViewState()Методът зарежда състоянието на страницата (ViewState) от сесията и го подготвя за използване;
- public virtual void ExportToPDF(ReportClass report, string name) Методът експортира към PDF формат инстанция на "ReportClass".

- **PageUnsecure** – Класът наследява клас “PageBase” и се използва за базов в публичните страници, които не изискват автентикация.
- **AdminBase** – Класът се наследява от страниците, които са предвидени за администратирвни нужди.
- **ChartBase** – Класът се наследява от страниците, които визуализират графика. Наследниците придобиват възможността да преизползват общ код.
- **ViewStateMgr** – Класът се използва за менажиране на състоянието на страниците (ViewState).
- **ChartMaster** – Мастер (Master Page) на страниците, които визуализират графика. Наследниците придобиват възможността да използват общи потребителски контроли.
- **UIHelper** – класът реализира методи за работа с интерфейсни контроли – напълване с данни (Databinding), извличане на текстови стойнсоти и индекси, манипулиране на контролите и др.

# 4. Сигурност

Основно нефункционално изискване към системата е да осигурява защита на данните. Очакванията за системата са тя да бъде широко достъпна, което означава че в никакъв случай не може да се разчита на добронамерeността на потребителите и с цел да не се компрометират нито създателите на системата, нито болничното заведение, както и да не се застрашава нормалният ход на работа е необходимо да се вземат необходимите мерки за ограничаване на рискове от подобно естество.

Обмяната на данни между клиента и сървъра, с помощта само на административна дейност върху сайта, се конфигурира да се извършва по сигурен канал за обмяна на данни, посредством HTTPS протокола, което елиминира опасността от злонамерени “подслушвачи”.

Достъпът до вътрешната част на сайта е посредством уникален потребителски номер и парола (или набор от пароли), всяка от които период на валидност. Паролите са случайно генерирани, съхранява в базата данни криптирана с асиметричен алгоритъм SHA1 и се представят на потребителя по време на заявката му за изследване, защото за достъп до всеки ред от данни за изследвания се изисква парола активна за периода на заявката. За удобство потребителят разполага в системата с опция да редактира паролите за своите изследвания.

Възможността за наличие на индивидуална парола за всяко изследване добавя допълнително ниво на сигурност на данните, като по този начин максимално се затруднява задачата на злонамерени трети лица да придобият достъп до поверителни данни за пациент, дори да знаят една (или повече) пароли.

След като веднъж автентикацията пред системата бъде извършена със задаване на потребителско име и парола, се използва автениткационен механизъм, при който сървърът изпраща на клиента криптиран пакет (Forms Authentication Ticket) с период на валидност. Потребителят на свой ред, при отправяне на заявка към сървъра, подава обратно криптирания пакет, за да докаже пред сървъра, че вече се е автентикирал.

Оторизацията в системата (механизмът на позволяване или отказване на достъп до функционалност и информация) се реализира чрез пароли с период на валидност. Потребителят получава права да вижда резултати, които са в период на валидност на някоя от паролите, които е въвел в системата в рамките на текущата сесия. За администриране на потребителите и задаване на настройки към системата се използва настолната система Мултилаб.

Взети са мерки сайтът да не бъде податлив на злонамерени атаки върху базата данни (SQL injection). Достъпът до нея се извършва през съхранени процедури и функции. Това дава положително отражение не само на скоростта на изпълнение на SQL заявките, които в този случай са вече компилирани и готови за изпълнение, но и чрез установяване на връзка до базата с потребител, който има права само за споменатите процедури и функции, се ограничава достъпът до другите таблици и обекти на базата.

По подразбиране ASP.NET уеб-сайтовете имат включена опция за валидиране на потребителския вход за наличие HTML тагове. Тъй като в сценариите на системата потребителят няма причина да въвежда тагове, можем да считаме, че те могат да бъдат забранени. В противен случай и при липса на сериозна валидация, съществува опасност от “Cross-site scripting” атака.

В допълнение системата валидира и входа за всеки тип и размер на данните, където това е известно. В случай на проблем потребителят бива информиран за всички очаквани грешки с описателно съобщение. В случай на възникване на неочаквани грешки поведението на системата и стабилността се следят от механизъм, който регистрира тези ситуации. Неочакваните грешки се записват във файл, което предоставя възможност на системния администратор да следи проблемно поведение или нетипични заявки към сайта.

Едновременно с това крайният потребител не получава описателна информация за фрагмента от код, който е генерирал изключение.

# 5. Тестване

Тестването на системата беше извършено на няколко етапа. На консултантите от фирма “Гамаконсулт” беше представена възможността чрез отдалечен достъп да следят развитието на продукта по време на целия процес на разработка, като коментарите и препоръките им бяха решаващи за успешната реализация на проекта.

След приключване на имплементацията, системата бе предоставена за тестване на доброволни начала на няколко кръга потребители – от непрофесионалисти, до дългогодишни специалисти в областта на информационните технологии. Основната цел на експеримента беше да се получат отзиви от потребителите за общото впечатление, което приложението създава с интерфейса и функционалността си. По време на потребителското тестване системата показа изключително добри показатели на скорост и надеждност, дори когато един от тестовите уеб-сървъри беше с характеристики значително под препоръчителните (Intel Pentium II 300MHz, 384MB RAM). Сред положителните отзиви логично имаше и място за забележки, като това бяха предимно коментари по разположението на контролите, отбелязване на технически грешки или препоръки за допълнителна функционалност.

В допълнение на потребителското тестване беше извършено и автоматизирано такова на няколко етапа.

По време на разработка производителността на приложението беше изследвана с автоматизирано средство (AQ Time (20)). След извършване на необходимите измервания, при които бе следено използването на ресурсите и бързината на системата бяха извършени необходимите оптимизации, за да се премине към процеса на окончателно тестване и анализиране на резултатите.

За целта първо беше разработен собствен механизъм под формата на допълнителен клас “Trace”, като поведението му се контролира от конфигурацията на приложението. Реализацията е базирана на абониране на трасиращия клас за всички основни събития на уеб-системата и запис на определени данни в изходен файл.

Посочените резултати(Таблица 1) са осреднени и не включват първото обръщение към страниците, тъй като ASP.NET извършва компилация при първия опит за достъп до ресурс.

Тестовата постановка е извършена със значително по-слаби ресурси от нормалните за производствена среда. Сървърът на базата данни, уеб-сървърът и клиентската страна се намираха физически на една работна станция с характеристики: Pentium 4 1.8GHz, 512 RAM, Windows XP SP 2.

| Страница | Време (Сървър) s | Време (клиент) s | Пълен Размер на страницата (Kb) | Размер на страницата (без ресурси) (Kb) | Кеш (Kb) | Забележка |
|---|---|---|---|---|---|---|
| /Login.aspx | 0.031 | 0.460 | 87 | 82 | 24 | |
| /Home.aspx | 0.095 | 0.710 | 88 | 80 | 16 | |
| /Requests.aspx | 0.725 | 1.100 | 212 | 204 | 80 | |
| /Report.aspx | 1.670 | 3.720 | 304 | - | - | PDF |
| /Results | 0.290 | 1.260 | 204 | 198 | 78 | |
| /ActivityChart.aspx | 0.415 | 3.25 | | | | |
| /ChangePassword.aspx | 0.0625 | 0.515 | 80 | 75 | 13 | |

**Таблица 1** Резултати от тестването на приложението за бързина и количество на обменените данни между клиента и сървъра.

Финален тест бе извършен на функционално ниво съгласно план за тестване утвърден от консултантите.

# 6. Заключение

Представената система удовлетворява поставените към нея изисквания. Модерните и утвърдени технологии, които са избрани за реализацията, направиха процеса на разработка бърз и структуриран и доведоха до създаването на програмно решение, което напълно покрива изискванията за комерсиален софтуер. Архитектурата на приложението, технологичните решения и обектния модел спомагат за лесната поддръжка и предоставят

възможност за развитие на функционалността и скалиране, без това да налага промяна на дизайна или сериозна преработка.

Съвременните подходи, развойни средства и инструменти (ASP.NET 2.0, Microsoft SQL Server 2005, Crystal Reports 2005) отварят пред приложението възможности за допълнително развитие и ни позволяват да говорим за системата не само в сегашно, но и в бъдеще време.

Разбира се обхватът на проблемите, които реалността представя, е много по-обширен от реализираното тук и съществуват не малко възможни решения за допълнения и подобрения. Например:

**WAP** - Все по-голяма популярност придобива понятието "мобилен офис". Възможността за достъп до офис приложения, документи и Интернет, навсякъде и по всяко време представлява интерес за все по-широк кръг потребители. Ето защо е перспективно изграждането на интерфейс към системата, който да е оптимизиран за достъп през Интернет чрез мобилни телефони, джобни компютри и други портативани устройства.

**Интерфейси към външни системи** – По пътя на България към изпълнение на единна стратегия за електронно здравеопазване, независими разработчици и софтуерни компании създават продукти, които се борят за място на пазара. Тези системи в голямата си част са напълно самостоятелни и реализират по свой начин малка част от бизнес модела в едно или няколко медицински заведения. До създаването на необходимата нормативна база потърпевшите от тази липса на унифицируемост остават само и единствено обикновените пациенти. Евентуална насока за развитие на системата би могло да бъде изграждането на входно/изходна комуникация към външни системи – било през уеб-услуги (web services), било чрез процеси на експортиране и импортиране в XML или друг отворен формат на личните досиета и изследвания.

**Електронен подпис** - (20) Електронният подпис представлява данни в електронен формат, удостоверяващи самоличност/идентичност, свързани към други данни в електронен формат – електронния документ, който се подписва. Удостоверението за електронен подпис осигурява автентичност (сигурност за източника на документа) и цялостност (гаранция за липса на промени по документа от издаването му). Често работодатели или органи на местната власт изискват служебна бележка за медицинско изследване. Медицински заведения и индивидуално практикуващи лекари все по-често се снабдяват с електронен сертификат, защото се убеждават в ползата му за спестяване на време при комуникацията с държавните институции. Подписването на медицинско изследване в електронен формат с частния ключ на

медицинското заведение неоспоримо доказва произхода на документа и неговата автентичност – факт, в който теоретично всеки би могъл да се увери, с помощта на публично достъпния публичен ключ.

Разбира се допълнения биха могли да се извършват и в чисто технологичен аспект. Добавяне на асинхронен JavaScript и XML (AJAX) би допринесло за намаляване на броя презареждания на страниците, както и до по-доброто усещане на потребителите при използване на системата. Добавяне на допълнителни рапорти би повишило информативността и би спомогнало на процеса на диагностика. Добавяне на многоезична поддръжка на няколко езика би направила системата достъпна за по-широк кръг потребители, каквито са чуждите гости на страната и др.

Актуалният проблем, който настоящата дипломна работа разглежда, има далеч повече аспекти, от разгледаните по-горе. Системата моделира малка част от бизнес процесите, които протичат в една лаборатория, а и има дълъг път, който трябва да се извърви, за да можем да твърдим, че в България съществува работещо електронно здравеопазване. В световен мащаб разполагаме с отлични решения, които да ни послужат като еталон, докато, вярвам, в България не бъдат създадени системи на дори по-високо ниво. Дали представеното тук решение ще бъде част от този процес, това може да покаже само времето.

# Библиография


1. Medical Laboratory Observer, “How to succeed with a lab-hospital interface” [http://www.allbusiness.com/health-care-social-assistance/ambulatory-health-services/101878-1.html], January 1986c
2. Wikipedia, “LIS” [http://en.wikipedia.org/wiki/Laboratory_information_system], June 2007
3. Гамаконсулт, “Мултилаб” [http://www.gammaconsult.com/default.asp?id=11&lang=bg]
4. Гарантирани е-здраве и лична информация на пациента http://www.ncgbg.com/html/body_ehealth.html]
5. Министерство на здравеопазването РБ, “Стратегия за внедряване на електронно здравеопазване в България” [http://www.mh.government.bg/pdf/stategia-LAST-2006.pdf]
6. Министерство на здравеопазването РБ, “Нови информационни технологии ще подобрят достъпа на българските граждани до по-качествени медицински услуги”, [http://www.mh.government.bg/stat.php?id=1979], Декември 2006
7. SkyWare Group Ltd., iLab, [http://ilab.skyware-group.com/overview.php], 2006
8. Лаборатория CibaLab, [http://www.cibalab.com/online.php], Юли 2007
9. SCC SoftComputer, SoftWeb, [http://www.softcomputer.com/products/softweb.php], Януари 2006
10. Библиотека за 2D графика ZedGraph, [http://sourceforge.net/projects/zedgraph], 2007
11. Visual CSharp Developer Center, [http://msdn2.microsoft.com/en-us/vcsharp/default.aspx], 2007
12. Светлин Наков и колектив, .NET Framework том.1 & том.2, [http://www.devbg.org/dotnetbook/] , 2006
13. Многослойна архитектура, [http://en.wikipedia.org/wiki/Multitier_architecture], Юни 2007
14. Sparx Systems, UML Компонентна диаграма (Component Diagram), [http://www.sparxsystems.com.au/resources/uml2_tutorial/uml2_componentdiagram.html], 2007
15. Sparx Systems, UML диаграма на разгръщане (Deployment Diagram), [http://www.sparxsystems.com.au/resources/uml2_tutorial/uml2_deploymentdiagram.html], 2007
16. Sparx Systems, UML диаграма на сценариите (Use Case Diagram), [http://www.sparxsystems.com.au/resources/uml2_tutorial/uml2_usecasediagram.html], 2007

17. Cryptanalysis of SHA-1, [http://www.schneier.com/blog/archives/2005/02/cryptanalysis_o.html], Февруари 2005
18. Wikipedia, “AES”, [http://en.wikipedia.org/wiki/Advanced_Encryption_Standard], Юли 2007
19. Automated QA, AQ time, [http://www.automatedqa.com/products/aqtime/index.asp], 2007
20. Българска търговско-промишлена палата, “Електронен Подпис”, [http://www.bcci.bg/bulgarian/ecertification/Q&A.htm], 2003