|  |  |
| --- | --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  Федеральное государственное автономное образовательное учреждение высшего образования | |
| **Национальный исследовательский ядерный университет «МИФИ»** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | |
|  |  |
| Факультет/Институт ИФИБ | Кафедра №46 «Компьютерные медицинские системы» |
| Специальность  (направление) 14.05.04 | Группа С13-301 |

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ**

**РАБОТА НА ТЕМУ:**

**«Система диагностики рака предстательной железы с применением искусственного интеллекта»**

|  |  |  |
| --- | --- | --- |
| Студент-дипломник | Матюхин Н.А. | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |
| Руководитель ВКР | Доцент, к.т.н.  Проничев А.Н. | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |
| Рецензент | Доцент, к.т.н.  Дмитриева В.В. | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |
| Заведующий кафедрой | Профессор, д.т.н. Никитаев В.Г. | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |

Москва 2019 г.

**Аннотация**

Пояснительная записка к выпускной квалификационной работе состоит из 6 подразделов. К записке прилагается приложение, включающее в себя 5 частей:

* + - 1. Глоссарий терминов;
      2. Документация разработчика системы диагностики рака предстательной железы;
      3. Лабораторная работа по системе для студентов IT-специальностей;
      4. Лабораторная работа по системе для студентов медицинских учреждений;
      5. Код программы.

Записка включает в себя 81 страницу, 31 изображение, 5 приложений.

**Реферат**

**Ключевые слова:** программирование, предстательная железа, распознавание, сегментация.

Целью дипломного проекта является разработка систему для диагностики стадий рака предстательной железы .

Пояснительную записку условно можно разделить на три основных раздела:

Постановка задачи дипломного проекта, содержащая следующее:

1. Цель дипломного проекта;
2. Задачи, решение которых необходимо в дипломном проекте;
3. Исходные данные для дипломного проекта.

Теоретическая часть:

1. Анализ предметной области;
2. Анализ объектной среды;
3. Анализ инструментальной среды;
4. Формирование требований к системе и подсистемам;

Практическая часть:

1. Концептуальная модель системы;
2. Проектное решение;
3. Реализация проекта;
4. Эксперимент;
5. Методика работы с программой;
6. Заключение.

Результат дипломного проекта – разработанная программа, руководства по использованию, пояснительная записка. Записка включает в себя 81 страницу, 31 изображение, 4 приложений.

**Задание на дипломный проект**

1. Фамилия, имя, отчество студента Матюхин Николай Андреевич.
2. Тема проекта Система диагностики рака предстательной железы с применением искусственного интеллекта.
3. Срок сдачи студентом готового проекта январь 2019 .
4. Место выполнения проекта НИЯУ МИФИ, кафедра №46 .
5. Руководитель проекта к.т.н. Проничев Александр Николаевич, НИЯУ МИФИ, заместитель заведующего кафедрой №46 «Компьютерные медицинские системы» .

(фамилия, имя, отчество, должность, место работы)

1. Цель работы : Разработать систему диагностики рака предстательной железы с применением искусственного интеллекта

2. Задание:

а) анализ литературы и обзор работ, связанных с проектом

* 1. Аль-Шукри С.Х., Ткачук В.Н. Опухоли мочеполовых органов. СПб; 2015; 309 с.
  2. Библиотека патологоанатома // Сертификация, стандарты и качество патологоанатомических исследований; Вып. 69, Санкт-Петербург: 2016.
  3. Бухаркин Б.В., Подрегульский К.Э. Рак предстательной железы. Клин онкология 2009; 1(1): 5—8.
  4. Грачева Л. Выявление и предоперационный прогноз клинически незначимого рака предстательной железы / / РМЖ. Т. 5. М4. -2007.
  5. Карякин О.Б. Приоритеты в лечении различных стадий рака предстательной железы. Материалы II Российской онкологической конференции. Современные тенденции развития лекарственной терапии опухолей. 2014 М.
  6. Коган М.И., Якимчук Т.П., Шишков А.В., Волдохин А.В. Сравнтельный анализ диагностических методов при первичном обследовании больных раком простаты. Урол и нефрол 2009; 3: 38—41.
  7. Козлов В.П., Мазанов г.п., Ментешов И.В. Отдаленные результаты комбинированного лечения рака предстательной железы // Кремлевская медицина. Клин. вестн. 2012.-М3.
  8. Лоран О.Б., Пушкарь Д.Ю., Канн яд. и др. Радикальная простатэктомия у больных раком предстательной железы без признаков экстракапсулярной инвазии / В кн.: Актуальные вопросы лечения онкоурологических заболеваний. Материалы II Всероссийской конф. Обнинск, 2017.

б) расчетно-конструкторская, теоретическая, технологическая часть

- Обзор литературы и статей по диагностике рака предстательной железы

- Анализ систем и подсистем по данной теме;

- Анализ исходных данных;

- Проведение предпроектного исследования;

- Планирование разработки системы;

- Разработка технического задания к системе;

- Формирование концептуальной модели программы;

- Формирование математической модели программы;

- Реализация программы;

- Разработка технического задания к базе данных;

- Формирование концептуальной модели базы данных;

- Разработка логической модели базы;

- Реализация базы данных;

- Разработка плана тестирования программы;

- Проведение тестирования;

- Оценка результатов тестирования;

- Разработка методики работы с программой;

- Разработка лабораторных работ по данной системе;

- Разработка руководства разработчика по созданной системе;

- Разработка руководства пользователя по созданной системе.

Отчетный материал проекта:

а) пояснительная записка;

б) графический материал (презентация);

- Цель ВКР;

- Задачи;

- Разработка системы;

- Описание программы;

- Выводы по ВКР.

**Календарный план работы над проектом**

(составляется руководителем с участием студента в течении первой недели с начала дипломного проектирования)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п\п | Наименование этапов работы | Сроки выполнения этапов | Степень готовности проекта в % к объему работы | Время выполнения |
| 1 | - Предпроектное исследование (Анализ предметной области, анализ объектной среды, анализ инструментальной среды) | 07.12.2018 – 18.12.2018 | 20 | 12 дней |
| 2 | - Разработка концептуальной модели системы;  - Разработка требований к системе и подсистемам | 19.12.2018 – 25.12.2018 | 30 | 7 дней |
| 3 | Проектирование системы | 26.12.2018 – 02.01.2018 | 50 | 8 дней |
| 4 | Физическая реализация | 03.01.2019 – 23.01.2019 | 90 | 21 день |
| 5 | Тестирование, экспериментальные исследования | 24.01.2019 – 26.01.2019 | 95 | 3 дня |
| 6 | Разработка методики применения разработанной системы | 27.01.2019 – 29.01.2019 | 100 | 3 дня |

Дата выдачи задания «07» декабря 2018 г.

|  |  |  |
| --- | --- | --- |
|  | Руководители дипломного проекта | (подпись, фамилия, имя, отчество) |
|  |  |  |
|  |  |  |

Задание принял к исполнению

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(фамилия, имя, отчество)

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ г.

**Содержание**

|  |  |
| --- | --- |
| Введение | 4 |
| 1. Предпроектное исследование | 7 |
| 1.1. Анализ предметной области | 7 |
| 1.2. Медицина | 8 |
| 1.3. Градации Глисона | 12 |
| 1.3.1. Градация 1 | 13 |
| 1.3.2. Градация 2 | 14 |
| 1.3.3. Градация 3 | 14 |
| 1.3.4. Градация 4 | 15 |
| 1.3.5. Градация 5 | 16 |
| 1.4. Информационные технологии | 17 |
| 1.4.1. Анализ инструментальной среды | 17 |
| 1.4.2. Анализ объектной среды | 18 |
| 1.4.3. Математическое обеспечение | 18 |
| 1.4.4. Программное обеспечение | 18 |
| 1.4.5. Аппаратное обеспечение | 19 |
| 1.4.6. Информационное обеспечение | 19 |
| 1.4.7. Методическое обеспечение | 20 |
| 2.Требования | 21 |
| 2.1. Требования к системе и подсистемам | 21 |
| 2.2. Требования системы | 21 |
| 3. Проектирование системы | 24 |
| 3.1. Архитектура системы | 24 |
| 3.2. Проектирование базы данных | 27 |
| 4 Физическая реализация | 30 |
| 4.1. Состав исходных файлов необходимы для сборки программы | 30 |
| 4.2. Основные модули программы | 33 |
| 4.2.1. Модуль логирования | 33 |
| 4.2.2. Модуль хранения общих данных | 35 |
| 4.2.3. Модуль управления переходами между экранами | 35 |
| 4.2.4. Модуль доступа к базе данных | 41 |
| 4.2.5. Модуль фильтров | 42 |
| 4.2.6. Модуль распознавателей | 43 |
| 4.3. Интерфейс программы | 45 |
| 5 Методическое обеспечение | 48 |
| 5.1 Состав методического обеспечения | 48 |
| 5.2 Руководство пользователя | 48 |
| 5.3 Руководство разработчика | 53 |
| 5.4 Разработка лабораторных работ | 72 |
| 6 Экспериментальные исследования | 74 |
| 6.1.Тестирование работоспособности программы | 74 |
| 6.1.1.Цель и проведение тестирования. | 74 |
| 6.1.2. Результаты тестирования | 75 |
| 6.2 Оценка эффективности работы системы при обработке реальных данных. | 79 |
| 6.2.1. Цель и план эксперимента. | 79 |
| 6.2.2. Результаты эксперимента. | 80 |
| Заключение | 83 |
| Глоссарий | 84 |
| Список используемой литературы | 85 |
| Приложение 1 |  |
| Приложение 2 |  |
| Приложение 3 |  |
| Приложение 4 |  |

**Введение**

Рак предстательной железы (РПЖ) — одно из онкологических заболеваний у мужчин, оно стоит на втором месте по уровню смертности, обусловленной раком. Рак предстательной железы редко встречается в возрасте до 50 лет, но его частота, так же, как и смертность от этого заболевания, неуклонно увеличивается с возрастом и достигает максимума на девятом десятке лет жизни.

Современные возможности диагностики РПЖ позволяют выявлять заболевание на ранних стадиях, однако более чем у 50—70% больных на момент диагностики обнаруживают 3—4-ю стадии заболевания. На ранних стадиях развития РПЖ протекает бессимптомно, диагноз часто ставится случайно при ректальном обследовании по поводу других заболеваний, в основном при подозрении на доброкачественную гиперплазию простаты. Другими словами, симптоматика РПЖ складывается из симптомов поражения соседних органов и самой предстательной железы.

Наиболее частой формой рака является аденокарцинома (синонимы: дифференцированная аденокарцинома, криброзный рак, альвеолярный рак, альвеолярная аденокарцинома, слизистый рак и др.) – злокачественная опухоль, возникающая из эпителия протоков и (или) ацинусов.

Клинически опухоль может протекать бессимптомно, по мере прогрессирования появляются дизурические явления. Метастазирует в кости и регионарные лимфатические узлы, реже в другие органы. Иногда опухоль прорастает в прилежащие органы малого таза и изредка в кости таза.

Из неэпителиальных злокачественных опухолей в предстательной железе может встречаться рабдомиосаркома, лейомиосаркома и др.

Чрезвычайно высокая актуальность проблемы рака предстательной железы привлекает внимание исследователей различных стран. В течение последних десятилетий появились сообщения об уровне заболеваемости. Выяснены иммунологические аспекты онкологического процесса в простате, а также сывороточные и иммунные маркеры болезни. Получили широкое применение при обследовании пациентов с подозрением на РПЖ современные диагностические клинико-лабораторные (ПСА и его дериваты) и клинико-инструментальные методы исследований. Однако, несмотря на кажущееся изобилие исследователей, эта тема требует постоянного наблюдения и осмысления.

Целью исследования является целью данного исследования является автоматизация диагностики рака предстательной железы, а так же увеличение точности диагностики и сведения к минимуму ошибок при постановке диагноза при болезни предстательной железы.

Для достижения намеченной цели поставлены и решены следующие задачи:

1. Ознакомиться с особенностями строения предстательной железы
2. Изучить современные диагностики рака предстательной железы
3. Ознакомиться с градацией Глиссона
4. Найти и изучить информацию выражения признаков шкалы Глиссона на цифровых изображениях биоптата предстательной железы
5. Оценить доступные средства для обработки цифровых изображений, а так же выделения на них особенностей, кластеризации и распознавания образов на цифровых изображениях
6. На основе доступных средств а так же математического обеспечения оценить возможные сложности при создании программного комплекса для интерактивного распознавания стадий Глиссона.
7. Спроектировать программный комплекс
8. Реализовать программный комплекс.
9. Составить тестовую модель и набор тестовых сценариев для оценки качества программного обеспечения
10. Провести тестирование, на основе тестирования составить отчет о применимости данного программного обеспечения в условиях промышленной эксплуатации

Практическая значимость работы заключается в том, что данная программа помогаем врачу более наглядно выделить части фотографии, на которые ему нужно обратить внимание, то есть визуально подсветить ацинусы, а также позволяет количественно рассчитать отношение площади, занимаемой пораженными ацинусами, к площади, которую занимают здоровые клетки. Такая возможность позволит врачам, избежать спорных ситуаций при определении стадии рака по шкале Глиссона.

**1. Предпроектное исследование**

Предпроектное исследование – это комплекс аналитических и специальных технических мероприятий, производимых с целью сбора исходной информации об архитектурных, функциональных и технологических параметрах объекта.

Цель: Определить и обосновать требования к системе и ее подсистемам.

Данный вид исследования можно разбить на 3 основные составляющие:

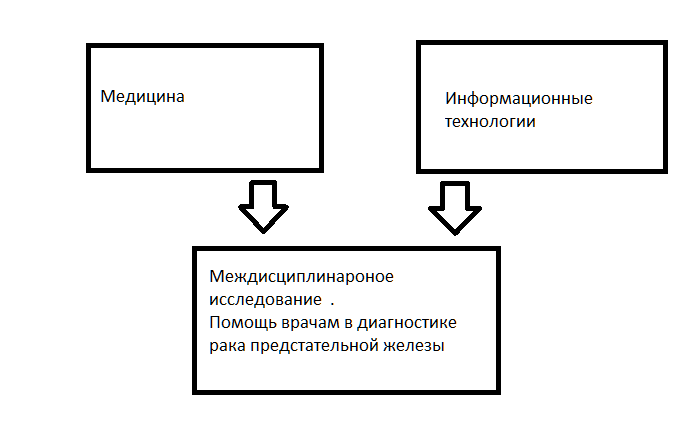
* анализ предметной области,
* анализ объектной среды
* анализ инструментальной среды.

**1.1. Анализ предметной области**

Целью проведения анализа предметной области является определение структуры системы для создания концептуальной модели, а также состава подсистем.

Предметной областью в задаче создания системы диагностики рассмотрение существующих методов телемедицины и способов обработки и передачи информации.

Разработка системы проводится на основе междисциплинарного исследования (рис. 1).



**Рис.1.** Междисциплинарные исследования

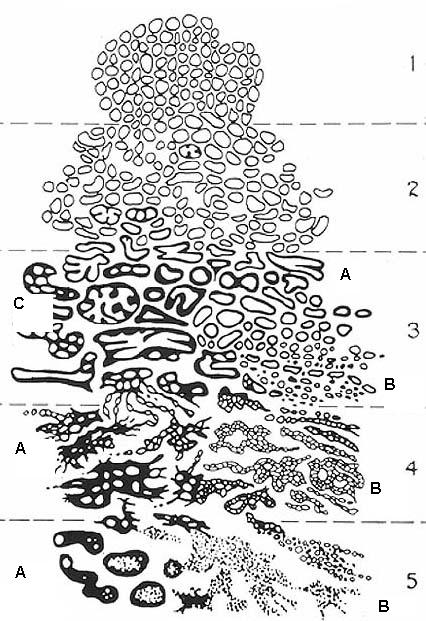
**1.2. Медицина**

При обнаружении в биоптате картины рака предстательной железы необходимо определение степени дифференцировки опухоли, от которой во многом зависит ее биологическое поведение и, следовательно, дальнейшая лечебная тактика. Общепринятая в России в настоящее время “Гистологическая классификация ВОЗ опухолей предстательной железы” основное внимание уделяет формально-гистологическому типу опухоли, который не является определяющим в плане установления ее биологического поведения и выбора наиболее эффективных методов лечения. В этом отношении более продуктивна система степеней гистологической дифференцировки рака предстательной железы по Глисону.

В 1966 году Глисоном была предложена система градации рака предстательной железы, получившая всемирное признание не только среди патологов, но и урологов, онкологов и радиологов. Система основана на том, что рак предстательной железы образует разные гистологические структуры, в нем наряду с высокодифференцированными участками, могут встречаться участки низкодифференцированного рака. По системе Глисона в препарате рака предстательной железы регистрируются два наиболее характерных участка, которые оцениваются по пятибалльной шкале. Индекс Глисона получается в результате суммирования баллов. Индексы 2 – 5 соответствуют высокодифференцированному, 9 – 10 – низкодифференцированному, 6 – 8 – раку со средней степенью дифференцировки (схема 2).

Существует прямая корреляция между индексом Глисона и способностью опухоли к инвазии и метастазированию, прогнозом заболевания и выживаемостью пациентов.

Поскольку в биоптатах простаты, полученных из разных ее участков, степень злокачественности опухоли различна, то принято указывать ее диапазон у конкретного больного.

**Схема 2.** Градации аденокарциномы предстательной железы по Глисону.

1. Плотно упакованные мелкие, однообразные железы, разделенные узкими прослойками стромы; четкий край опухоли.
2. Рыхло упакованные железы, несколько отличающиеся друг от друга размерами и формой; неправильные очертания опухоли.

3.А. Рассредоточенные крупные железы с неправильными очертаниями.

3:В Рассредоточенные, очень мелкие железы.

3:С – папиллярные и криброзные массивные комплексы с гладкими контурами.

4:А – слившиеся железы, «рваная» зона инвазии.

4:В – такого же характера структуры, образованные большими светлыми клетками («гипернефроидный» вариант).

5:А – почти солидные округлые массы с центральным некрозом («комедокарцинома»).

5:В – анапластические струтуры, «рваная» зона инвазии.

При анализе выживаемости больных Глисон обнаружил, что в случаях, когда в разных участках опухоль имела разное строение, уровень смертности оказывался между двух ожидаемых значений. Он предложил использовать оба показателя, что и составило «сумму Глисона». Величина этой суммы колеблется от 2 баллов (1+1) до 10 (5+5). Одинаковые цифры суммируются, когда опухоль имеет однотипную характеристику по дифференцировке в разных участках.

«Сумма Глисона» оказалась чрезвычайно точным прогностическим признаком, особенно значимым является показатель 7 и более, свидетельствующий о высоком проценте инвазии капсулы простаты, семенных пузырьков и метастатического поражения лимфатических узлов.

Согласно этой системе различают пять степеней дифференцировки опухоли. При изучении препаратов с наличием рака предстательной железы патолог должен выделить две основные картины: “первичную”, или преобладающую, занимающую наибольшую площадь образца, и “вторичную”, т. е. вторую по величине, а затем определить степень дифференцировки по Глисону каждой из них. Дополнительные участки строения меньших площадей и участки, занимающие меньше 5% общей площади рака, игнорируются. Сумма степеней и определяет индекс Глисона. В случае практически полной однородности структуры для получения индекса определяемая степень Глисона удваивается. Чем ниже индекс Глисона, тем эффективнее лечение и лучше прогноз для больного.

Классификация Глисона имеет две уникальные особенности. Вопервых, она основана только на структуре тканей образца простаты, цитологические особенности не оцениваются. Второй уникальной особенностью классификации Глисона является то, что в ее основе не лежит самая высокая (наименее дифференцированная) гистологическая градация в образце опухоли. Большинство аденокарцином предстательной железы не являются однородными и могут включать множество гистологических вариантов. Однако прогноз при раке простаты определяется соотношением наиболее представленной гистологической градации аденокарциномы и второй по встречаемости градации. Следовательно, в системе Глисона, суммируются преобладающая и вторая по распространенности градация (если она составляет не менее 5 % всей опухоли), таким образом, высчитывается показатель Глисона, например показатель Глисона равен 3+4=7 (необходимо, чтобы велась запись встреченных образцов самой распространенной и второй по распространенности градаций, с последующим определением показателя Глисона).

К синонимам «показатель Глисона» относятся «сумма Глисона», «комбинированный показатель Глисона». Если в опухоли имеется только одна градация, то для получения показателя Глисона необходимо удвоить номер градации, т.е. если вся опухоль состоит из градации №3, то показатель Глисона равен 3+3=6. Иногда вместе с двумя ведущими градациями встречается третья по распространенности градация. Если же она, или даже четвертая по распространенности градация имеет 4-й или 5-й номер, о ней необходимо упоминать после указания показателя Глисона, даже если они составляют менее 5% опухоли, например показатель Глисона 3+3=6, с четвертой по встречаемости градацией №4.

Определение индекса Глисона при обнаружении рака в биоптате предстательной железы совершенно необходимо. Однако и система Глисона не лишена недостатков. Во-первых, по биоптату не всегда удается определить биологическое поведение опухоли для индивидуального больного: рак предстательной железы характеризуется большим разнообразием гистологических структур даже в пределах одной опухоли, а биопсийный образец имеет весьма небольшие размеры. Во-вторых, биологическое поведение опухоли может быть обусловлено потерей дифференцировки на небольших  участках и не связано с доминирующими структурами.

Более точное предсказание о поведении опухоли можно сделать на основании ее объема, определяемого при радикальной простатэктомии. Объем опухоли, как правило, коррелирует со степенью ее дифференцировки и в равной степени с наличием метастазов в лимфатические узлы и отдаленных метастазов. Поэтому он является важным фактором в определении клинической стадии заболевания и в сочетании со степенью дифференцировки наиболее точно отражает биологическое поведение опухоли. Данные по Глисону показывают достаточно высокую корреляцию между показателем Глисона и прогнозом болезни.

**1.3. Градации Глисона**

Первые три степени (степени 1, 2, 3), расцениваемые как наиболее хорошо дифференцированные, сходны по структуре с нормальной предстательной железой. Эпителиальные ткани хорошо отграничены от окружающей стромы, железы образованы одним слоем эпителиальных клеток, окружающих железистые просветы. Тем не менее, выявляется менее упорядоченное ветвление желез по сравнению с доброкачественными простатическими протоками и ацинусами. Злокачественные железистые структуры имеют тенденцию к разнообразию размеров и формы желез, величины просветов.

Различия первых трех степеней незначительны. Если железы однородны, а признаки инвазии отсутствуют или выражены слабо, опухоль может быть отнесена к 1-й или 2-й степени дифференцировки. Эти степени могут быть также установлены, если клетки имеют светлую цитоплазму, которая более, чем темная, увеличивает сходство с нормальной тканью. В очень маленьких образцах отличие от нормальной ткани должно основываться не только на архитектонике, оно также должно быть подтверждено цитологическими критериями, например, очень крупными ядрышками. Важным исключением из этих правил определения хорошо дифференцированных опухолей является криброзный вариант, который также относится к 3-й степени дифференцировки по Глисону. Отличие криброзного варианта от рака 4-й степени дифференцировки — небольшие размеры опухолевых комплексов, сопоставимые с размерами крупных протоков, округлая форма, четкие границы, хорошо выраженный стромальный компонент.

Опухоли 4-й степени дифференцировки по Глисону характеризуются большим разнообразием гистологических проявлений и отсутствием сформированных желез, свойственных первым трем степеням.

Опухоли 5-й степени дифференцировки являются наиболее слабо дифференцированными, с едва различимыми или практически отсутствующими железистыми структурами, с крупными полями опухолевых клеток практически без каких-либо просветов. Опухолевые клетки обычно имеют скудную цитоплазму. К 5-й степени дифференцировки обычно относят рак типа камедокарциномы. Под этим вариантом описывается интрадуктальный рак с наличием небольших криброзных участков по периферии и с тотальным некрозом клеток в центре опухолевых комплексов. Это единственный вариант рака предстательной железы, в котором отмечается некроз опухолевых клеток. Статистически доказано, что наличие подобных неинвазивных участков связано с крайне плохим прогнозом, сопоставимым с прогнозом при инвазивном раке 5-й степени дифференцировки.

**1.3.1. Градация 1**

Первая градация состоит из очень хорошо ограниченного узелка одинарных, обособленных, очень тесно расположенных желез, которые не инфильтрируют прилегающую нормальную ткань предстательной железы. Железы круглые или овальные. Они достаточно большие, в сравнении с большинством опухолей с 3-й градацией, и имеют приблизительно одинаковую форму и размер. Цитоплазма клеток аденокарцином 1 и 2 градаций обычно несколько окрашивается бледнее, чем в окружающих нормальных тканях, однако эти данные не учитываются в классификации Глисона. Градация №1 встречается очень редко, обычно в зонах формирования аденокарцином. Если же она обнаруживается, она обычно ассоциирована с опухолью 2 градации. Более того, дифференциальная диагностика между первой и второй градацией не столь существенна, т.к. обе градации имеют сходный прогноз (рисунок 1-2 Приложение 1).

**1.3.2. Градация 2**

Вторая градация состоит из достаточно хорошо ограниченного узелка одинарных, обособленных желез. Однако, железы более свободно расположены, и не так схожи друг с другом, как в первой градации. Может наблюдаться минимальная инвазия неопластических желез в окружающую нормальную ткань предстательной железы. Как и при 1 градации, размер желез в основном больше чем большинства опухолей с 3 градацией. Они круглые или овальные, и имеют открытый просвет. Железы сглажены и закруглены, не имеют угловую форму, как при градации 3. Как и при 1 градации, цитоплазма клеток 2 градации обычно больше по размерам и слабо окрашена, чем у опухолей средних градаций. Вторая градация также, но не всегда, встречается в зонах формирования карцином (рисунок 3 Приложение 1).

**1.3.3. Градация 3**

Третья градация, в отличие от первой и второй, имеет очевидный инфильтративный характер с инвазией в прилегающие нормальные ткани простаты. Железы в пределах третьей градации различаются по форме и размеру, часто удлинены или имеют угловую форму. Они обычно маленького размера («микрожелезы»), меньше чем железы в первой и второй градациях. Однако, некоторые железы третьей градации могут иметь размер от среднего до большого. Маленькие железы третьей градации, в отличие от маленьких слабо дифференцированных желез четвертой градации, являются отдельными железистыми единицами, поэтому мысленно возможно нарисовать окружность вокруг каждой из них. Часто цитоплазма желез 3 градации (хотя оценка цитоплазмы не включается в систему Глиссона) более интенсивно окрашена, что может быть необходимо для распознавания третьей градации от первой и второй.

Криброзные железы также могут составлять третью градацию по Глисону. Такие железы несколько больше в размере, чем нормальные простатические железы, и имеют правильный наружный контур. Они внешне напоминают внутрипротоковую криброзную карциному молочной железы. Криброзные железы третьей градации необходимо дифференцировать от таковых четвертой градации, внутрипротоковой криброзной пролиферации и простатической протоковой аденокарциномы. Небольшие инфильтрирующие железы третьей градации почти всегда сопровождают криброзные железы третьей градации (рисунок 4-17 Приложение 1).

**1.3.4. Градация 4**

Железы четвертой градации уже не отделены друг от друга, как в 1-3 градациях. Они выглядят как сливающиеся, плохо дифференцированные железы, с редко определяемым просветом. Иными словами, они выглядят как криброзные.

Слившиеся железы представляют собой цепочки, гнезда, или железистые массы, которые больше не разделены друг от друга промежуточной стромой. Слившиеся железы содержат малочисленные участки стромы, что может внешне напоминать частичное разделение желез. Следовательно, слившиеся железы могут иметь фестончатый характер по периферии. «Гипернефроидные структуры», описанные Глисоном, является редким вариантом слившихся желез, напоминающим почечно-клеточную карциному.

В оригинальном описании Глисона четвертая градация не включала плохо дифференцированных желез. Однако, считается, что плохо дифференцированные железы следует включить в четвертую градацию. Различение малых ацинарных желез третьей градации и плохо дифференцированных желез четвертой градации временами может оказываться затруднительным.

Криброзные железы четвертой градации имеют или большие размеры, включая криброзные пластины, или малые криброзные железы с неправильными инфильтрирующими окружающие ткани краями. Малые криброзные железы с неправильными инфильтрирующими краями четвертой градации необходимо дифференцировать от криброзных желез третьей градации, в которой малые криброзные железы имеют правильную форму краев. При биопсии простаты под фрагментами криброзной карциномы подразумевают большие криброзные железы или пласты с небольшим количеством окружающей стромы, что определяется как четвертая градация. Важно различать третью градацию от четвертой, потому что это позволяет различить показатели Глисона 6 и 7, а последний из них имеет значительно худший прогноз (рисунок 18-73 Приложение 1).

**1.3.5. Градация 5**

В пятой градации железы практически не дифференцированы. Она состоит из плотных тяжей или одиночных клеток. Опухолевые гнезда с центральным комедонекрозом также классифицируются как пятая градация. Данные о том, можно ли относить криброзные железы, характерные для четвертой градации, к пятой градации, если имеет место комедонекроз, противоречивы. Разделение плохо дифференцированных желез четвертой градации от скопления опухолевых клеток в виде гнезд и тяжей, в которых практически отсутствуют элементы дифференцировки, или имеются только вакуоли, является очень сложным. Однако, это разделение не является слишком существенным, потому что в этом случае любая комбинация двух градаций даст показатель Глисона 8, 9 или 10, все из которых относятся к низко дифференцированным опухолям (рисунок 74-81 Приложение 1).

**1.4. Информационные технологии**

**1.4.1. Анализ инструментальной среды**

Целью проведения данного вида исследований является разработка требований к подсистемам.

Структуру информационной системы составляет совокупность отдельных ее частей, называемых подсистемами. Итак, подсистема – это часть системы, выделенная по какому-либо признаку.

Общую структуру информационной системы можно рассматривать как совокупность подсистем независимо от сферы применения. Таким образом, структура любой информационной системы может быть представлена совокупностью обеспечивающих подсистем(Рис. 2)



**Рис. 2** Обеспечения информационной системы

**1.4.2. Анализ объектной среды**

Объектом исследования являются цветные фотографии биоптата предстательной железы размера 500 на 500 пикселей, формата jpeg

**1.4.3. Математическое обеспечение**

Математическое обеспечение – совокупность математических методов, моделей, алгоритмов обработки информации, используемых при решении задач в информационной системе (функциональных и автоматизации проектирования информационных систем).

Создание математического обеспечения заключается в разработке алгоритмов и машинных программ задач, решаемых по подсистемам.

К математическому обеспечению в случае данной работы относятся алгоритмы формирования наполнения базы данных.

Этап формирования базы данных представляет собой наполнение информацией файла. Для этого разрабатывается подсистема, функционал которой направлен на реализацию данного этапа.

**1.4.4. Программное обеспечение**

При разработке программного обеспечения следует верно подобрать необходимый инструментарий, в нашем случае языки программирования и среду разработки.

Были рассмотрены языки C++, Java, С# и Python для реализации поставленной задачи, но в конечном итоге было принято решение использовать язык C++, из-за наличия на кафедре разработанных библиотек для облегчения написания программы. Программа будет написана в среде Qt Creator 5.8.0. на языке C++ с компилятором Mingv 4.3 win 32. И с использованием базы данных SqlLite

**1.4.5. Аппаратное обеспечение**

Были рассмотрены варианты потенциально необходимого аппаратного обеспечения, но в итоге было заключено, что никакой специфической аппаратной поддержки быть не должно и программа должна работать на любой ЭВМ.

Минимальный набор комплектующих для нормального функционирования системы:

1. Жесткий диск;

2. Дисплей;

3. Клавиатура;

4. Оперативная память;

5. Материнская плата;

6. Процессор;

7. Блок питания

Если любого из выше указанных устройств не хватает или с ними проблемы (поломка), то система не запуститься, а также будет выведена ошибка компьютера.

**1.4.6. Информационное обеспечение**

Информационное обеспечение в данной работе включает в себя СУБД для хранения исследований, информации об исследовании, изображений исследований и информации о запросе по данному исследованию, который был сделан с помощью взаимодействия врачей средствами разрабатываемой системы.

SQLite - мощная из-за технологий ее сервисных библиотек. Наиболее быстрая СУБД, благодаря базированию ее на файловой структуре, что дает нам, вследствие, более широкий набора инструментария работы нежели сетевые СУБД.

После оценки преимуществ и недостатков приведённых баз данных можно сделать следующие выводы: «SQLite» быстрая, функциональная

СУБД, которая подходит для небольших локальных программ. Таким образом было принято решение выбрать СУБД «SQLite», которая обладает достаточно широким функционалом, хорошей производительностью и относительной простотой.

**1.4.7. Методическое обеспечение**

Методическое обеспечение включает в себя руководство пользователя и разработчика к разрабатываемой системе. В руководстве пользователя описаны шаги работы с системой и перечислены все элементы интерфейса с их предназначением при работе с системой. В руководстве разработчика описаны шаги для сборки системы из ее исходных кодов. Также в руководстве разработчика приводится описание всех функций и переменных в исходном коде системы.

**2.Требования**

**2.1. Требования к системе и подсистемам**

**Требования** – это точное описание полезных для пользователя характеристик, ожидаемых от продукта.

Продуктов в данном случае является система, которая может выделить важные при диагностике рака предстательной железы области на фотографии биоптата предстательной железы и выполнить распознавание стадии Глисона. А также позволяет сохранить результат распознавания в базу знаний и сравнить полученный результат с похожими результатами из базы знаний.

**Название системы**

Помощь в диагностике рака предстательной железы

**Назначение системы**

Обработка и получение информации с медицинских устройств с возможность сохранения и отправки запроса по полученной информации по сети.

**Общее описание системы**

Система должна уметь работать с изображениями снимков биоптата предстательной железы формата jpeg размера не меньше чем 500 на 500 пикселей формата RGB.Должна выделять и классифицировать ацинусы на фотографии, на основе этого принимать решение о стадии рака.Так же должна уметь сохранять полученный результат в виде : изображение + стадия рака + площадь занимаемая каждым из типов ацинусов в базу знаний SQLLite, так же предпочтительно чтобы программа имела возможность работать в отсутствии установленной на компьютере базы знаний.

**2.2. Требования системы**

Необходимо соблюдать следующие требования:

* **Требования к входным данным**

1. Входные данные должны быть в формате.JPEG
2. Входные данные должны иметь размер не более чем 700 на 700 пикселей
3. Изображения должны быть фотографиями биоптата предстательной железы

* **Требования к выходным данным**

В данной программе выходными данными являются сохраненные в базу знаний резултьтаты работы программы, которые содержат в себе

1. Изображение в формате byteArray
2. Плозадь занимаемая на изображении каждым из типов ацинусов по классификации Глисона.
3. Результирующая оценка стадии по стадии Глисона

* **Требования к интерфейсу программы**

Интерфейс программы должен быть интуитивно понятным. В интерфейсе должно быть изображение из загруженного файла исследования и информация об исследовании. Изображения должны иметь возможность масштабирования. Также пользователю должно быть удобно изменять параметры функций программы.

* **Требования к составу выполняемых функций**

1. Загрузка файла исследования в программу и его отображение в интерфейсе программы
2. Сохранение исследования в базе данных
3. Демонстрация пользователю результатов исследования, а также подсвечивание признаков на основании которых было принято решение и стадии Рака

* **Требования к базе данных**

1. Возможность работать с большим объёмом данных минимум 100000 записей.
2. Требования к ПО
3. Среда разработки (IDE) - Qt\_Creator [http://www.qt.io/ru/];
4. Язык программирования - C++ с использованием фреймворка Qt

* Наличие драйверов SQL под Windows для Qt Creator, поскольку без них невозможно использование базы в системе.
* **Требования к ЭВМ**

1. Система должна корректно запускаться и работать на ЭВМ со следующими требованиями:
2. ОС: Windows 7 64 бит и выше;
3. Процессор: 2-х ядерный с частотой работы от 1.6 ГГц;
4. 2 Гб оперативной памяти;
5. От 2-х ГБ свободного места на жестком диске;
6. Клавиатура, мышь, монитор;
7. Видеокарта с поддержкой разрешения 1920x1080.

* **Требования к пользователю**

К квалификации пользователя, работающего с данной системы, предъявляются следующие требования:

1. Владение компьютером на уровне пользователя;
2. Ознакомление с методикой работы с программой;
3. Преждевременное прочтение руководства пользователя по данной программе.

**3. Проектирование системы**

Цель: спроектировать экспертную систему гистологической диагностики рака предстательной железы, которая будет удовлетворять заявленным требованиям.

Задачи:

1) разработать концептуальную модель системы;

2) спроектировать архитектуру системы.

Данная программа носит данное название, так как в ней содержится набор компонентов для быстрого создания приложений с оконным интерфейсом. А так же содержит ряд архитектурных решений, которые бы позволили в краткие сроки понять структуру программы. Для того чтобы данная программа могла быть промышленным решением, в нее будет необходимо внести ряд доработок.

Предназначение программы в том, что она способна находить на фотографиях биопсии предстательной железы ацинусы, распознавать их, а так же делать выводы о стадии рака, на основе того, сколько процентов площади фотографии занимают ацинусы имеющие те или иные признаки стадий по шкале Глисона.

**3.1. Архитектура системы**

Разработка архитектуры системы включает должна включать в себя:

1) разработку программных интерфейсов между базой данных и компьютерной программой;

2) разработка системы модулей.

Таким образом, разработка архитектуры системы сводится к следующим задачам:

1) разработать модуль, обеспечивающий взаимодействие с базой данных

2) разработать модуль взаимодействия с пользователем

3) Разработать модуль который пишет системные сообщения

4)Разработать модуль фильтрации и классификации

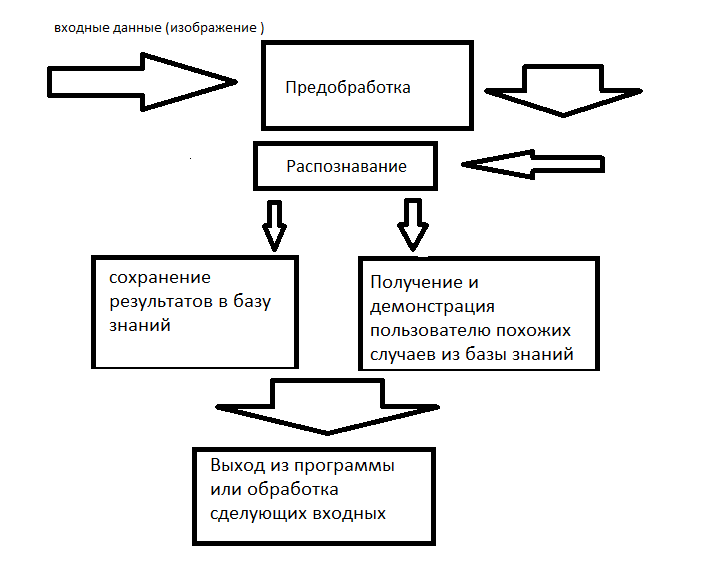
5)Разработать модуль хранения общих данных

Приложение имеет сервисно - ориентированную архитектуру

Так как её принцип работы основан на том, что есть контекст приложения, в котором содержится набор сервисов, которые можно использовать. Создаются и конфигурируются все сервисы в одном единственном месте, некоторые сервисы для универсальности а так же для удобства доработок объявляются через интерфейсы, чтобы в случае доработок, можно было не меняя основной код программы, а только поняв параметры инициализации в контексте сервисов, можно было бы полностью поменять логику программы.

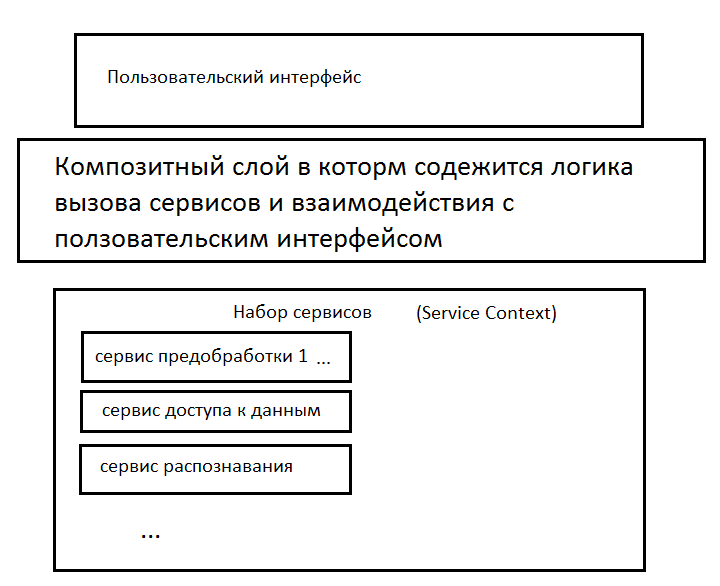
Так же это дает возможность вызывать сервис из любого места программы, без повторной инициализации. То есть, если например в ходе написания программы выясняется, что нужно обратиться к базе данных не на одном экране, а на нескольких, то нам не нужно создавать экземпляр класса обеспечивающего доступ к данным достаточно просто обратиться к контексту приложения и взять оттуда необходимый сервис. Так же это позволяет значительно уменьшить количество переменных принадлежащих, какому-либо объекту, так как все сервисы уже вынесены.

**1) Концептуальная модель**



**Рис.3.** Концептуальная модель системы

**2) Логическая модель**



**Рис.4.** Логическая модель системы

**3.2. Проектирование базы данных**

При проектирование базы данных учитывалось, что требуется решить следующие задачи:

* Использовать базу данных sqlLite
* База знаний должна хранить следующие данные
* эталонные гистограммы вместе с дескрипторами , то есть с метками, которые бы говорили о том к какой стадии рака относится гистограмма.

Так же база знаний должна хранить фотографии пользователя со всеми данными, которые могли бы быть получены в результате обработки программы, а именно: процент площади фотографии занимаемый ацинусами всех типов рака, так же информация о самой стадии (то есть стадия), и сама фотография.

База данных должна поддерживать следующие операции

1) Запрос всех эталонных гистограмм и их описания

2) Запрос на сохранение в базу эталонных гистограмм с описанием

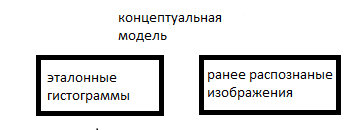
3) Запрос на сохранение фотографий пользователей и все их данный которые можно получить при обработке фотографии с помощью данной программы

4) Запросы на выборку пользовательских фотографий с дополнительными критериями поиска, а именно с фильтрацией по стадии рака, а так же возможность фильтрации всем признакам

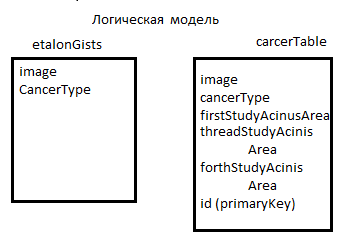
Так же база знаний должна удовлетворять требованиям первых 3 –х нормальных форм

По данным требования поострены модели :

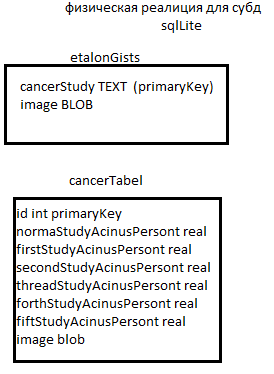
логическая модель, концептуальная модель, физическая реализация



**Рис.5.** Концептуальная модель системы



**Рис.6.** Логическая модель системы



**Рис.7.** Физическая реализация

**Физическая реализация**

**Скрипты создания**

CREATE TABLE carncerTabel(id INT,cancerstady TEXT,firstStadyAcinusPersent real,threadStadyAcinusPersent real,forthStadyAcinusPersent real ,image BLOB

CREATE TABLE gistogrammsTable(cancerstady TEXT ,image BLOB );

**Скрипты наполнения**

INSERT INTO gistogrammsTable(cancerstady,image) VALUES (:cancerstady,:image);

INSERT INTO carncerTabel (id,cancerstady, firstStadyAcinusPersent,threadStadyAcinusPersent,forthStadyAcinusPersent,image)

**4 Физическая реализация**

**4.1. Состав исходных файлов необходимы для сборки программы**

form.ui

main.cpp

mainwindow.cpp

platform.pro

diplom/

startform.ui

startform.h

form.ui

form.h

form.cpp

startform.cpp

servicecontext.h

servicecontext.cpp

resultform.ui

resultform.h

resultform.cpp

imageutils.h

constants.h

recognizers/

sizerecognizer.h

sizerecognizer.cpp

recognizerschain.cpp

recognizerschain.h

formrecognizer.cpp

gistogrammrecognizer.cpp

gistogrammrecognizer.h

formrecognizer.h

acinus.h

acinus.cpp

abstractrecognizer.h

abstractrecognizer.cpp

filters/

mysuperfilter.h

mysuperfilter.cpp

gaussfilter.h

gaussfilter.cpp

filterutils.h

filterutils.cpp

coodrs.h

cords.cpp

blackwhitefilte.h

blackwhitefilte.cpp

binarizer.h

binarizer.cpp

abstractfilter.h

abstractfilter.cpp

database/

vo/

imagedbmodel.h

imagedbmodel.cpp

da/

sqllitedataaccess.h

sqllitedataaccess.cpp

idataaccess.h

idataaccess.cpp

dataaccessfactory.h

dataaccessfactory.cpp

csvdataaccess.h

csvdataaccess.cpp

platform /

widgetwithoutscreen.h

widgetwithoutscreen.cpp

widgetwithoutmodel.h

widgetwithoutmodel.cpp

viewer.h

viewer.cpp

utils.h

utils.cpp

subprocessmodel.h

subprocessmodel.cpp

property.h

property.cpp

minwindow.h

mainwindow.cpp

defoultwidgetModel.h

defoultwidgetmodel.cpp

defoultwidget.h

defoutwidget.cpp

logger/

loggerThread.h

loggerThread.cpp

loggerFactory.h

loggerFactory.cpp

logger.h

logger.cpp

iappender.h

iappender.cpp

hasLogger.h

hasLogger.cpp

conselLoggerAppender.h

conselLoggerAppender.cpp

cache/

cachefactory.h

cachefactory.cpp

cache.h

cache.cpp

**4.2. Основные модули программы:**

1)Модуль логгирования

2) Модуль хранения общих данных

3) Модуль управления переходами между экранами

4) модуль фильтров

5)Модуль доступа к базе данных

6)Модуль распознавателей

**4.2.1. Модуль логгирования**

Данный модуль предназначен для записи служебной информации во время выполнения программы. Например, о ток какие фотографии были загружены или какие были результаты обработки. В основном данная информация нужна разработчику, а не пользователю, так как эта информация чаще всего помогает найти ошибки в уже работающих приложениях.

Но также бывает, что её используют не по прямому назначению.

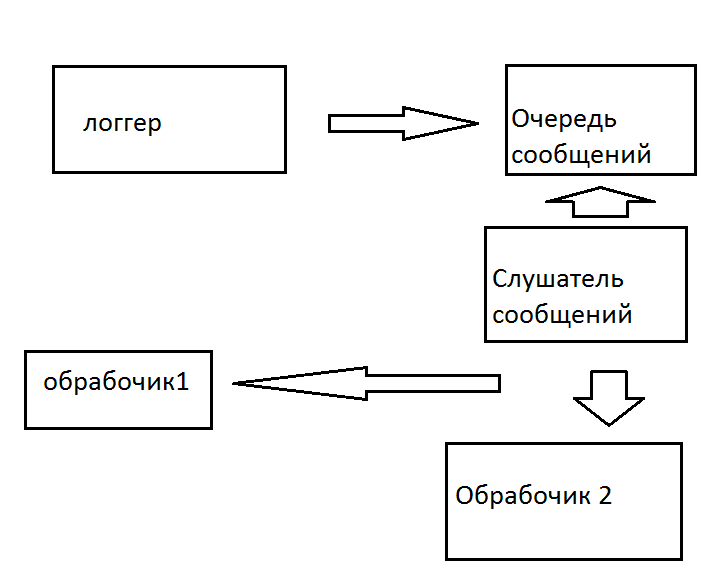
Описание работы:

1) Logger кладет в очередь сообщений сообщение, которые вы хотите записать.

2)Слушатель событий очереди работает в отдельном потоке и слушает события очереди, если в очередь пришло сообщение, то он берет пришедшее в очередь сообщение, и передает ее всем обработчикам, которые у него зарегистрированы.

3) Обработчик сообщения записывает сообщение туда, куда умеет

Схема работы



**Рис.8.**Логическая модель модуля логгирования

Для подключения логгера к своему классу, унаследуйте класс HasLogger, так же в предоставленной мною реализации есть всего один обработчик сообщений, а именно обработчик который умеет писать только в консоль, чего не достаточно, для промышленного решения.

Для добавления собственного обработчика реализуйте интерфейс iappender

И добавьте при инициализации логгера совой обработчик, подобным образом

Logger::Logger()

{

Logger::appenders->append(new ConseleAppneder());

LoggerThread \*t = new LoggerThread();

t->start();

}

Класс LoggerFactory контролирует то, что у вас в программе будет только один экземпляр логера.

**4.2.2. Модуль хранения общих данных**

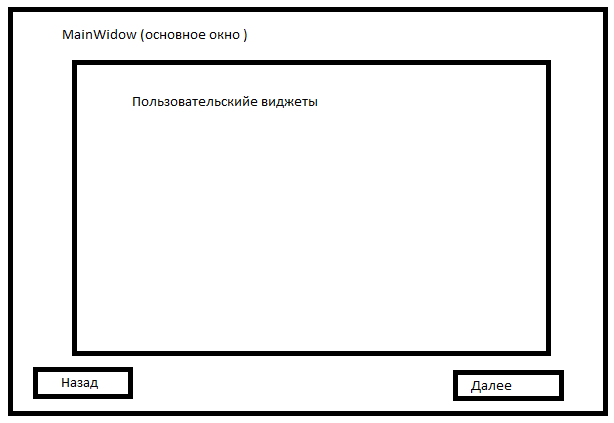
Данный модуль состоит из фабрики, которая предоставляет из себя

Реализация паттерна одиночка(singleton) и дто в виде класса Cache

Для того чтобы расширить функционал данного сервиса, нужно добавить нужное вам поле в класс cache, либо сделать свою реализация класса cache, в таком случае, для этого вам нужно будет будет сделать геттеры и сеттеры виртуальными и сделать своего наследника класса Cache, после этого вам нужно будет подменить в классе CacheFactory при создании экземпляра класса Cache, на свою реализацию, либо объявить Cache интерфейсом.

**4.2.3. Модуль управления переходами между экранами**

Принципиальная схема работы с точки зрения программиста

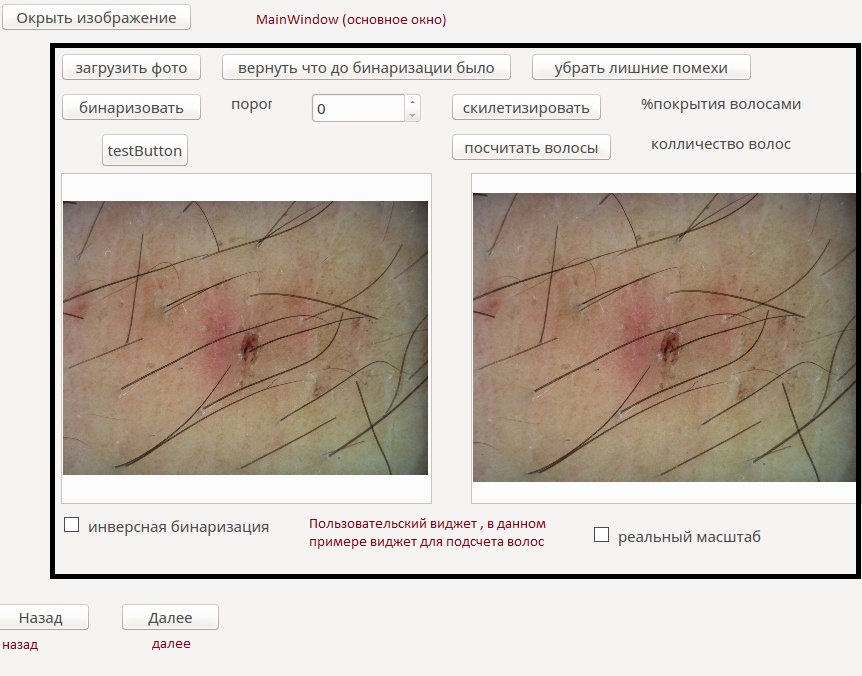


**Рис.9.** Иллюстрация работы модуля переключения между окнами

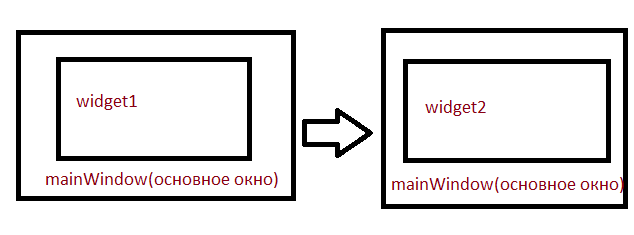
MainWindow (основное окно внешняя обертка вокруг виджетов, которые встраиваются в центр окна)

Пользовательские виджеты, находятся в середине главного окна, внутри них должна быть реализована основная логика работы с пользователем (например обработка изображения )

Пример такого расположения приведен на рисунке 10.

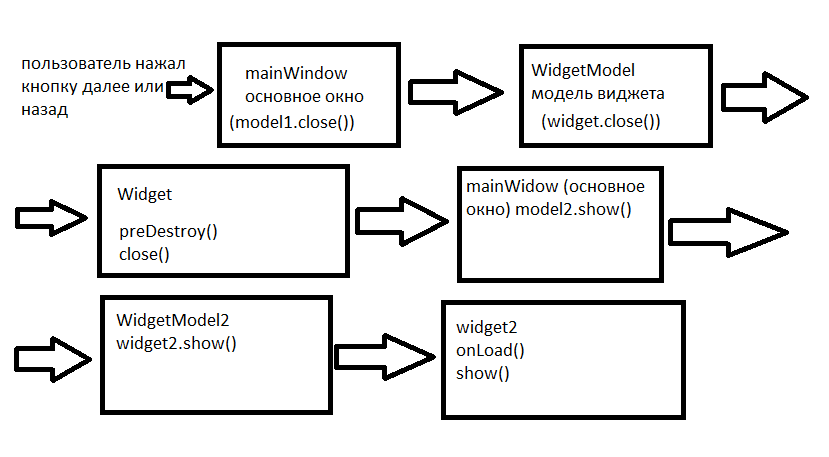


**Рис.10.** Пример работы модуля переключения между виджетамии



**Рис.11.** Схема переключения виджетов

Схема взаимодействия компонентов представлена на рисунке 12



**Рис.12.** Физическая реализация модуля переключения между окнами

DefaultWidget - класс от которого должны быть унаследованны все виджеты используемые в программе содержит в себе экземпляры классов:

* protected: Cache, logger
* private: mainWindow

Класс предназначен для того чтобы разработчики, использующие модуль без лишних действий, уже имели доступ ко всем функция системы

Содержит так же в себе набор функций, которые разработчику необходимо переопределить, если ему нужен дополнительный функционал

virtual void preDestroy(); - функция которая будет вызвана перед закрытием виджета

virtual void onLoad(); -функция которая будет вызвана перед открытием окна

Например, перед открытием окна необходимо взять из хранилища данных актуальное изображение, с которым работает пользователь для этого нужно в виджете который вы создали прописать

void onLoad(){

this.image = cache.getActualImage();

show(image);

}

virtual void onReturnFromSubProcess(); - функция для возврата из под процесса в основной процесс. То есть если ваш виджет был вызван как по процесс после выполнения действий необходимо вызвать эту функцию.

virtual void returnFromSubProcess(); - функция которая будет вызвана при возврате из под процесса в основной процесс.

void unlockNextButton(); - функция для разблокировки кнопки далее

void lockNextButton(); - функция для блокировки кнопки далее

void goToSubProcess(QString processName); - функция для перехода в под процесс.

Для перехода в под процесс необходимо вызвать эту функцию указав названия виджета, который будет являть под процессом.

void nextWidget(); для завершения работы с виджетом , в случае, если подтверждения от пользователя не требуется или же наоборот подтверждение пользователем действий находится внутри виджета, для перехода к следующему процессу нужно вызвать эту функцию

DefaultWidgetModel – модель виджета, служит для работы того чтобы работать с виджетами не напрямую, а через модель, это нужно для того чтобы разработчик имел возможность определить произвольное поведение своего виджета в стандартных ситуациях.

Например, у программе определены действия, которые выполняются перед открытием и закрытием окна, то есть когда происходит закрытие окна, то происходит вызов функции закрытия модели окна (общем случае вызов функции у класса defaultwidgetModel) close()

Внутри которой происходит вызов функции preDestroy.

То есть если вам нужно определить свои действия которые должны происходить при закрытии окна вы можете создать класс унаследованный от defautWidgetModel и в нем переопределить функцию close().

Класс содержит функции

virtual void show(); - функция для открытия вашего виджета, внутри себя она вызовет.

widget.onLoad();

затем widget.show();

virtual void close(); функция для закрытия вашего виджета, внутри себя она вызовет

widget.preDestroy(); затем widget.close();

virtual void onReturnFromSubProcess(); - функция которая будет вызвана при возврате из подпроцесса при вызове фнутри вызывается

widget.onReturnFromSubProcess() затем widget.show();

SubProcessModel класс обёртка для виджета, который будет вызван как под процесс.

Имеет функции show() и close() так как они не виртуальные (то есть переопределить их нельзя ) от и вся их внутренняя логика будет скрыта от разработчиков

MainWindow основной класс программы в котором и содержится вся логики управления пользовательскими виджетами

Имеет поля:

Cache, logger,

Widgets - список моделей виджетов которые будет вызваны по кнопкам далее и назад, от есть

Этот список это по факту последовательность подпрограмм в том виде, в котором её будет видеть пользователь

SubWidgets – map моделей виджетов, которое могут быть вызваны как под процессы

Примечание: один и тот же виджет может быть добавлен и в основной список виджетов (widgets) и всписок под процессов

Содержит функции

MainWindow\* addWidgets(DefaultWidgetModel\* model); Добваление виджета в список виджетов последовательного процесса

MainWindow\* addSupProcess(DefaultWidgetModel\* model); добавление виджета в QMap которая содержит модели для вызова под процессов.

Пример использования (находится в файле main.cpp)

int main(int argc, char \*argv[])

{

QApplication a(argc, argv);

MainWindow w;

ServicesContext::init();

w.addWidgets(newDefaultWidgetModel(newStartForm(w.centralWidget())));

w.addSupProcess(newSubProcessModel(new ResultFrom(w.centralWidget())));

w.show();

return a.exec();

}

**4.2.4. Модуль доступа к базе данных**

Даныный модуль инкапсулирует в себе все взаимодействие с базой данных.

Работа с данным модулем происходит через интерфейс

IDataAccess. Это сделано для того чтобы можно было сделать несколько реализаций данного класса для разных баз данных. Так же дает возможность выбрать способ взаимодействия самостоятельно, например, кому-то может показаться, что база данных на локальной машине это не удобно, и реализовать этот интерфейс через http запросы на получение и манипуляцию данными.

В написанный мной модуль предоставляет две реализации

SqlLiteDataAccess то есть реализация для базы данных sqlLite

и так же предоставляет реализацию для работы в файлом как с базой данных, что впринципе не так плохо в случае если у кого либо, кто собирается дорабатывать и тестировать свои доработки на компьютере, на котором отсутствует база данных sqlLite.

Для того чтобы программа использовала именно sqlLite а не

Реализацию с файлом необходимо :

1 ) скачать и установить систему управления базой данных sql lite

1. Создать в sqllite базу данных
2. В папке с программой есть файл dbproperty.property

В этом файле необходимо указать пусть к созданной в sqllite базе данных в следующем формате

dbName=<path>

В противном случае, если программа не сможет подключиться к базе данных и автоматически переключится на режим работы с файлом, который находится в папке с программой

**4.2.5. Модуль фильтров**

В данном случае я выделил набор фильтров как модуль только по функциональному назначению, в реальности же он не является модулем, а представляет собой набор классов, которые умеют тем или иным образов выполять предварительную обработку изображения

В Данном случае это три фильтра

1. MySyperFilter ();

Данный фильтр умеет искать на бинаризованном изображении

(то есть ему на вход нужно передать QImage у которого у каждого пикселя все три цветовые компоненты либо 255 (белые), либо 0(черные) ) обьекты белого цвета, и возвращать только те обьекты. которые больше заданного числа пикселей (параметр treshold)

vector<QHash<int,PixelType>\*>\* doFilter(QImage &image,int treshold)

Структура ответа сервиса:

Для оптимизации затрат памяти на хранение объектов было принято решение хранить объекты в следующем в виде мапы, в который в качестве ключа указано число, в котором закодирована координата пиксела по x,y. Кодируется она в следующем виде

int key = y\*10000 плюс x ;

данная связка позволяет с помочью одного единственного числа хранить две координаты пиксела, которые бы однозначно его характеризовали.В качестве значения, в мапу передаётся enum Pizel Type, он служебный его структуру разработчика и тем кто собирается его использовать знать не обязательно.

2)Binarizer – тот класс который по хорошему стоило бы переписать если, кто решит доработать программу. Данный метод приводит изображение к формату, который способен распознать класс MySuperFilter, а именно QImage у которого у каждого пикселя все три цветовые компоненты либо 255 (белые)

Данный бинаризатор пороговый, его можно сделать адаптивным, что значительно бы улучшило качество распознавания, а также позволило бы избежать лишнего взаимодействия с пользователем

3)BlackWhiteFilter класс который умеет делать из изображения в трех цветах изображение в оттенках серого, то есть у каждого пискля изображения, его цветовые компоненты равны

4.2.6. Модуль распознавателей

**Интерфейс**

Интерфейс данного модуля это класс RecognizerChain

Который имеет следующую сигнатуру методов

QList<Acinus\*>\*doChain(vector<QHash<int,PixelType>\*>\*objects,QImage\* image);

В данном случае это исходное изображение, а так же результат работы фильтра MySuperFilter, который описан выше.

Данный метод создает из объектов типа QHash<int,PixelType> объекты типа Acinus, которые содержат информацию о том к какому типу ацинуса, по шкале Глисона принадлежит объект.

Затем после того как он создаст объекты, вызываются по очереди все зарегистрированные обработчики. В данном случае это sizerecognizer

Который умеет распознавать ацинусы первой стадии, которые значитально отличаются по размеру от ацинусов других стадий и FormRecognizer, который умеет оценивать форму ацинуса. Каждый обработчик по своему усмотрению может добавлять и удалять признаки стадий глиссона у ацинуса.

Но для некоторых распознавателей требуется это делать строго в определенном порядке, поэтому и класс и называется RecognizerChain то есть цепочка распознавателей. Например в моей реализации два распознавателя один по размеру, второй по форме ацинуса. И если вызвать сначала распознавательно по форме, а потом по размеру, то распознаватель по форме ошибочно добавит признаки 3-й и 4-й стадии для ацинусов подходящей формы, при том что по размеру эти ацинусы должны относиться к третей стадии.

Для промышленной эксплуатации, двух распознавателей недостаточно

Чтобы добавить свой распознаватель, реализуйте интерфейс

AbstractRecognizer

А так же добавьте свою реализацию при инициализации класса RecognizerChain в контексте сервисов

Пример:

Класс Serviceconext.cpp

void ServicesContext::init()

{

FormRecognizer \*circlRecognizer = new FormRecognizer();

SizeRognizer \*sizeRecognizer = new SizeRognizer();

recongnizerChain->recognizers->append(sizeRecognizer);

recongnizerChain->recognizers->append(circlRecognizer);

}

Так же к данному модуля относится класс, который принимает решение о том какую стадию в целом присвоить случаю на фотографии

AttributeRecognizer

Данный класс работает следующим образом:

Он вычисляет площади занимаемые ацинусами в соответствии со шкалой глиссона, и ацинусов какого типа окажется на фотографии больше по площади. Тот тип что окажется доминирующим по площади, будет определяющим при выборе стадии рака.

Gistogrammrecognizer

Данный класс предназначен для анализа гистограммы изображения

Поэтому методика работы следующая :

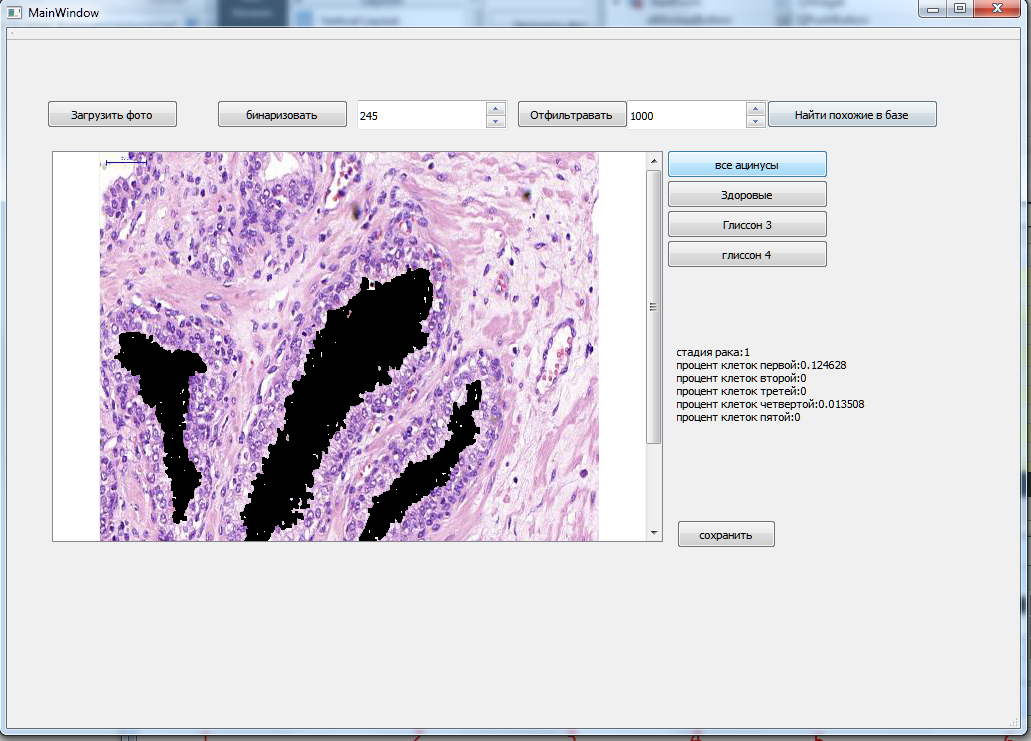
1 ) Строится гистограмма

2) Рассчитываются признаки гистограммы

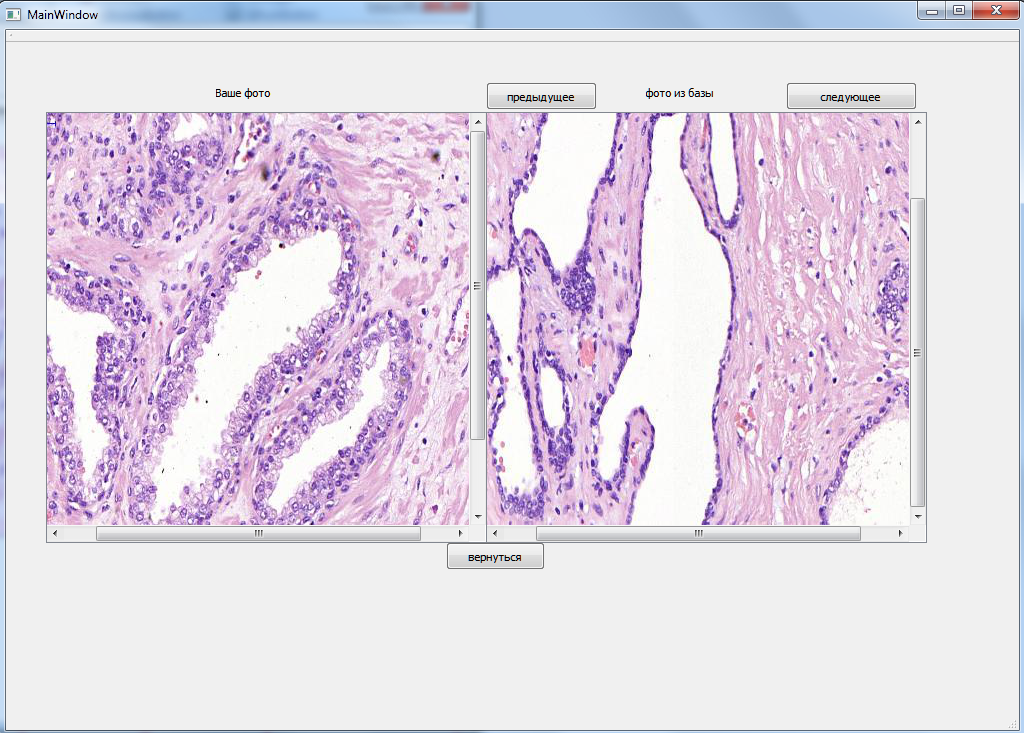
3) Классификатор на основе признаков определяет стадию рака

**4.3. Интерфейс программы**

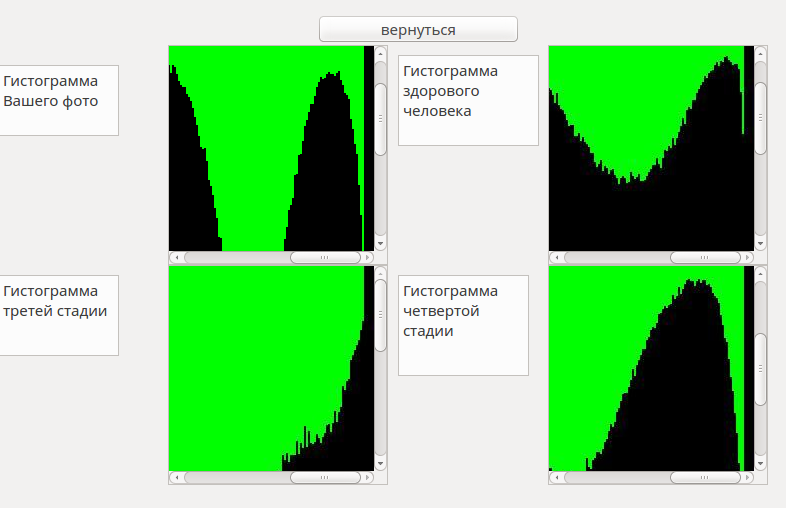
Интерфейс программы состоит из трех экранов, как представлено на изображениях Рис 13 , Рис 14 и 15, подробное инструкция работы пользователя с программой главе методическое обеспечение в пункте руководство пользователя.



**Рис 13.** Интерфейс программы



**Рис 14.** Скрин поиска похожих изображений в базе знаний



**Рис.15** Просмотр эталонных гистограмм и гистограммы пользователя

**5 Методическое обеспечение**

Методическое обеспечение– это планирование, разработка и создание оптимальной системы учебно-методической документации и средств обучения, необходимых для эффективной организации образовательного процесса в рамках времени и содержания, определяемых профессиональной образовательной программой.

**5.1 Состав методического обеспечения**

В состав методического обеспечения входит:

* Руководство пользователя
* Руководство разработчика
* Создание лабораторных работ

**5.2 Руководство пользователя**

Система должна корректно запускаться и работать на ЭВМ со следующими требованиями:

* ОС: Windows 7 64 бит и выше;
* Процессор: 2-х ядерный с частотой работы от 1.6 ГГц;
* 2 Гб оперативной памяти;
* От 2-х ГБ свободного места на жестком диске;
* Клавиатура,
* Мышь,
* Монитор;
* Видеокарта с поддержкой разрешения 1920x1080.

## Требования к пользователю

К квалификации пользователя, работающего с данной системы, предъявляются следующие требования:

1. Владение компьютером на уровне пользователя;
2. Ознакомление с методикой работы с программой;
3. Преждевременное прочтение руководства пользователя по данной программе.

**Методика работы с программой**

1)Настройка программы

Для ознакомительного использования программы, дополнительные настройки не требуются.

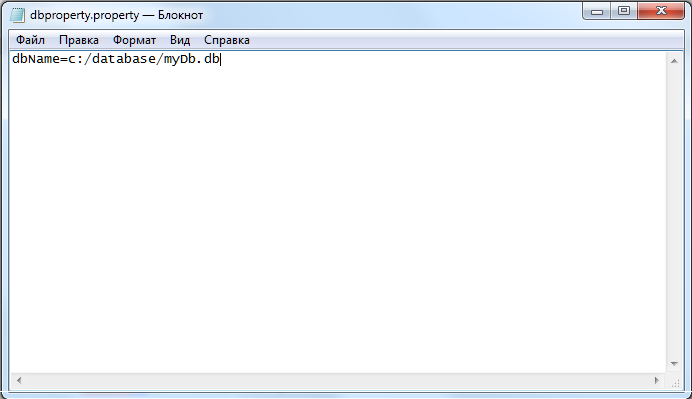
Для использования программы в других целях вам потребуется задать в путь в базе данных

Для этого откройте папку с создаейте файл dbproperty.property

В нем укажите путь к базе данных sqllite в следующем виде

dbName=<путь к базе данных>

Пример

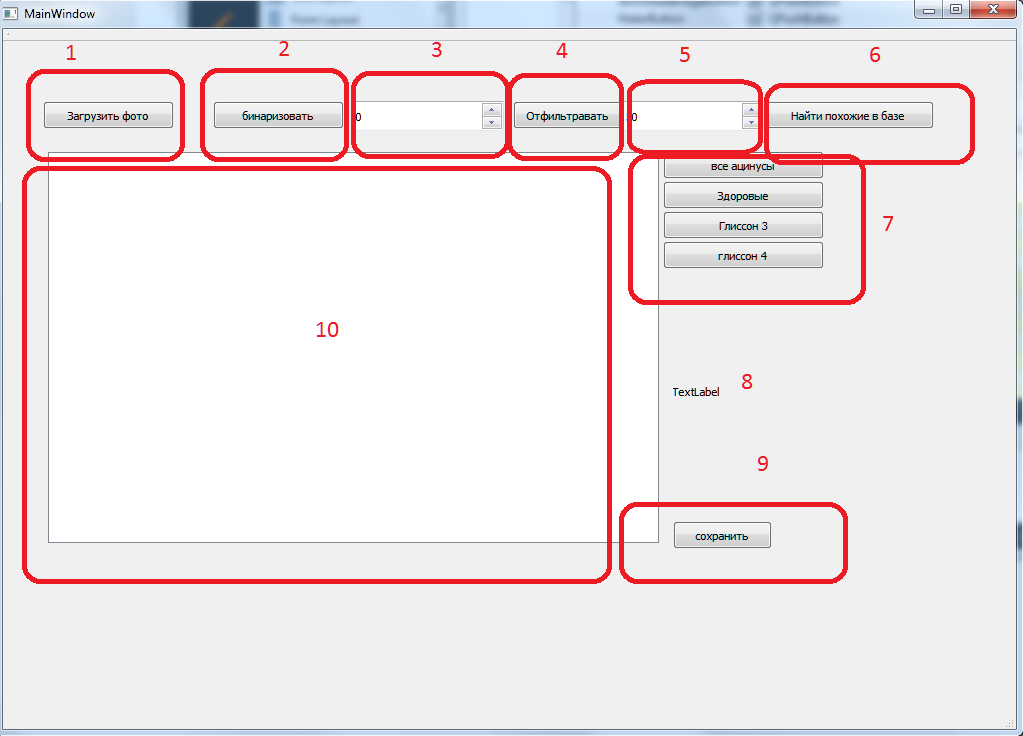


**Рис.16.** Настройка пути к БД

2) Использование программы

Интерфейс программы

В интерфейсе программы имеется два окна с которыми вы будете работать



**Рис.17**. Главное окно программы

Работа с окном :

\*Загрузите в программу изображение, нажав кнопку 1 (загрузить фотографию) \

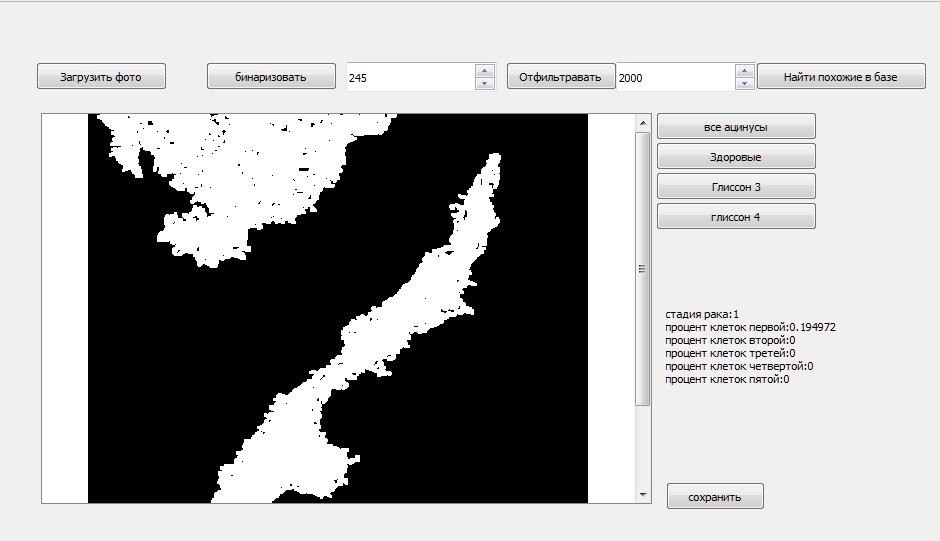
\*Выберете порог бинаризации введя е го в поле 3 , порог бинаризации выбирается из тех соображений , чтобы те области изображения которые не являются ацинусами были как можно меньше особенно важно чтобы помехи , в виде одиночных пикселей не образовывали тесные группы , иначе при фильтрации они могут быть восприняты , как один большой ацинус.

\*Выберите порог минимального количества писклей для объекта , который будет считаться ацинусом вписав его в поле5 и нажмите кнопку 4 (отфильтровать)

Результат вы увидите в поле 8

Примечание ! Не выбирайте порог меньше 100 (в пикселях)

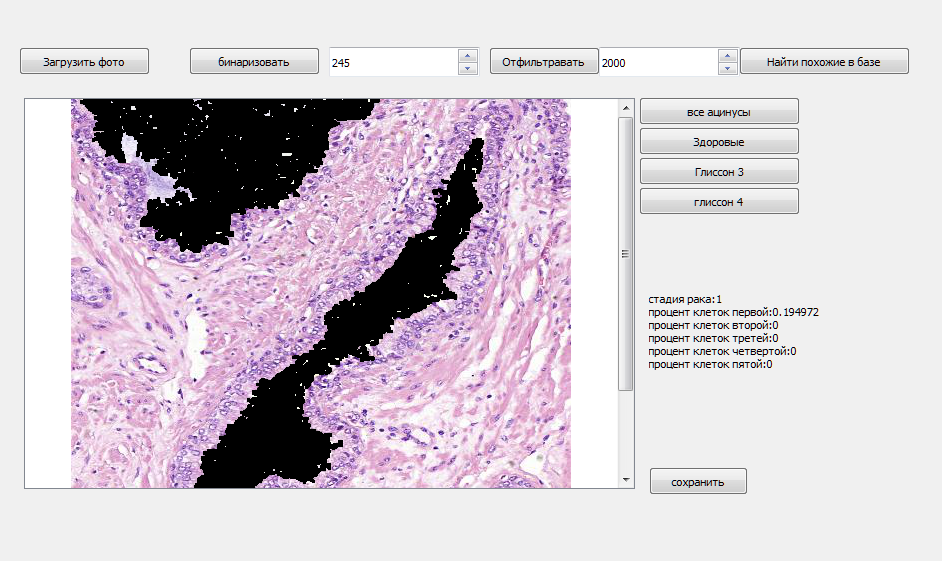
Пример:



**Рис.18**. Результат распознования

Для того чтобы посмотреть как выделены ацинусы на входном изображении воспользуйтесь блоком клавиш 7

Пример : после того как мы получили результат как описано выше , я выбрал показать здоровые ацинусы



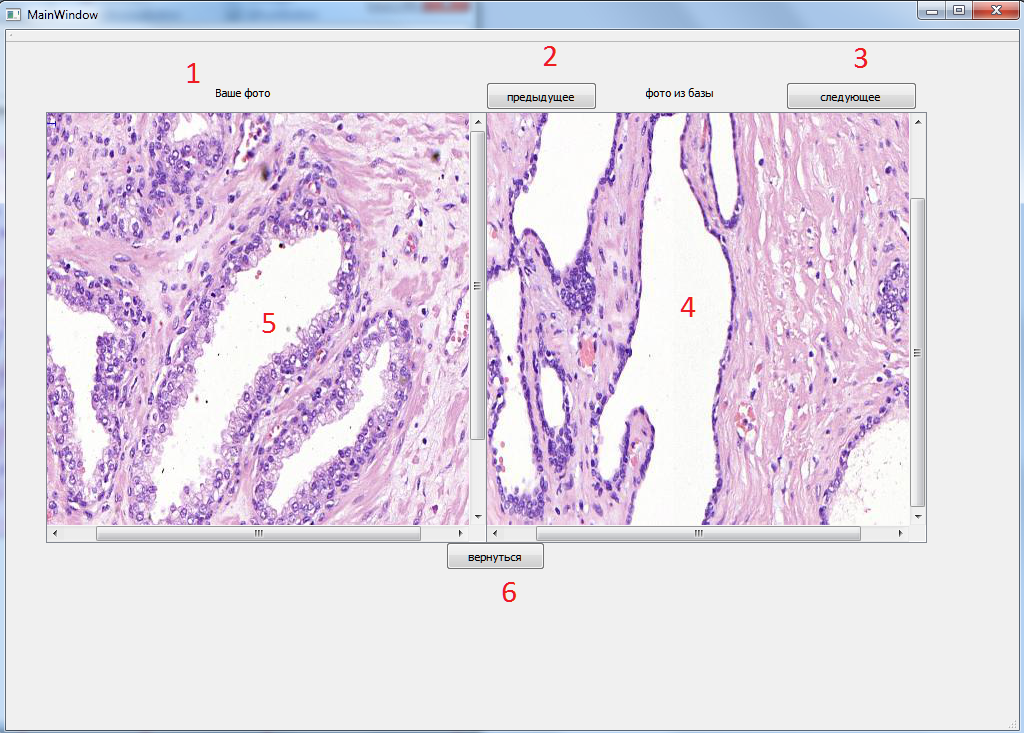
**Рис.18**. Демонстрация отдельных типов ацинусов

\*Далее вы можете либо сохранить изображение, либо поискать похожие результаты в базе данных.

Для того чтобы сохранить результаты, нажмите кнопу 9(сохранить)\

Для того чтобы посмотреть похожие результаты в базе нажмите кнопку 6 (найти похожие в базе)

В этот момент вы перейдете на экран 2



**Рис.19**. Экран сравнения результатов

Ваше изображение будет выведено в поле 5 изображение из базы будет выведено в поле 4.Так как в базе может быть много похожих изображений, вы можете просматрить изображения используя кнопки навигации 2(предыдущее) и 3 (следующее)

Для того чтобы вернуться на первый экран нажмите кнопку 6 (вернуться)\

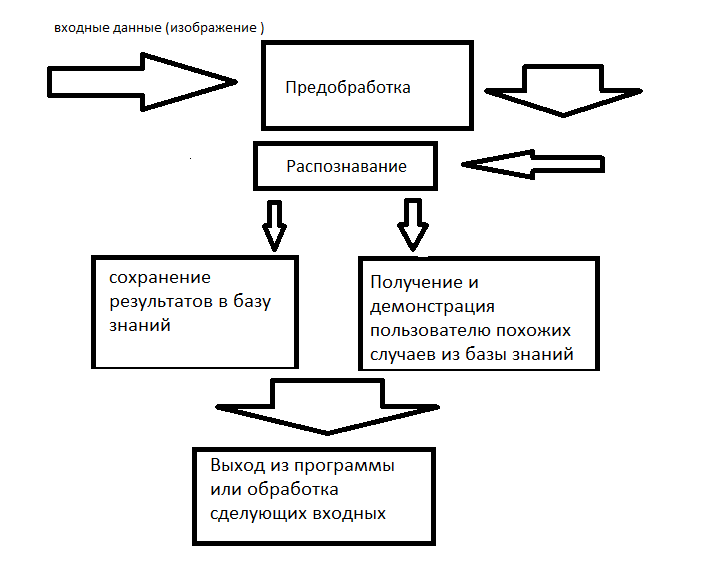
Для того чтобы начать работать со следующим изображением , повторите заново все шаги пункта 2 (работа с программой )

**5.3 Руководство разработчика**

Система диагностики рака предстательной железы

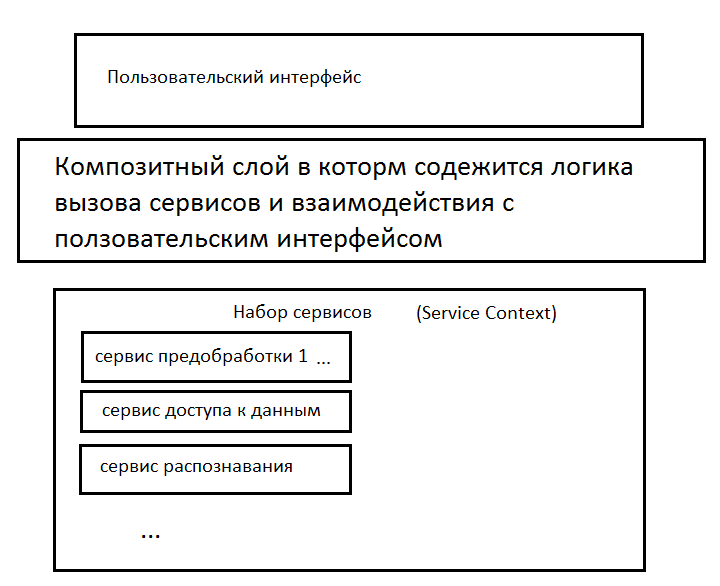
Предназначение программы в том , что она способна находить на фотографиях биопсии предстательной железы ацинусы , распознавать их , а так же делать выводы о стадии рака , на основе того , сколько процентов площади фотографии занимают ацинусы имеющие те или иные признаки стадий по шкале Глисона а так же на основе гистограммы.

**1) Концептуальная схема работы приложения**



**Рис.20**. Концептуальная схема работы

**2) Логическая схема приложения рис 21**



**Рис.20**. Логическая схема работы

Приложение имеет сервисного ориентированную архитектуру

Так как её принцип работы основан на том , что есть есть контекст приложения , в которм содержится набор сервисов , которые можно использовать . Создаются и конфигурируются все сервисы в одном единственном месте , некоторые сервисы для универсальности а так же для удобства доработок объявляются через интерфейсы , чтобы в случае доработок , можно было не меняя основной код программы , а только поняв параметры инициализации в контексте сервисов , можно былобы полностью поменять логику программы.

Так же это дает возможность вызывать сервис из любого места программы , без повторной инициализации .То есть , если например в ходе написания программы выясняется , что нужно обратиться к базе данных не на одном экране , а на нескольких, то нам не нужно создавать экземпляр класса обеспечивающего доступ к данным достаточно просто обратиться к контексту приложения и взять оттуда необходимый сервис . Так же это позволяет значительно уменьшить количество переменных принадлежащих, какому либо объекту, так как все сервисы уже вынесены.

**Состав исходных файлов необходимы для сборки программы**

form.ui

main.cpp

mainwindow.cpp

platform.pro

diplom/

startform.ui

startform.h

startform.cpp

servicecontext.h

servicecontext.cpp

resultform.ui

resultform.h

resultform.cpp

imageutils.h

constants.h

recognizers/

sizerecognizer.h

sizerecognizer.cpp

recognizerschain.cpp

recognizerschain.h

formrecognizer.cpp

formrecognizer.h

acinus.h

acinus.cpp

abstractrecognizer.h

abstractrecognizer.cpp

filters/

mysuperfilter.h

mysuperfilter.cpp

gaussfilter.h

gaussfilter.cpp

filterutils.h

filterutils.cpp

coodrs.h

cords.cpp

blackwhitefilte.h

blackwhitefilte.cpp

binarizer.h

binarizer.cpp

abstractfilter.h

abstractfilter.cpp

database/

vo/

imagedbmodel.h

imagedbmodel.cpp

da/

sqllitedataaccess.h

sqllitedataaccess.cpp

idataaccess.h

idataaccess.cpp

dataaccessfactory.h

dataaccessfactory.cpp

csvdataaccess.h

csvdataaccess.cpp

platform /

widgetwithoutscreen.h

widgetwithoutscreen.cpp

widgetwithoutmodel.h

widgetwithoutmodel.cpp

viewer.h

viewer.cpp

utils.h

utils.cpp

subprocessmodel.h

subprocessmodel.cpp

property.h

property.cpp

minwindow.h

mainwindow.cpp

defoultwidgetModel.h

defoultwidgetmodel.cpp

defoultwidget.h

defoutwidget.cpp

logger/

loggerThread.h

loggerThread.cpp

loggerFactory.h

loggerFactory.cpp

logger.h

logger.cpp

iappender.h

iappender.cpp

hasLogger.h

hasLogger.cpp

conselLoggerAppender.h

conselLoggerAppender.cpp

cache/

cachefactory.h

cachefactory.cpp

cache.h

cache.cpp

Основые модули программы :

1)Модуль логгирования

2) Модуль хранения общих данных

3) Модуль управления переходами между экранами

4) модуль фильтров

5)Модуль доступа к базе данных

6)Модуль распознавателей

**Настройка проекта**

Для того чтобы запустить проект вам потребуется qt creator версии 4.8 и более поздней, операционная система значения не имеет, так как если у вас qt creator с полным комплектом (вместе с компилятором ) то он уже уже умеет работать с операционной системой, а значит и проект вы сможете там собрать и запустить.

Никаких сторонних библиотек в проекте нет , только библиотеки входящие в состав qt creator( в полный пакет они точно входят )

Среда разработки (IDE) - Qt\_Creator [http://www.qt.io/ru/];

Так же рекомендуемые системные требования

Процессор intel i5 или более новый

6гб оперативной памяти или более

Монитор с разрешением 600\*1000 или более

(данное требование связано с тем что в данный момент размеры создаваемых пользовательских окон имеют такой размер)

Так же требования для входных данных это изображения формата jpg , с другими форматами не будет работать сохранение данных файл вместо базы данных .

Так же обязательно нужно будет установить sql lite

Версия sqlLite должна быть 3 или более поздняя http://www.sqliteexpert.com/download.html

Так же рекомендуется установить sqlitmen это графическая оболочка для удобства использования sqllite http://ru.softoware.org/database-software/download-sqliteman-for-windows.html

**Модуль логгирования :**

Данный модуль предназначен для записи служебной информации во время выполнения программы. Например о ток какие фотографии были загружены или какие были результаты обработки. В основном данная информация нужна разработчику а не пользователю, так как эта информация чаще всего помогает найти ошибки в уже работающих приложениях .

Но так же бывает что её используют не по прямому назначению.

Описание работы:

1 ) Logger кладет в очередь сообщений сообщение , которые вы хотите записать.

2)Слушатель событий очереди в работает в отдельном потоке и слушает события очереди, если в очередь пришло сообщение, то он берет пришедшее в очередь сообщение, и передает ее всем обработчикам, которые у него зарегистрированы.

3) Обработчик сообщения записывает сообщение туда, куда умеет

Схема работы. 2.

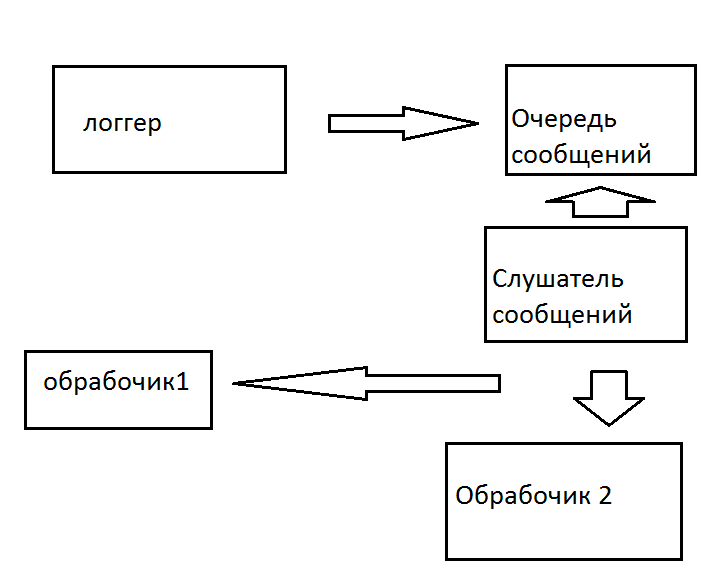


Схема 2 схема работы модуля логгирования.

Для подключения логгера к своему классу, просто унаследуйте класс HasLogger, так же в предоставленной мною реализации есть всего один обработчик сообщений а именно обработчик который умеет писать только в косоль, чего не достаточно, для промышленного решения.

Для добавления собственного обработчика реализуйте интерфей iappender

И добавьте при инициализации логгера совой обработчик, подобным образом.

Logger::Logger()

{

Logger::appenders->append(new ConseleAppneder());

LoggerThread \*t = new LoggerThread();

t->start();

}

Класс LoggerFactory конролирует то , что у вас в программе будет только один экземпляр логера .

**Модуль общих данных**

Данный модуль состоит из фабрики , которая предоставляет из себя

Реализация паттера одиначка(singleton) и дто в виде класса Cache

Для того чтобы расширить функционал данного сервиса, нужно просто добавить нужное вам поле в класс cache , либо сделать свою реализация класса cache, в таком случае, для этого вам нужно будет будет сделать геттеры и сеттреы виртуальными и сделать своего наследника класса Cache, после этого вам нужно будет подменить в классе CacheFactory при создании экземпляра класса Cache, на свою реализацию, либо объявить Cache интерфейсом.

Пример:

1. Предположим есть класс cache

Cache.h

class Cache {

public :

getA();

getB();

private:

int a ;

int b ;

}

Cache.cpp

Cache::Cache(){}

int Cache::getA(){return a;}

Int Cache::getB(){return b;}

Который создается в классе CacheFactory следующим образом

Cache\* CacheFactory::getInstance(){

if (instance == NULL || instance ==0 ){

instance = new Cache();

}

return instance;

}

1. Добавление своей реализации путем переделывания Cache в интерфейс

Для этого модифицируем класс Cache

Cache.h

Abstract class Cache {

public :

virtual int getA() =0 ;

virtual int getB() = 0 ;

}

Файл Cache.cpp удаляем ну или как минимум удаляем из него реализации методов.

Важно,что вся все методы должны сохранить обратную совместимость. То есть нельзя удалять методы или менять сигнатуру, можно только добавлять новые.

Делаем собственную реализацию

Myrealization.h

class MyRealization : public Cache {

public :

virtual int getA();

virtual int getB() ;

}

Myrealizatin.cpp

MyRealization:: MyRealization (){}

int MyRealization::getA(){return 22;}

Int MyRealization::getB(){return 11;}

}

1. Меняем объявление в классе LoggerFatory

Cache\* CacheFactory::getInstance(){

if (instance == NULL || instance ==0 ){

instance = new MyRealization ();

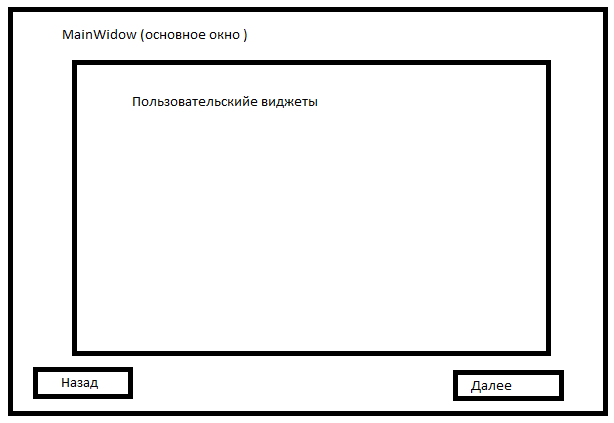
}

return instance;

}

**Модуль перехода между экранами**

Принципиальная схема работы с точки зрения программиста Рис.4

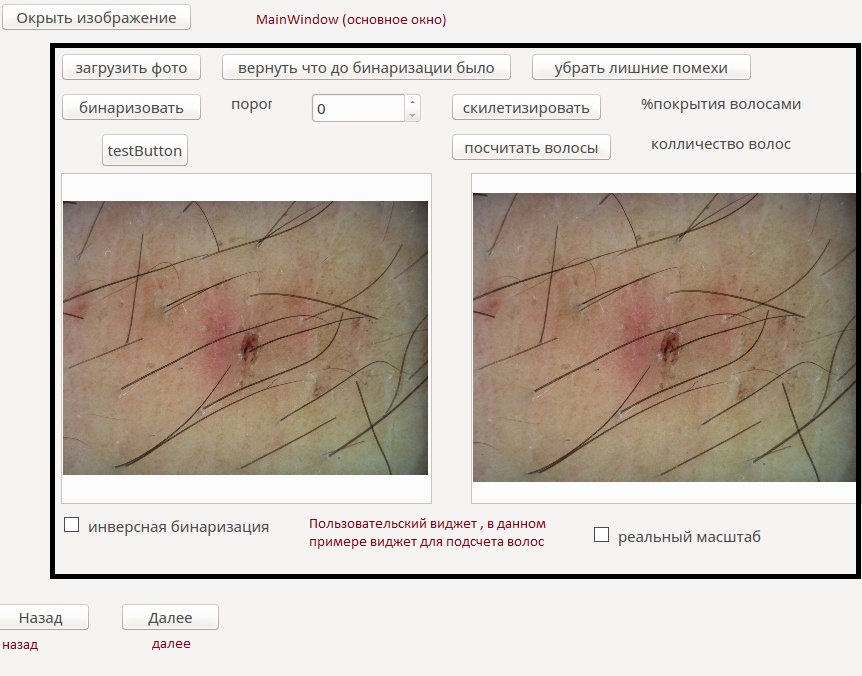


**Рис. 21**. Иллюстрация работы модуля перехода между экранами.

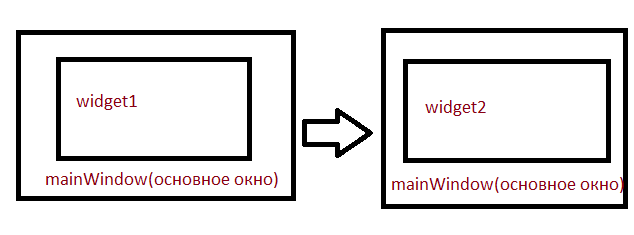
MainWindow (основное окно внешняя обертка вокруг виджетов, которые встраиваются в центр окна)

Пользовательские виджеты, находятся в середине главного окна, внутри них должна быть реализована основная логика работы с пользователем (например обработка изображения)

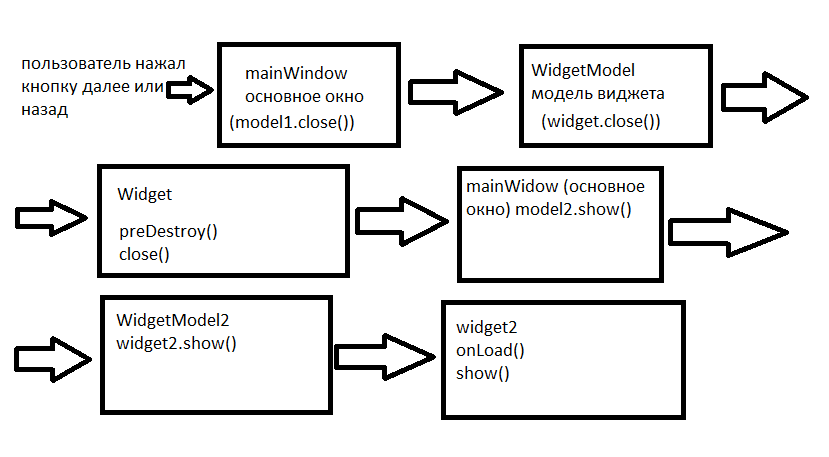
Пример такого расположения приведен на рисунке 22.



**Рис. 21**. Пример работы модуля перехода между экранами



**Рис. 22**. Схема переключения виджетов



**Рис.23**. Схема взаимодействия компонентов

Описание основных классов и методики работы с ними

) DefaultWidget - класс от которого должны быть унаследованы все виджеты используемые в программе содержит в себе экземпляры классов:

protected: Cache, logger

private: mainWindow

класс предназначен для того чтобы разработчики, использующие модуль без лишних действий, уже имели доступ ко всем функция системы

содержит так же в себе набор функций, которые разработчику необходимо переопределить, если ему нужен дополнительный функционал

virtual void preDestroy(); - функция которая будет вызвана перед закрытием виджета

virtual void onLoad(); -функция которая будет вызвана перед открытием окна

например, перед открытием окна необходимо взять из хранилища данных актуальное изображение, с которым работает пользователь для этого нужно в виджете который вы создали прописать

void onLoad(){

this.image = cache.getActualImage();

show(image);

}

virtual void onReturnFromSubProcess(); - функция для возврата из под процесса в основной процесс . То есть если ваш виджет был вызван как по процесс после выполнения действий необходимо вызвать эту функцию.

virtual void returnFromSubProcess(); - функция которая будет вызвана при возврате из под процесса в основной процесс .

void unlockNextButton(); - функция для разблокировки кнопки далее

void lockNextButton(); - функция для блокировки кнопки далее

void goToSubProcess(QString processName); - функция для перехода в под процесс.

для перехода в под процесс необходимо вызвать эту функцию указав названия виджета, который будет являть под процессом.

void nextWidget(); для завершения работы с виджетом, в случае, если подтверждения от пользователя не требуется или же наоборот подтверждение пользователем действий находится внутри виджета, для перехода к следующему процессу нужно вызвать эту функцию.

6)DefaultWidgetModel – модель виджета, служит для работы того чтобы работать с виджетами не напрямую а через модель, это нужно для того чтобы разработчик имел возможность определить произвольное поведение своего вижета в стандартных ситуациях.

Например у программе определены действия которые выполняются перед открытием и закрытием окна, то есть когда происходит закрытие окна , то происходит вызов функции закрытия модели окна (общем случае вызов функции у класса defaultwidgetModel) close()

Внутри которой происходит вызов функции preDestroy.

То есть если вам нужно определить свои действия которые должны происходить при закрытии окна вы можете создать класс унаследованный от defautWidgetModel и в нем переопределить функцию close().

Класс содержит функции

virtual void show(); - функция для открытия вашего виджета , внутри себя она вызовет.

widget.onLoad();

затем widget.show();

virtual void close(); функция для закрытия вашего виджета , внутри себя она вызовет

widget.preDestroy(); затем widget.close();

virtual void onReturnFromSubProcess(); - функция которая будет вызвана при возврате из подпроцесса при вызове фнутри вызывается

widget.onReturnFromSubProcess() затем widget.show();

7) SubProcessModel класс обёртка для виджета, который будет вызван как под процесс.

Имеет функции show() и close() так как они не виртуальные (то есть переопределить их нельзя ) от и вся их внутренняя логика будет скрыта от разработчиков.

8)MainWindow основной класс программы в котором и содержится вся логики управления пользовательскими виджетами.

Имеет поля:

Cache, logger,

Widgets - список моделей виджетов которые будет вызваны по кнопкам далее и назад, от есть этот список это по факту последовательность подпрограмм в том виде, в котором её будет видеть пользователь.

SubWidgets – map моделей виджетов, которое могут быть вызваны как под процессы.

Примечание: один и тот же виджет может быть добавлен и в основной список виджетов (widgets) и всписок под процессов.

Содержит функции:

MainWindow\* addWidgets(DefaultWidgetModel\* model); Добавление виджета в список виджетов последовательного процесса

MainWindow\* addSupProcess(DefaultWidgetModel\* model);добавление виджета в мапу которая содержи модели для вызова под процессов.

Пример использования (находится в файле main.cpp)

int main(int argc, char \*argv[])

{

QApplication a(argc, argv);

MainWindow w;

ServicesContext::init();

w.addWidgets(new DefaultWidgetModel(new StartForm(w.centralWidget())));

w.addSupProcess(new SubProcessModel(new ResultFrom(w.centralWidget())));

w.show();

return a.exec();

}

**Модуль фильтров**

В данном случае я выделил набор фильтров как модуль только по функциональному назначению , в реальности же он не является модулем , а представляет собой набор классов , которые умеют тем или иным образов выполять предварительную обработку изображения

В Данном случае это три фильтра

1. MySyperFilter ();

Данный фильтр умеет искать на бинаризованном изображении

(то есть ему на вход нужно передать QImage у которого у каждого пикселя все три цветовые компоненты либо 255 (белые), либо 0(черные) ) обьекты белого цвета , и возвращать только те обьекты . которые больше хаданного числа пикселей (параметр treshold)

vector<QHash<int,PixelType>\*>\* doFilter(QImage &image,int treshold)

Структура ответа сервиса:

Для оптимизации затрат памяти на хранение обьктов было принято решение хранить обьекты в следующем в види мапы , в который в качестве ключа указано число, в котором закодирована координата пиксела по x ,y. Кодируется она в следующем виде

int key = y\*10000 плюс x ;

данная связкаа позволяет с помошью одного единственного числа хранить две координаты пиксела , которые бы одназначно его характиризовали .В качестве значения, в мапу передается enum Pizel Type, он служебный его структуру разработчика и тем кто собирается его использовать знать не обязательно.

2)Binarizer – тот класс который по хорошему стоило бы переписать если , кто решит доработать программу . Данный метод приводит изображение к формату, который способен распознать класс MySuperFilter , а именно QImage у которого у каждого пикселя все три цветовые компоненты либо 255 (белые)

Данный бинаризатор пороговый , его можно сделать адаптивным , что значительно бы улучшило качество распознавания , а так же позволило бы избежать лишнего взаимодействия с пользователем

3)BlackWhiteFilter класс который умеет делать из изображения в трех цветах изображение в оттенках серого , то есть у каждого пискла изображения, его цветовые компоненты равны

**Модуль доступа к базе данных**

Даныный модуль инкапсулирует в себе все взаимодействие с базой данных .

Работа с данным модулем происходит через интерфейс

IDataAccess . Это сделано для того чтобы можно было сделать несколько реализаций данного класса для разных баз данных . Так же дает возможность выбрать способ взаимодействия самостоятельно , напрмер , кому-то может показаться , что база данных на локальной машине это не удобно , и реализовать этот интерфейс через http запросы на получение и манипуляцию данными .

В написанный мной модуль предоставляет две реализации

SqlLiteDataAccess то есть реализация для базы данных sqlLite

и так же предоставляет реализацию для работы в файлом как с базой данных.

Для того чтобы программа использовала именно sqlLite реализацию необходимо :

1 ) скачать и установить систему управления базой данных sql lite

1. Создать в sqllite базу данных
2. В папке с программой есть файл dbproperty .property

В этом файле необходимо указать пусть к созданной в sqllite базе данных в следующем формате

dbName=<path>

В противном случае, если программа не сможет подключиться к базе данных и автоматически переключится на режим работы в файлом , который она создаст в папке с программой

**Модуль фильтров**

Api данного модуля это класс RecognizerChain

Который имеет следующую сигнатуру методов

QList<Acinus\*>\* doChain(vector<QHash<int,PixelType>\*> \*objects,QImage\* image);

В данном случае это исходное изображение , а так же результат работы фильтра MySuperFilter , который описан выше .

Данный метод создает из объектов типа QHash<int,PixelType> объекты типа Acinus , которые содержат информацию о том к какому типу ацинуса , по шкале Глисона принадлежит объект .

Затем после того как он создаст объекты , вызываются по очереди все зарегистрированные обработчики . В данном случае это sizerecognizer

Который умеет распознавать ацинусы первой стадии , которые значитально отличаются по размеру от ацинусов других стадий и FormRecognizer , который умеет оценивать форму ацинуса . Каждый обработчик по своему усмотрению может добавлять и удалять признаки стадий Глисона у ацинуса.

Но для некоторых распознавателей требуется это делать строго в определенном порядке , поэтому и класс и называется RecognizerChain то есть цепочка распознавателей . Например в моей реализации два распознавателя один по размеру , второй по форме ацинуса . И если вызвать сначала распознавательно по форме , а потом по размеру, то распознаватель по форме ошибочно добавит признаки 3-й и 4-й стадии для ацинусов подходящей формы , при том что по размеру эти ацинусы должны относиться к третей стадии .

Для промышленной эксплуатации, двух распознавателей недостаточно

Чтобы добавить свой распознаватель, реализуйте интерфейс

AbstractRecognizer

А так же добавьте свою реализацию при инициализации класса RecognizerChain в контексте сервисов.

Пример :

Класс Serviceconext.cpp

void ServicesContext::init()

{

FormRecognizer \*circlRecognizer = new FormRecognizer();

SizeRognizer \*sizeRecognizer = new SizeRognizer();

recongnizerChain->recognizers->append(sizeRecognizer);

recongnizerChain->recognizers->append(circlRecognizer);

}

Так же к данному модуля относится класс который принимает решение о том какую стадию в целом присвоить случаю на фотографии

AttributeRecognizer

Данный класс работает следующим образом:

Он вычисляет площади занимаемые ацинусами в соответствии со шкалой Глисона, и ацинусов какого типа окажется на фотографии больше по площади. Тот тип что окажется доминирующим по площади, будет определяющим при выборе стадии рака.

**5.4 Разработка лабораторных работ**

Были разработаны лабораторные работы для студентов медицинских специальностей и для студентов IT специальностей.

**Лабораторная работа для студентов медицинских специальностей.**

Лабораторная работа для студентов медицинских специальностей включает в себя :

\*описание простых алгоритмов обработки изображений

\*Методика работы программы

\*Иллюстрация работы с этими алгоритмами

\* Требования к отчету

При разработке лабораторной был сделан упор на междисциплинарное исследование, то есть цель лабораторной работы, дать понимание о том как происходит обработка изображения людям, которые далеки от ит специальностей. Поэтому в данной лабораторная работа предлагает студентам ознакомиться с двумя простыми способами распознавания , а именно с пороговой бинаризаций и с анализом гистограмм .

**Лабораторная работа для студентов ит специальностей**

Лабораторная работа для студентов медицинских специальностей включает в себя :

\*описание простых алгоритмов обработки изображений

\*Методика работы программы

\*Иллюстрация работы с этими алгоритмами

\* Требования к отчету

Цель лабораторной работы познакомить студентов младших курсов с принципами построения многомодульных программных систем. В данной работе студенты должны будут ознакомиться с тем что такое микросервисная архитектура. Как и почему должно быть реализовано логгирование (запись информации о работ программы ) , что такое одиночка (singleton) как и где его можно применить. Какие есть способы реализации бизнес логики на основе разработанного мною модуля для управления состояниями экрана. В качестве практической работы студентам предлагается реализовать добавить в программу экран, при этом они должны буду. Так же студентам предлагается реализовать не сложный алгоритм построения гистограмм.

Лабораторные работы представлены в приложении 4 «лабораторная работа , для ит специалистов» и в приложении 5 «лабораторная работа для студентов медицинских специальностей».

**6 Экспериментальные исследования**

**6.1.Тестирование работоспособности программы**

**6.1.1.Цель и проведение тестирования.**

Цель тестирования: Проверка работоспособности программы и соответствия поведения программы ожидаемому

Разработка плана проведения тестирования

В качестве плана проведения тестирования предложен следующий подход:

Каждое требование к программе должно быть проверено в случае правильных и возможных неправильных действиях пользователя. Система должна уметь обрабатывать ситуации неправильных входных данных.

Успехом проведенного тестирования будет считаться сохранение работоспособности программы и выполнение всех требований, поставленных к системе.

Тестирование системы производилось при следующих характеристиках компьютера:

Процессор: Intel(R) Core(TM) i5-4200U @ 1.8 GHz

Оперативная память: 6 GB

Видеокарта: Intel Integrated 4000M

Операционная система: Windows 7

Тестирование программы будет проводиться по описанным в данном плане сценариям работы с программой.

Методика тестирования:

1. Загрузка корректного изображения фомрата jpeg размером 500 \*500. Ожидаемый результат - отображение информации в программе
2. Бинаризация изображения

Обижаемый результат –> изображение бинаризовано.

Проверка: будет выведено изображение в монохромном формате

1. Фильтрация

Ожидаемый результат ->

Пункт 1 вывод на экран объектов, которые мы считаем ацинусами, так же вывод программы о том, какая стадия рака и информация о том сколько по площади занимает ацинусы того или иного типа.

Проверка: на экране отображены только ацинусы, а так же искомая информация.

Пункт 2: При нажатии клавиш с выбором типа отображения выводится исходное изображение, на котором выделены только ацинусы того типа, который выбран.

1. Сохранение

Ожидаемый результат -> результат сохранен в базе - Проверка ->

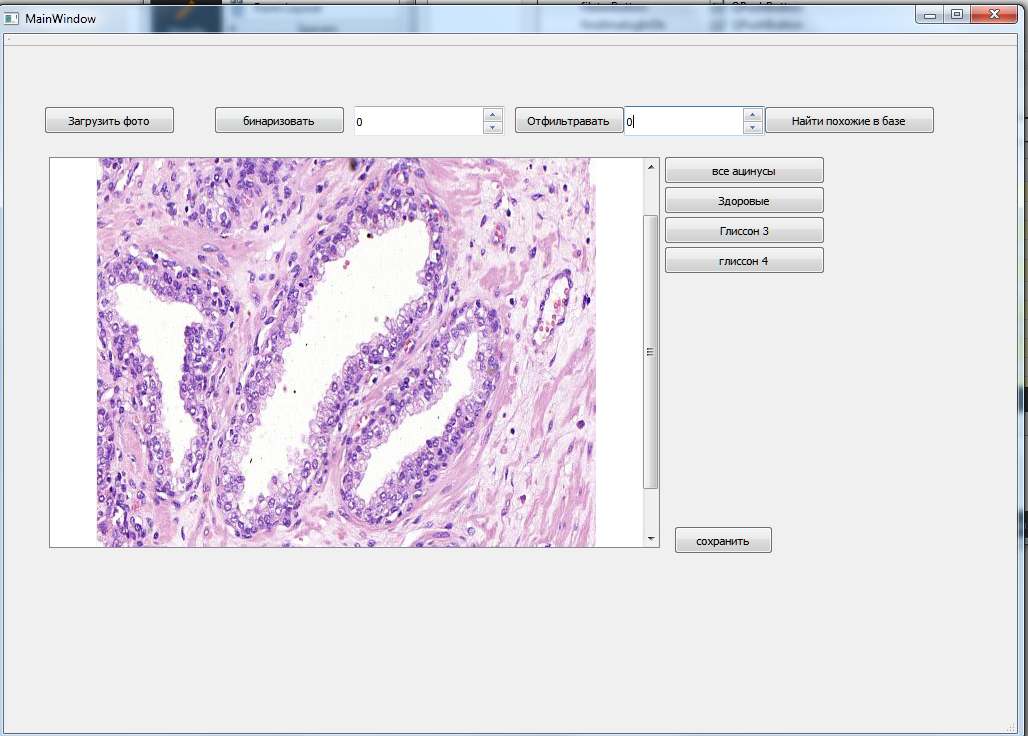
Первый способ: Зайти, а базу данных с помочью программы sqllitman и проверить наличие записи в таблице

Второй способ: после того как мы сохранили изображения, повторно выполняем действия по загрузке изображения в базу, бинаризации и фильтрации, затем нажимаем кнопку “посмотреть похожие” и в таком случае мы должны будем увидеть там наше изображение в списке похожих изображений из базы знаний.

5. Построение гистограммы, догрузка в базу данных эталонных гистограмм

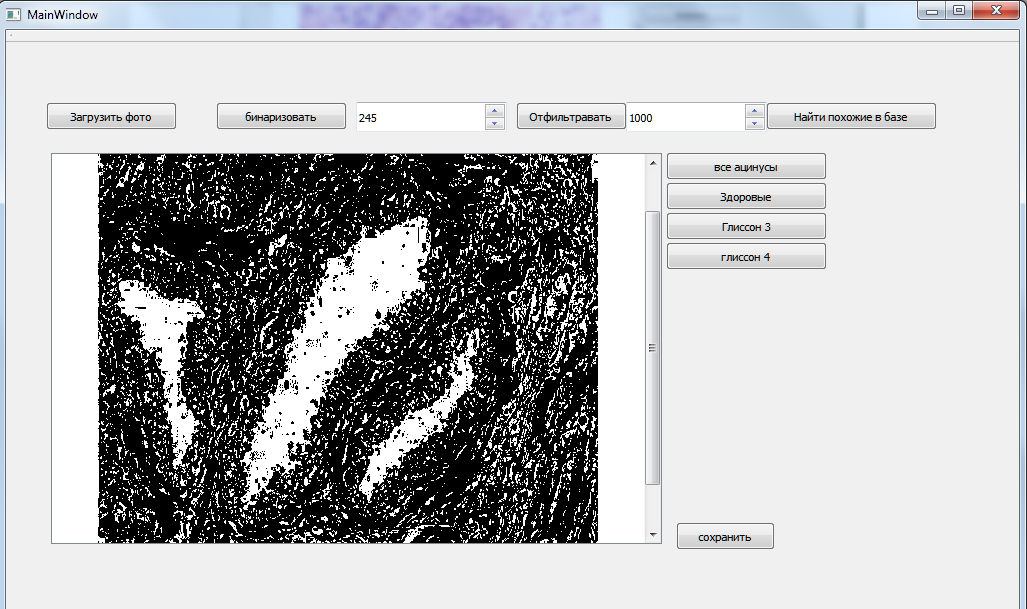
**6.1.2. Результаты тестирования**

1. Загружаем изображение



**Рис.24.** Загрузка изображения

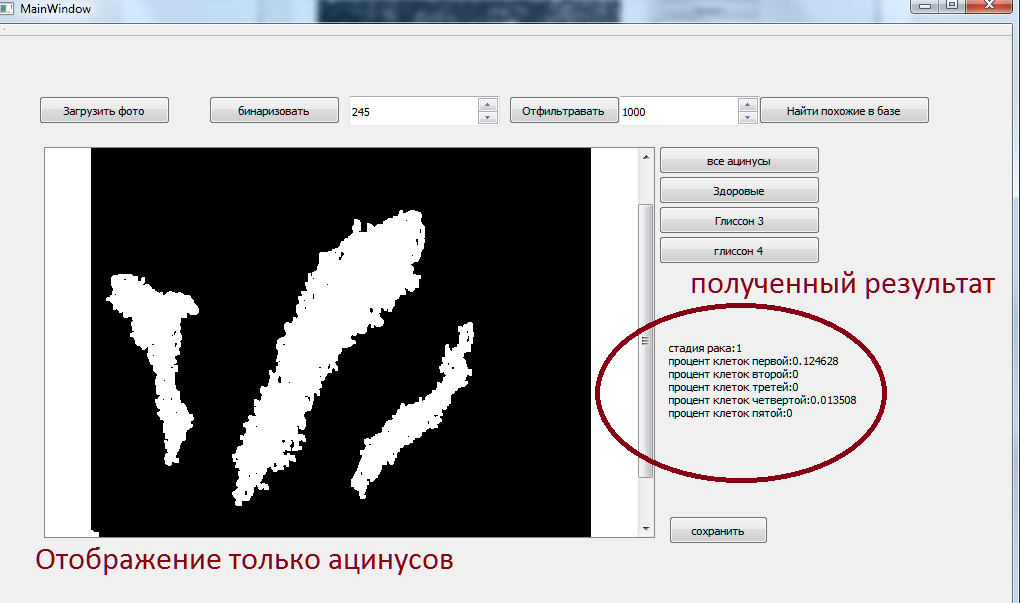
2. Происходит бинаризация



**Рис.25.** Бинаризация

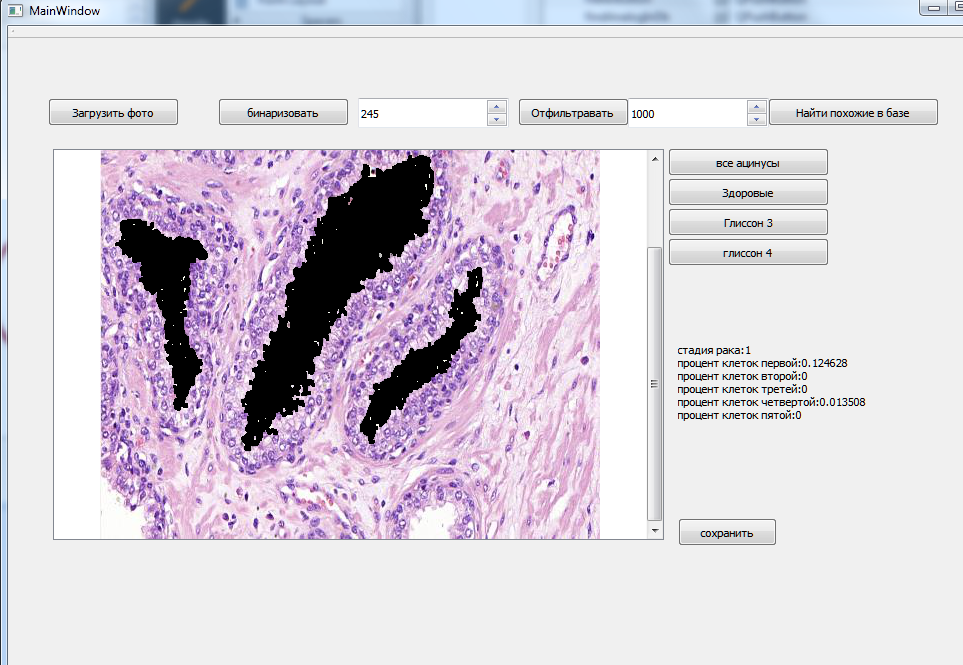
3. Распознавание и фильтрация

Пункт 1:



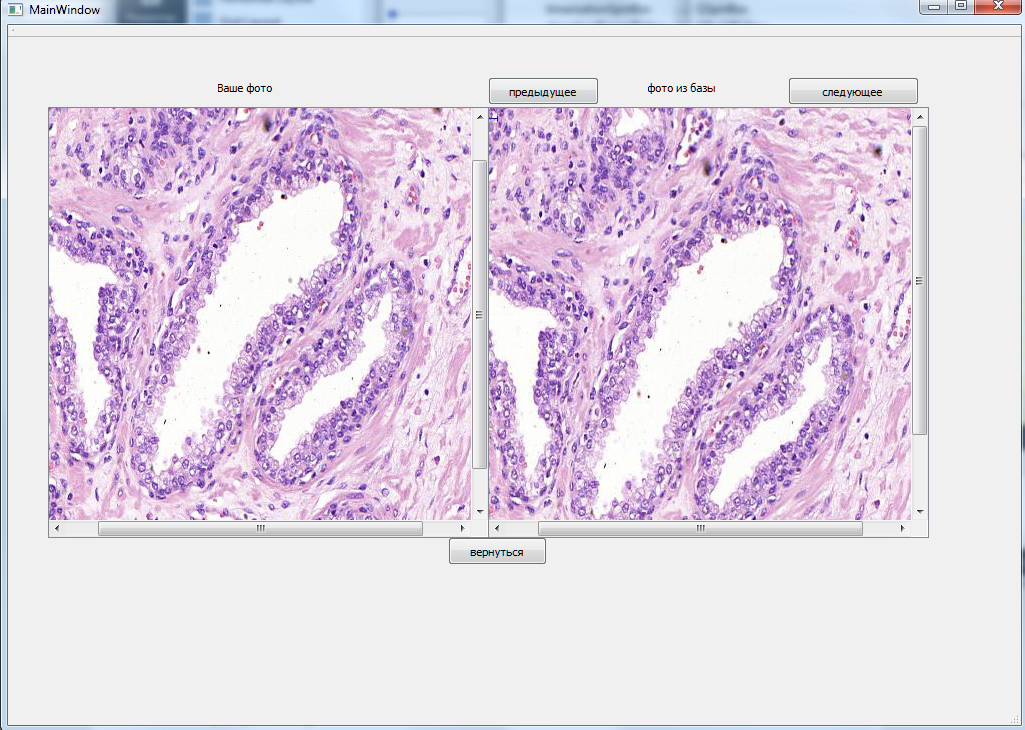
**Рис. 26.** Распознавание и фильтрация (1)

Пункт 2 отображение (опция: показать только здоровые ацинусы)



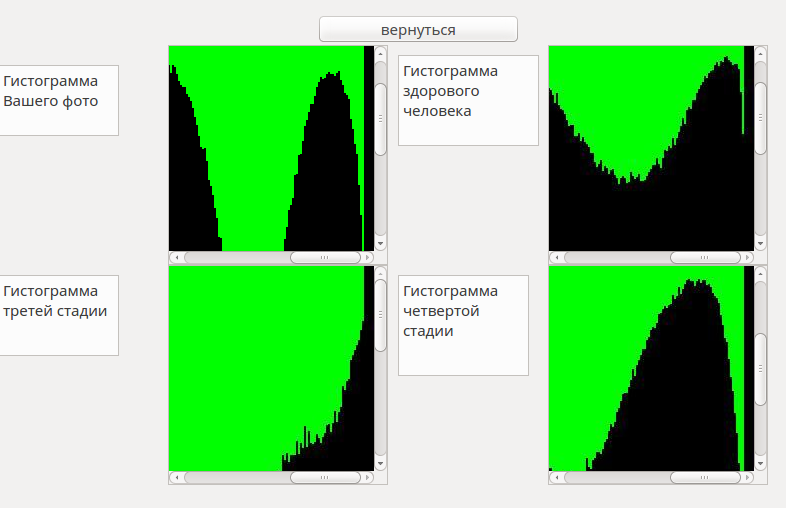
**Рис. 27.** Распознавание и фильтрация (2)

4. Сохранение (проверка с помощью поиска изображения в списке похожих)



**Рис. 28.** Сохранение

5. Вывод гистограмм



**Рис. 29.** Гистограммы

Результат работы программы в сценариях работы с программой, указанных в плане тестирования совпал с ожидаемым.

Во время проведения тестирования не ожидаемых ошибок программы и случаев аварийного завершения программы не было.

Программа удовлетворяет требованиям к базе данных, составу выполняемых функций и интерфейсу.

Изображения на которых проводилось тестирование приложены в архиве Приложение 1 Тестовые изображения

**6.2 Оценка эффективности работы системы при обработке реальных данных.**

**6.2.1. Цель и план эксперимента.**

Цель: определить области применимости системы , её характеристики быстродействия и точности . Так же нахождения слабых мест программы для избежание ошибок эксплуатации . Так же данный эксперимент необходим для того, чтобы выявить рекомендуемые параметры (те параметры , которые вводит пользователь)

План эксперимента :

* Последовательно загрузить в программу изображения размером 500\*500 и обработать с помощь доступных в данной программе средств .
* Оценить точность распознавания по формулу n/N, где n — количество снимков, на которых программа правильно распознала стадию Глисона , N — количество фотографий на которых проводили эксперимент.
* Оценить время необходимое программе для бинаризации.
* Оценить время необходимое программе для получения данных из базы данных .
* Оценить время необходимое для построения гистограммы
* Оценить качество выделения и распознавания ацинусов.
* Найти случаи в которых программа выделяет ацинусы наиболее правильно.
* Найти случаи в которых программа ошибочно.
* Определить рекомендуемые параметры.

**6.2.2. Результаты эксперимента.**

Эксперимент проводился на компьютере с операционной системой windows 7. Процессор core i5 4200U, 6гб оперативной памяти.

В эксперименте было использовано 27 снимков биоптатов предстательной железы, из них 4 фотографии 4-й стадии 17 фотографий здоровой ткани и 6 фотографий 3 -й стадии рака предстательной железы по шкале Глисона.

По итогам проведения эксперимента, программа ни разу не ошиблась.

Такие результаты можно объяснить тем, что при распознавании система с больших коэффициентом доверия использует анализ гистограмм. То опирается на сведенья о цвете и яркости изображения. На практике цвет и яркость изображений могут сильно отличаться от тестовой выборки. Так же на эти характеристики может влиять условия съемки, при которых были получены снимки биоптатов. Например цвет зависит от толщины исследуемого материала, а яркость может зависить от освещенности.

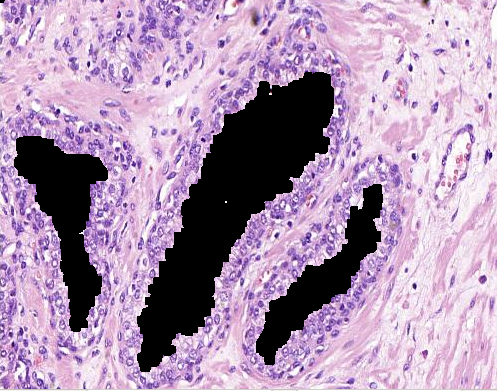
Время необходимое для бинаризации меньше 1 секунды

Время необходимое для распознавания о меньше 1 секунды

Время необходимое для поиска похожих изображений в базе знаний для человека незаметно, меньше секунды

Время, которые необходимое для того, чтобы получить их базы знаний эталонные гистограммы и построить гистограммы изображения загруженного пользователем для человека незаметны , меньше секунды.

По результатам эксперимента определено, что система лучше всего выделяет ацинусы, на фотографиях биоптатов здоровых клеток. Пример представлен на Рис 30.

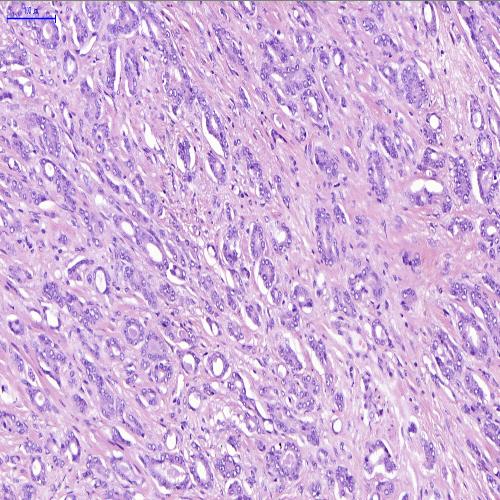


**Рис. 30.** Выделение ацинусов на снимках биоптата здоровой предстательной железы

Для фотографий, на которых области ацинусов близки по размеру хотя бы к одной трети изображения (то например да рис.20 протяженность в ацинусов в длину по размеру больше половины размера изображения по вертикали ). То рекомендуется фильтру размера ацинуса передавать параметр от 300 до 500.

Наихудшие результаты распознавания типов ацинусов программа показала, на изображениях со сравнительно небольшими размерами ацинусов.

Пример представлен на Рис. 31.



**Рис. 31.** Изображение с малым размером ацинусов

Для распознавания данного типа клеток рекомендуется использовать для фильтра размера ацинусов значения 50 .

Так как программа определяет стадию рака не только по форме и размерам ацинусов на результат распознавания ошибочно выделлыне ацинусы не оказывают сильного влияния.

**Заключение**

Настоящая работа посвящена теме разработки системы диагностики рака предстательной железы по фотографиям биоптатов.

В работе получены следующие результаты:

* В предпроектном исследовании были рассмотрены различные алгоритмы и подходы к решению проблемы
* Разработаны требования к системе
* Разработана система диагностики рака предстательной железы по фотографиям биоптатов на основе предоставленного изображения биоптата

В системе удалось реализовать все поставленные задачи.

Применение данной системы дает пользователю возможность получить информацию о стадии рака в соответствии с критериями Глисона, а так же дает возможность увидеть на основании чего программа приняла данное решение.

**Глоссарий**

1. Ацинус - Небольшая сумка или полость, окруженная секреторными клетками желез
2. Виджет – класс в иерархии наследования имеет родителя класса QWidget
3. Биоптат – материал полученный при взятии биопсии

**Список используемой литературы**

* 1. Аль-Шукри С.Х., Ткачук В.Н. Опухоли мочеполовых органов. СПб; 2015; 309 с.
  2. Библиотека патологоанатома // Сертификация, стандарты и качество патологоанатомических исследований; Вып. 69, Санкт-Петербург: 2016.
  3. Бухаркин Б.В., Подрегульский К.Э. Рак предстательной железы. Клин онкология 2009; 1(1): 5—8.
  4. Грачева Л. Выявление и предоперационный прогноз клинически незначимого рака предстательной железы / / РМЖ. Т. 5. М4. -2007.
  5. Карякин О.Б. Приоритеты в лечении различных стадий рака предстательной железы. Материалы II Российской онкологической конференции. Современные тенденции развития лекарственной терапии опухолей. 2014 М.
  6. Коган М.И., Якимчук Т.П., Шишков А.В., Волдохин А.В. Сравнтельный анализ диагностических методов при первичном обследовании больных раком простаты. Урол и нефрол 2009; 3: 38—41.
  7. Козлов В.П., Мазанов г.п., Ментешов И.В. Отдаленные результаты комбинированного лечения рака предстательной железы // Кремлевская медицина. Клин. вестн. 2012.-М3.
  8. Лоран О.Б., Пушкарь Д.Ю., Канн яд. и др. Радикальная простатэктомия у больных раком предстательной железы без признаков экстракапсулярной инвазии / В кн.: Актуальные вопросы лечения онкоурологических заболеваний. Материалы II Всероссийской конф. Обнинск, 2017.