Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет  
информатики и радиоэлектроники»

Институт информационных технологий

Кафедра «Микропроцессорные системы и сети»

|  |  |
| --- | --- |
|  | **К защите допустить**  Заведующая кафедрой  \_\_\_\_\_\_\_\_\_\_\_\_\_ Н.Л. Боброва  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2018 |

Пояснительная записка

к дипломной работе по теме:

«Программное средство для расчета химического состава образцов»

|  |  |
| --- | --- |
|  | Дипломник В.В. Захаренков  (дата, подпись) |
|  | Руководитель А.А. Москалёв  (дата, подпись) |
|  | Консультанты:  по ЕСКД и ЕСПД В.Н.Мухаметов  (дата, подпись) |
|  | по специальности А.А.Москалёв  (дата, подпись) |
|  | Рецензент  (дата, подпись) |

Минск 2018

Содержание

[Определения, обозначения и сокращения 2](#_Toc524619572)

[Введение 2](#_Toc524619573)

[1 Аналитический обзор и постановка задачи 2](#_Toc524619574)

[1.1 Программы для автоматизации расчетов химических анализов солевых образцов 2](#_Toc524619575)

[1.2 Сравнительный анализ существующего ПО 2](#_Toc524619576)

[1.2.1 Шаблоны MathCAD 2](#_Toc524619577)

[1.2.2 Файлы электронных таблиц MS Excel 2](#_Toc524619578)

[1.2.3 Итоговое сравнение 2](#_Toc524619579)

[1.3 Постановка задачи 2](#_Toc524619580)

[2 Методы и модели, положенные в основу проекта 2](#_Toc524619581)

[2.1 Методика определения принадлежности солевых образцов к соответствующему типу 2](#_Toc524619582)

[2.2 Методы оценки корректности проведения анализа и применения расчетной схемы 2](#_Toc524619583)

[2.3 Метод определения сухого веса образца 2](#_Toc524619584)

[3 Проектирование приложения 2](#_Toc524619585)

[3.1 Описание функциональности приложения 2](#_Toc524619586)

[3.2 Разработка базы данных 2](#_Toc524619587)

[3.2.1 Анализ объектов предметной области 2](#_Toc524619588)

[3.3 Разработка структуры программного средства 2](#_Toc524619589)

[4 Разработка и реализация основных алгоритмов 2](#_Toc524619590)

[4.1 Общая схема работы программной системы для расчета химического состава образцов 2](#_Toc524619591)

[4.2 Старт приложения 2](#_Toc524619592)

[4.3 Авторизация пользователя в системе 2](#_Toc524619593)

[4.4 Определение расчетной схемы 2](#_Toc524619594)

[4.5 Рекурсивное определение скорректированного сухого веса образца 2](#_Toc524619595)

[4.6 Определение коэффициентов корреляции калибровочной прямой 2](#_Toc524619596)

[4.7 Фильтрация образцов по дате отбора и лабораторным номерам 2](#_Toc524619597)

[4.8 Обновление данных калибровочной прямой 2](#_Toc524619598)

[4.9 Сравнение результатов расчета 2](#_Toc524619599)

[5 Тестирование приложения 2](#_Toc524619600)

[5.1 Подготовка к тестированию приложения 2](#_Toc524619601)

[5.1 Оценка результатов тестирования 2](#_Toc524619602)

[Заключение 2](#_Toc524619603)

[Список использованных источников 2](#_Toc524619604)

[Приложение А Текст программы 2](#_Toc524619605)

# Определения, обозначения и сокращения

В настоящей пояснительной записке применяются следующие термины и определения:

**БД** —база(ы) данных.

**Гигроскопическая вода** — вода, содержащаяся в образце, но не участвующая в образовании кристаллогидратов.

**Интеркалибровка** — контрольные измерения, проводимые в рамках нескольких лабораторий (либо одной лаборатории) с целью выявления источников систематических погрешностей применяемых методик.

**Корректность химического анализа** —под корректностью проведения химического анализа нахождение точности определения каждого отдельного компонента в заданном интервале, а также отсутствие неопределенных (необнаруженных) компонентов.

**Кристаллогидратная вода** — определенная молекулярная пропорция воды, химически соединенной с некоторыми веществами (кристаллогидратами) в кристаллическом состоянии.

**Лабораторный номер** — буквенно-цифровой код однозначно идентифицирующий данный образец; в рамках данного дипломного проекта уникален в пределах календарного года.

**ОЗУ** — оперативное запоминающее устройство.

**ПО** — программное обеспечение.

**ПС** — программная система.

**СУБД** — система управления базами данных.

**Химический анализ** — совокупность методов исследования, употребляемых для определения состава химических соединений или их смесей. В настоящей работе под совокупность методов понимается совокупность количественных и качественных методов.

**ЦПУ** — центральное процессорное устройство.

**EF** — Entity Framework.

**MVVM (Model-View-ViewModel)** Модель-Представление-Модель представления — архитектурный шаблон проектирования.

**RLS (Row Level Security)** Безопасность на уровне строк — технология, позволяющая контролировать доступ к строкам таблиц базы данных, основываясь на характеристиках пользователя, выполняющего запрос (например, членство в группе или же контекст исполнения) (1). Внедрена в ПС Microsoft SQL Server начиная с версии 2016.

# Введение

Валовой химический анализ солевых пород — достаточно распространенный анализ для химико-геологических лабораторий. Количество таких анализов в одной лишь лаборатории может достигать нескольких сотен в год. При этом каждый из анализов дает содержание отдельных ионов независимо от того, в каком виде эти соединения (минералы) входят в состав исследуемых образцов породы, и представляет собой массив данных, состоящий из нескольких десятков параметров. В задачу химика-аналитика входит, помимо прочего, представление количественного минералогического состава пород и определение их количественных отношений.

Одним из путей решения такой задачи (в общем случае достаточно нетривиальной) является расчетный путь. При таком подходе за основу берутся данные валового анализа. Затем проводятся предварительные расчеты с целью определить значения коэффициентов, по которым в дальнейшем и определяется требуемая модельная схема, отражающая стехиометрическое соотношение ионов в солевых образцах. Требуемые пороговые значения этих коэффициентов определены по результатам многолетних исследований (2), позволившим условно разделить все многообразие встречающихся геохимических форм на ограниченное количество основных условных модельных схем, значительно упростив тем самым расчетную задачу. После определения расчетной схемы производится окончательный расчет с использованием формул, применимых для данной конкретной схемы. Количественным показателем качества самого химического анализа, а также соответствие выбранной расчетной схемы природному составу конкретного образца, является близость сумм ионных и минеральных форм к 100 %, а также близость разности между суммами ионных и минеральных форм к нулю.

Расчетный путь связан с необходимостью проведения многоступенчатых расчетов, при которых нередки ошибки, напрямую связанные с человеческим фактором, а также потерей точности из-за округления при проведении многочисленных операций деления и умножения. В этой связи актуальным представляется решение следующих задач:

- автоматизация расчетов валовых химических анализов солевых образцов;

- создание базы данных для хранения исходных данных анализов, результатов расчетов, а также калибровочных данных, различных настроек и пр.;

- автоматизация подготовки сводных отчетов по результатам расчета анализов, выполненных по заданным схемах, в заданный интервал времени;

- облегчение проведения сравнения результатов расчета «параллельных» анализов.

Целью дипломного проекта является повышение оперативности выполнения обработки данных анализов, увеличение надежности и достоверности получаемых результатов и в целом повышение экономической эффективность работы.

Для достижения поставленной цели необходимо создание программного средства, позволяющего автоматизировать выполнение обработки данных и обеспечить решение указанных задач.

Для решения поставленных задач предполагается использовать следующие программные средства:

- Microsoft Visual Studio Community 2017 версия 15.7.5;

- Язык программирования C# целевая платформа .NET Framework версия 4.5.2, WPF Framework;

- Microsoft Entity Framework версия 6.0 в качестве средства обеспечения доступа к базе данных;

- СУБД Microsoft SQL Server 2016 Express, представляется наиболее приемлемым вариантом, принимая в расчет тот факт, что в химических (геохимических) лабораториях, как правило, отсутствует штатный сотрудник, который следил бы за состоянием и работой полноценного MS SQL Server’а, а также особенность данной версии, состоящей в автоматическом создании и запуске требуемой инфраструктуры SQL Server’а, что позволяет приложению использовать базу данных без решения сложных конфигурационных задач, в случае если подключение осуществляется через строку подключения (3).

Ожидается, что создание программного средства позволит значительно ускорить проведение расчетов, позволит сэкономить время подготовки отчетности за счет ее автоматизации, что безусловно должно повысить экономическую эффективность работы как отдельных сотрудников, так и лаборатории, выполняющей данный вид анализов, в целом.

# 1 Аналитический обзор и постановка задачи

Многоступенчатость расчетов, желание исключить влияние человеческого фактора и максимально ускорить и упростить проведение расчетов результатов химических анализов солевых образцов неизменно приводили к попыткам создания некоего подобия программных систем.

## 1.1 Программы для автоматизации расчетов химических анализов солевых образцов

В связи с тем, что затрагиваемая область достаточно узка, удалось обнаружить лишь два варианта решения. Первый вариант предполагает использование шаблонов системы MathCAD и требует наличия предустановленного пакета САПР MathCAD версии не ниже 6.0. Второй вариант реализован в виде файлов электронных таблиц MS Excel версии не ниже 2010.

## 1.2 Сравнительный анализ существующего ПО

Для составления адекватной оценки представляется важным определить перечень параметров, по которым будет проводиться оценка существующих программных систем. Подобными параметрами могут служить следующие:

- представляет ли система единое целое, а не набор утилит. Данный параметр представляется важным с точки зрения удобства использования;

- предоставляет ли система возможность формирования отчета по результатам проделанной работы;

- возможно ли сохранение результатов расчета во внешние файлы общеизвестных распространенных форматов;

- позволяет ли система проведение сравнения результатов расчета. Данный параметр особенно важен для химических (геохимических) лабораторий, участвующих в интеркалибровках;

- предоставляет ли система возможность одновременной работы с результатами, полученными для разных исходных данных для анализа (титров, калибровок и проч.);

- существует ли в рамках системы единое хранилище для исходных данных. Данный параметр представляется особенно важным для химических (геохимических) лабораторий, проводящих данный вид анализов на периодической основе, т.к. в таких лабораториях периодически возникает необходимость сравнения результатов анализов для аналогичных (одинаковых) образцов;

-предоставляет ли система возможность визуального контроля калибровочных кривых. Данный параметр позволяет проводить не только оценку качества используемой калибровочной кривой (ее качество может быть также оценено численно по коэффициенту корреляции), но также своевременно выявлять неисправности либо сбои в работе спектрофотометрического оборудования, используемого, как правило, для получения параметров таких кривых;

- существует ли в рамках системы ограничение на количество одновременно обрабатываемых образцов;

- существует ли в рамках системы ограничение на количество анализов, ассоциированных с данным конкретным образцом.

### 1.2.1 Шаблоны MathCAD

Так как Mathcad — это система компьютерной алгебры, ориентированная на подготовку интерактивных документов с вычислениями и визуальным сопровождением, работать с шаблонами MathCAD достаточно удобно: все операции осуществляются в пределах рабочих листов, на которых уравнения и выражения отображаются графически, любое внесение изменений в данные визуализированной калибровочной кривой отражается как на ее графике, так и на конечных результатах. Для ввода формул и данных можно использовать как клавиатуру, так и специальные панели инструментов. Стоит, однако, отметить, что при всех явных преимуществах данного решения оно, тем не менее, обладает рядом серьезных недостатков, существенно затрудняющих использование в повседневной практике химических (геохимических) лабораторий. К таковым можно отнести:

- система состоит из набора шаблонов, каждый из которых предназначен для решения лишь некоторой части задачи, после чего пользователь должен открыть следующий шаблон, вручную внести данные и продолжить при необходимости вычисления. Такой подход не исключает так называемый человеческий фактор, а внесенные при переносе (копировании) ошибки сложно отследить. Повторить расчет в таком случае наиболее простое и быстрое решение;

- отсутствует возможность формирования итогового отчета по проведенным расчетам (т.е. предполагается, что пользователь будет копировать результаты расчета для каждого показателя в требуемую итоговую форму). При таком подходе не исключаются ошибки, отслеживание и исправление которых по временным затратам может перечеркнуть все выгоды по применению шаблонов;

- сохранение результатов расчета возможно либо в виде XML-файлов, либо в двоичном проприетарном формате, причем каждый файл соответствует одному этапу расчета одного образца, такой подход очень удобен в случае применения в исследовательской практике, когда требуется многократно возвращаться к данным одного анализа, однако значительно затрудняет проведение расчетов при серийных экспериментах;

- отсутствует встроенная возможность проведения интеркалибровки либо сравнения результатов расчета для однотипных образцов;

- отсутствует единое хранилище для исходных данных и результатов расчетов;

- поскольку набор шаблонов предназначен для обсчета исключительно одного образца за один цикл, это автоматически налагает ограничения в 1 как на количество одновременно обрабатываемых образцов, так и на количество анализов, ассоциированных с данным конкретным образцом, кроме того это также делает невозможным работу с результатами анализов, полученными для различных исходных настроек.

### 1.2.2 Файлы электронных таблиц MS Excel

Значительно больше возможностей имеется в решении, осуществленном в виде электронных таблиц MS Excel. Все расчеты проводятся в одном файле, на нескольких листах, причем четко разделены листы для ввода исходных данных, листы, содержащие итоговые ведомости, а также лист, содержащий данные и графическое представление калибровочной кривой. Пересчет результатов при изменении параметров производится автоматически (если иное не установлено в настройках Excel). Результаты расчетов представлены в виде итоговых ведомостей, сформированных в зависимости от автоматически определяемой расчетной схемы. Однако данное решение, также не лишено недостатков, а именно:

- поскольку все образцы, обрабатываемые в каждом конкретном файле электронных таблиц, используют единую калибровочную кривую, а также единые настройки (концентрации титрующих растворов, их нормальности и пр.), совместить расчет двух и более образцов, использующих разные вышеуказанные параметры, в одном файле оказывается невозможным;

- провести интеркалибровку либо сравнение возможно лишь при совпадении параметров, упомянутых в предыдущем пункте, а также несколько доработав электронные таблицы, что предполагает наличие соответствующих навыков у пользователя;

- количество образцов, обрабатываемых в одном файле (за один раз) ограничено 30;

- поскольку в данном решении не проводится разграничение между анализом образца и образцом, это автоматически налагает ограничение на количество анализов, ассоциированных с данным конкретным образцом, равное 1 (система не предполагает, что один образец может быть многократно проанализирован);

- отсутствует единое хранилище для исходных данных и результатов расчетов;

-экспортировать результаты расчета можно использую штатные средства MS Excel.

### 1.2.3 Итоговое сравнение

Итоговые результаты сравнительного анализа двух программных систем приведены в Таблица 1.

Таблица 1 – сравнительная оценка реализации основных требований пользователя в ПО, применяемом для автоматизации расчетов химических анализов солевых образцов

|  |  |  |
| --- | --- | --- |
|  | Шаблоны MathCAD | Файл Excel |
| Система представляет собой единое целое (не набор утилит) | - | + |
| Возможность формирования отчета по результатам | - | + |
| Сохранение результатов расчета во внешние файлы | - | + |
| Возможность проведения сравнения результатов расчета | - | - |
| Возможность одновременной работы с результатами, полученными для разных настроек | - | - |
| Наличие единого хранилища для исходных данных | - | - |
| Визуальное представление калибровочных кривых | + | + |
| Количество одновременно обрабатываемых образцов | 1 | 30 |
| Количество анализов, ассоциированных с данным конкретным образом | 1 | 1 |

На основании анализа существующих программных решений можно заключить, что наиболее близким (отвечающим большему числу заданных параметров) из рассмотренных вариантов можно назвать прототип, использующий файлы электронных таблиц MS Excel.

## 1.3 Постановка задачи

Проектируемая программная система предназначается для проведения расчетов результатов химических анализов солевых образцов. Методология, по которой будут осуществляться расчеты приведена в (2). Для облегчения решения основной задачи, а также с целью минимизации ошибок, связанных с необходимостью повторного введения данных, в рамках выполнения задачи следует организовать единое хранилище для данных анализов, настроек и данных калибровочных кривых. Наиболее полно требованиям отвечает СУДБ MS SQL Server Express Edition, так как предоставляет возможности, доступные в большинстве платных СУБД, но является при этом бесплатной. Ограничения, налагаемые бесплатной (Express Edition) версией по сравнению с ближайшей платной (Standard Edition):

- максимальный объем БД 10 Гб;

- отсутствие сервиса MS SQL Server Agent;

- искусственные ограничения, налагаемые на физические ресурсы сервера: 1Гб ОЗУ, только одно физическое ЦПУ и некоторые другие;

являются несущественными для решения данной задачи, так как максимальный объем в 10 Гб является явно недостижимым для средней химической лаборатории, проводящей порядка 1000 анализов в год, даже с учетом нескольких лет эксплуатации. Отсутствие MS SQL Server Agent не скажется на работе СУБД, так как создаваемое ПС не предполагает решения никаких административных задач СУБД, требующих действий по расписанию.

В свою очередь СУБД MS SQL Server легко интегрируется в решения, выполненные на языке программирования C# с применением технологии .NET и фреймворка MS Entity Framework. Применение же системы для построения клиентских приложений Windows Presentation Foundation (WPF), входящей в качестве подсистемы в .NET Framework, позволит отделить логику приложения от визуальной части (представления), как и предполагается в случае использования шаблона проектирования MVVM. Применение всех вышеупомянутых схем может значительно ускорить и упростить решение поставленной задачи.

Сохранение такой важной для данной задачи информации, как данные калибровочных кривых в общей СУБД, позволит сократить время на подготовку химических анализов, так как в случае использования СУБД данные калибровочных кривых могут быть совместно использованы несколькими химиками, выполняющими свои собственные аналитические задачи.

Использование в проекте Entity Framework (EF) предполагает выбор между двумя походами, допустимыми в рамках EF:

- Database First, позволяющий писать приложения для существующих баз данных (4);

- Code First, обычно использующийся, при наличии уже существующего приложения, содержащего модель данных (5). Эта модель, как правило, описывается с помощью нескольких классов и кода взаимодействия между этими классами.

Оба подхода приводят к созданию экземпляра класса контекста и использованию этого объекта для выполнения необходимых задач. Для разработки приложений больше подходит, конечно, подход Code First, так как позволяет многократно создавать и/или изменять требуемую БД исходя из меняющейся модели. Однако такое упрощение работы с созданием БД оставляет проверку сложных условий, налагаемых на свойства классов модели (как, например, ограничение на допустимый интервал значений свойства класса, зависящее от значения свойства этого же либо связанного класса, или же допустимость равенства значения свойства класса NULL, в зависимости от значения другого свойства класса), полностью на стороне языка программирования приложения, что не всегда допустимо. Решение задачи в рамках данного дипломного проектирования предполагает наличие таких свойств у классов, контроль интервалов значений которых зависит от значений других свойств, поэтому создаваемой программной системе необходимо наличие БД, контролирующей соблюдение именно таких ограничений на стороне БД. Соответственно подход Database First является наиболее предпочтительным.

Система должна позволять пользователю совершать следующие действия:

- проводить манипуляции с образцами, а именно создавать новый образец, изменять данные для уже существующего образца, просматривать информацию об образце, а также удалять образцы, осуществляя при этом контроль за возможностью проведения таких операций;

- добавлять к выбранному образцу данные химических анализов (от 1 до 30 за одну операцию), изменять данные таких анализов, в том числе вносить дополнительную информацию об анализе, а также удалять данные анализов. Подобные операции должны осуществляться в отдельном диалоговом окне;

- осуществлять аутентификацию пользователя на основании введенной пары логин-пароль. Проверка должна выполняться на стороне БД посредством выполнения системной хранимой процедуры sp\_setapprole. Фильтрация информации, предназначенной для отображения или изменения, должна осуществляться на уровне базы данных за счет использования технологии RLS с учетом данных проведенной аутентификации;

- давать графическое представление калибровочной прямой, с отображением точек, на основании, которых она построена, а также выводить данные по тангенциальному коэффициенту, ординаты пересечения с осью Y и позволять пользователю оценивать качество аппроксимации через коэффициент корреляции – R2;

- давать возможность фильтрации списка образцов по дате отбора (с заданием интервала дат), а также по лабораторному номеру. При отсутствии заданного для фильтрации лабораторного номера, система должна выводить все образцы, даты отбора которых попадают в заданный временной интервал;

- не должна налагать ограничений на количество анализов, связанных с данным конкретным образцом;

- давать возможность каждому анализу обладать своими собственными начальными установками, включая калибровочную кривую;

- определять оптимальную расчетную схему для анализа, оставляя тем не менее за пользователем право окончательного решения о том, по какой схеме проводить расчет;

- предоставлять пользователю функционал для ввода начальных параметров, значения которых будут присваиваться соответствующим параметрам анализов по умолчанию; такого рода параметры должны сохраняться в настройках программной системы, извлекаться при ее старте и быть уникальными для каждого конкретного пользователя конкретного ПК;

- предоставлять возможность осуществлять просмотр, корректировку, а также задание новых данных для калибровочных кривых не только из диалогового окна работы с калибровками, но и из диалогового окна работы с анализами для образцов;

- предотвращать изменение калибровочных данных пользователем в случае, если эти данные уже используются для расчета результатов другого анализа (т.е. в случае, если количество ссылок на калибровочную кривую больше или равно 1);

- давать предварительную оценку качества проведения химического анализа для выбранной схемы, путем сравнения суммарных величин с параметром, отражающим допустимый толеранс (параметр должен задаваться пользователем с сохранением в настройках системы, извлекаться при ее старте и быть уникальным для каждого конкретного пользователя конкретного ПК);

- позволять проводить сравнение результатов расчетов двух анализов между собой, при условии, что для данных анализов выбраны одинаковые расчетные схемы; оценка сходимости результатов должна осуществляться на основании критериев, задаваемых пользователем для каждой схемы в отдельности, (параметры сохраняются в настройках системы, извлекаются при ее старте и являются уникальными для каждого конкретного пользователя конкретного ПК);

- предоставлять результаты расчетов для выбранных анализов выбранных образцов в виде итоговых ведомостей установленного образца, при этом результаты расчета сводятся для каждой расчетной схемы в отдельную ведомость;

- предоставлять возможность сохранять результаты расчета в сторонний файл.

# 2 Методы и модели, положенные в основу проекта

## 2.1 Методика определения принадлежности солевых образцов к соответствующему типу

Солевая форма выражения результатов анализов солевых образцов весьма удобна для ряда построений и выводов, поэтому, несмотря на некоторые условности пересчетов на нее, она широко используется в химических и геохимических лабораториях.

Связывание ионов в соли производится по определенным правилам. Наибольшее распространение получил принцип связывания ионов соли в порядке возрастания растворимости последних. Следуя этому принципу, ионы связывают в первую очередь в малорастворимые соединения, а затем – во все более и более растворимые.

Общая схема последовательного связывания ионов в соли следующая:

- Ca2+ связывают последовательно с CO, HCO3-, SO, и Cl-;

- Mg2+ также последовательно связывают с CO, HCO, SO, и Cl-, если первые три иона не были полностью израсходованы на связывание Ca2+;

- K+ всегда связывают с Cl-;

- остатки несвязанных анионов связывают с Na+, с ним же связывают Br- и B4O7-;

- второстепенные катионы, содержащиеся в солевых образцах в небольших количествах, такие как Li+, Sr2+ и другие, связывают с Cl-, а анионы – PO43-, J- и другие – с Na+.

Все многообразие составов природных солевых образцов можно условно разделить на три основных химических типа (2):

- карбонатные;

- сульфатные (выделяя среди них подтипы сульфатнонатриевый и сульфатномагниевый);

- и хлоридный.

Каждый из этих типов характеризуется составом своей солевой массы и коэффициентами (отношениями между содержанием отдельных солей).

Для карбонатного типа солевых образцов характерный коэффициент рассчитывается по формуле

|  |  |
| --- | --- |
|  | (1) |

где – нормальная концентрация карбоната натрия;   
 – нормальная концентрация гидрокарбоната натрия;   
 – нормальная концентрация сульфата натрия.

Для сульфатного типа солевых образцов в связи с наличием дополнительного деления на подтипы рассчитываются два характерных коэффициента и по формулам

|  |  |
| --- | --- |
|  | (2) |

где – нормальная концентрация сульфата натрия;   
 – нормальная концентрация сульфата магния.

|  |  |
| --- | --- |
|  | (3) |

где – нормальная концентрация сульфата магния;   
 – нормальная концентрация хлорида магния.

Для хлоридного типа солевых образцов характерный коэффициент рассчитывается по формуле

|  |  |
| --- | --- |
|  | (4) |

где – нормальная концентрация хлорида магния;   
 – нормальная концентрация хлорида кальция.

Однако приведенные коэффициенты подходят для уже проанализированных солевых образцов, у которых определено содержание характерных солей. Поэтому помимо коэффициентов, определяемым по формулам (1)—(4), для отнесения солевого образца с еще не определенным типом солевой схемы необходимо пользоваться коэффициентами, определяемыми из количеств эквивалентных ионов в солевом образце. Расчетные формулы для таких коэффициентов приведены ниже.

|  |  |
| --- | --- |
|  | (5) |

где – эквивалентная концентрация карбонат-иона;   
 – эквивалентная концентрация гидрокарбонат-иона;   
 – эквивалентная концентрация иона кальция;   
 – эквивалентная концентрация иона магния.

|  |  |
| --- | --- |
|  | (6) |

где – эквивалентная концентрация карбонат-иона;   
 – эквивалентная концентрация гидрокарбонат-иона;   
 – эквивалентная концентрация сульфат-иона;   
 – эквивалентная концентрация иона кальция;   
 – эквивалентная концентрация иона магния.

|  |  |
| --- | --- |
|  | (7) |

где – эквивалентная концентрация сульфат-иона;   
 – эквивалентная концентрация иона кальция.

|  |  |
| --- | --- |
|  | (8) |

где – эквивалентная концентрация карбонат-иона;   
 – эквивалентная концентрация гидрокарбонат-иона;   
 – эквивалентная концентрация иона кальция.

Общее соотношение коэффициентов, характеризующих возможность данного солевого образца быть отнесенным к той или иной расчетной схеме приведены в Таблица 2.

Таблица 2 – Соотношение коэффициентов для характеристики типов солевых образцов

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Тип природных солевых образцов | |  |  |  |  |
| Карбонатный | | > 1 | ≫ 1 | ≫ 1 | ≫ 1 |
| Сульфатный | сульфатнонатриевый | ≤ 1 | ≥ 1 | ≫ 1 |  |
| сульфатномагниевый | ≪ 1 | ≤ 1 | ≫ 1 |  |
| Хлоридный | | ≪ 1 | < 1 | ≤ 1 | < 1 |

Как видно из Таблица 2 для сульфатного типа солевых образцов отдельно рассматриваются случаи, когда > 1 и < 1. Это обстоятельство приводит к дополнительному делению сульфатнонатриевых и сульфатномагниевых подтипов солевых образцов на подтипы I (суммарное содержание карбонат и гидрокарбонат ионов меньше содержания иона кальция) и II (суммарное содержание карбонат и гидрокарбонат ионов больше содержания иона кальция) и, соответственно, необходимости учитывать это деление при проведении расчетов.

## 2.2 Методы оценки корректности проведения анализа и применения расчетной схемы

Корректность проведения химического анализа солевого образца в ионной форме оценивается по формуле

|  |  |
| --- | --- |
|  | (9) |

где – сумма эквивалентных концентраций всех анионов;   
 – сумма эквивалентных концентраций всех катионов;   
 – допустимое отклонение.

|  |  |
| --- | --- |
|  | (10) |

где – сумма массовых процентных концентраций всех солей;   
 – массовая процентная концентрация нерастворимого осадка;   
 – массовая процентная концентрация общей воды;   
 – допустимое отклонение.

## 2.3 Метод определения сухого веса образца

Так как в влага может входить в состав кристаллогидратов, например, карналлита KCl·MgCl2·6H2O. В расчетную задачу входит также подзадача определения доли воды, участвующей в образовании кристаллогидратов, а также связанная с ней подзадача определения скорректированного сухого веса вещества.

Поскольку из всех определяемых катионов только Mg способен образовывать кристаллогидраты, а также поскольку в основной массе случаев таким кристаллогидратом является карналлит (2). Расчет ведется по формулам:

|  |  |
| --- | --- |
|  | (11) |

где – вновь определенный вес кристаллизационной воды;   
– сухой вес магния до коррекции;   
– коэффициент, равный отношению атомного веса воды, содержащейся в карналлите к атомному весу карналлита.

|  |  |
| --- | --- |
|  | (12) |

где – скорректированный вес гигроскопической воды;   
– содержание (вес) воды, определенный при 180°С;   
 – вновь определенный вес кристаллизационной воды.

|  |  |
| --- | --- |
|  | (13) |

где – скорректированный сухой вес образца;   
– сырой вес образца;   
 – скорректированный вес гигроскопической воды.

|  |  |
| --- | --- |
|  | (14) |

где – скорректированный сухой вес образца;   
– сырой вес образца;   
 – скорректированный сухой вес магния;   
 – сырой вес магния.

Проверка достаточности корректировки производится по формуле:

|  |  |
| --- | --- |
|  | (15) |

где – скорректированный сухой вес магния;   
– сухой вес магния до коррекции;   
 – допустимое отклонение, как правило равное минимальной цене деления лабораторных весов (т.е. 0.0001).

Если требуемая точность коррекции не достигнута, расчет повторяют, принимая значения скорректированных весов магния и образца за нескорректированные веса. При достижении требуемой точности корректировки значения скорректированных весов магния и образца считают верными.

# 3 Проектирование приложения

## 3.1 Описание функциональности приложения

Диаграмма бизнес-вариантов использования системы показана на Рисунке 1. Основные задачи, решаемые на данном этапе следующие: различные манипуляции с образцами, работа с данными калибровочных кривых, а также аутентификация в системе (либо смена пользователя).

Часть диаграммы использования системы, включающая авторизацию и задание начальных параметров, отражена на .Рисунке 2

Рисунок 1 – Бизнес варианты использования системы

Рисунок 2 – Варианты использования системы (часть авторизация и задание параметров)

Часть диаграмма вариантов использования системы, отражающая основные функции, необходимые пользователю для работы с образцами, показана на Рисунке 3. Основные задачи, решаемые на данном этапе следующие: ввод нового образца, просмотр и/или корректировка данных существующего образца, удаление образца.

Рисунок 3 – Варианты использования системы (часть работа с образцами)

Часть диаграмма вариантов использования системы, отражающая основные функции, необходимые пользователю для работы с данными калибровочных кривых, показана на Рисунке 4. Основные задачи, решаемые на данном этапе следующие: ввод новой калибровочной кривой, редактирование данных существующей калибровочной кривой, просмотр графика калибровочной кривой, удаление существующей калибровочной кривой (в случае, если отсутствуют связанные с ней данные анализов), выбор текущей калибровочной прямой для определения значений концентрации (в случае работы из окна редактирования данных анализов), а также установление текущей калибровочной прямой в качестве калибровки по умолчанию.

Рисунок 4 – Варианты использования системы (часть работа с данными калибровочных кривых)

Часть диаграмма вариантов использования системы, отражающая основные функции, необходимые пользователю для работы с данными анализов, показана на Рисунке 5. Основные задачи, решаемые на данном этапе следующие: ввод данных новых анализов, редактирование данных существующих анализов, расчет результатов (с обязательным выбором схемы расчета). После того, как произведен расчет возможно распечатать его результаты, или сравнить результаты расчета одного анализа с результатами расчета другого (при условии одинаковости расчетной схемы), также экспортировать результаты расчета в файл в формате MS Excel.

На основании анализа предметной области, аналогов разрабатываемого программного средства, а также исходя из поставленной задачи, сформирована трассировочная матрица, приведенная в Таблице 3.

Рисунок 5 – Варианты использования системы (часть работа с данными анализов)

Таблица 3 – Трассировочная матрица

| № | Пожелание заказчика | № | Функциональное требование | Вариант использования |
| --- | --- | --- | --- | --- |
| 1. | Создавать новый образец, изменять данные существующего образца, просматривать информацию об образце, удалять образец. Фильтровать образцы по дате отбора и лабораторному номеру | 1.1 | Отобразить список образцов | Вывод списка образцов |
| 1.2 | Фильтровать список образцов | Фильтрация списка образцов |
| 1.3 | Добавить новый образец | Добавление нового образца |
| 1.4 | Просмотреть параметры образца | Просмотр параметров образца |
| 1.5 | Редактировать описание и дату отбора образца | Редактирование параметров образца |
| 1.6 | Удалить образец из базы | Удаление образца |
| 2. | Добавлять к выбранному образцу данные анализов, просматривать данные анализов, изменять и удалять их. | 2.1 | Добавить данные анализа к образцу | Добавление анализа |
| 2.2 | Просмотреть данные анализа | Просмотр данных анализа |
| 2.3 | Изменить данные анализа | Изменение данных анализа |
| 2.4 | Удалить данные анализа | Удаление данных анализа |
| 2.5 | Рассчитать данные анализов | Расчет данных анализов |
| 2.6 | Задавать расчетную схему | Задание расчетной схемы |
| 2.7 | Сравнивать результаты расчета анализов | Сравнение результатов расчета |
| 2.8 | Печатать результаты расчета анализов | Печать результатов расчета |
| 2.9 | Экспортировать результаты расчета анализов во внешний файл в формате Excel | Экспорт результатов расчета |
| 2.10 | Оценивать качество результатов расчета по выбранной схеме | Оценка качества расчета |

Продолжение Таблицы 3

| № | Пожелание заказчика | № | Функциональное требование | Вариант использования |
| --- | --- | --- | --- | --- |
| 3. | Просматривать, создавать, изменять калибровки; отображать графическое представление калибровочных прямых, просматривать данные калибровочных прямых и изменять эти данные. | 3.1 | Просмотреть список калибровок | Просмотр списка калибровок |
| 3.2 | Создать калибровку | Создание калибровки |
| 3.3 | Удалить калибровку | Удаление калибровки |
| 3.4 | Изменить калибровку | Изменение калибровки |
| 3.5 | Просмотреть данные калибровочной прямой | Просмотр данных калибровки |
| 3.6 | Изменить данные калибровочной прямой | Изменение данных калибровки |
| 3.7 | Просмотреть графическое представление калибровочной прямой | Просмотр графика прямой |
| 3.8 | Определять концентрацию по показаниям прибора | Определение концентрации |
| 4. | Задавать начальные параметры | 4.1 | Просматривать начальные параметры | Просмотр начальных параметров |
| 4.2 | Изменять начальные параметры | Изменение начальных параметров |
| 5. | Осуществлять авторизацию пользователей на основе введенной пары логин – пароль; фильтровать информацию с учетом проведенной авторизации | 5.1 | Авторизовать пользователя | Авторизация пользователя |
| 5.2 | Фильтровать информацию на основе введенных логина и пароля | Фильтрация данных |

## 3.2 Разработка базы данных

### 3.2.1 Анализ объектов предметной области

Сущности предметной области описываются в Таблица 4.

Таблица 4 – Сущности предметной области

|  |  |
| --- | --- |
| Сущность | Текстовое пояснение |
| Образец | Некоторая часть природного солевого материала, отобранная с соблюдением правил отбора образцов в определенное время и в определенном месте. Для каждого образца предполагается проведение одного или более химических анализов с целью выявления химического состава (ионная форма) и расчета солевого состава (солевая форма). |
| Анализ | Действие, проведенное в определенное время с использованием совокупности методов, позволяющих установить элементный и молекулярный состав исследуемого объекта или содержание отдельных его компонентов. Конечной целью анализа является получение численных характеристик содержания отдельных его компонентов (ионный состав). |
| Калибровка | Набор пар значений концентрация-показание прибора, снятых для эталонного набора образцов в определенное время с целью выявления зависимости между концентрацией определенного вещества (иона) и показаниями прибора. В настоящей работе рассматривается только линейная зависимость (в соответствии с законом Бугера-Ламберта-Бера) между концентрацией и оптической плотностью раствора, находящегося в кювете установленного размера. |
| Расчетная схема | Определенная последовательность преобразования химического состава образца, выраженного в ионной форме, в солевую форму; характеризуется коэффициентом. |

Инфологическая модель базы данных приведена на . Для простоты на схеме не отображена часть атрибутов сущности «Анализ», отвечающая за хранение непосредственно результатов анализа, т.е. не участвующая в связях.

Рисунок 5 – Инфологическая модель базы данных

Итоговая схема базы данных приведена на Рисунок 6. При проектировании базы данных было решено не хранить в ней расчетные данные, так как в процессе работы химика-аналитика с результатами анализа, введенные данные подвергаются корректировке (некоторые этапы анализа требуют повторного исполнения), к тому же разные расчетные схемы требуют хранения разных данных, поэтому в таблице, содержащей данные анализов, отсутствует атрибуты, предназначенные для хранения результатов расчета (в том числе, соответствующий расчетной схеме). К тому же изменение выходных солевых формул стандартных солевых схем потребует изменения программного кода (кода модуля, отвечающего за расчет), а в случае хранения итоговых расчетных значений в базе данных, потребовалось бы также изменение структуры таблицы, хранящей такие данные. В базе данных так же отсутствует таблица, отвечающая сущности «Расчетная схема». Функциональная нагрузка, соответствующая сущности, реализована в виде перечисления (Enum) на стороне C#.

Рисунок 6 – Схема базы данных 

В таблицах, соответствующих сущностям «Образец» (Sample) и «Анализ» (SaltAnalysis) введен дополнительные атрибут PrincipalID, отвечающий за реализацию доступности записей в этих таблицах в зависимости от пользователя, прошедшего аутентификацию, осуществляемую на уровне самой базы данных по технологии RLS. Значение атрибута установлено по умолчанию равным database\_principal\_id текущего пользователя.

В таблице Sample введено ограничение на значение атрибута SamplingDate, хранящим дату отбора образца, таки образом, что дата отбора может быть равна текущей дате, либо лежать в прошлом. Здесь так же добавлен вычисляемый хранимый атрибут (PERSISTED (6)), содержащий год отбора пробы. Данный атрибут участвует в организации ограничения, контролирующем уникальность атрибута LabNumber (лабораторный номер) в пределах календарного года.

В таблице CalibrationData введено ограничение, обеспечивающее уникальность калибровочной точки в пределах указанного диапазона, данной калибровки.

В таблице SaltAnalysis введены простые ограничения, обеспечивающие соответствие показателей здравому смыслу (например, значение аликвот не может быть меньше или равным нулю и т.д.), а также более сложные ограничения, также позволяющие сохранять логику приложения, например, значение атрибута, соответствующего весу высушенного образца при 110 °С должно находиться между значением веса сырого образца и весом пустого тигля, а также в случае если значение атрибута, соответствующее весу высушенного образца при 180 °С не равно NULL, оно должно быть меньшим или равным значению веса, высушенного образца при 110 °С, но большим веса пустого тигля.

## 3.3 Разработка структуры программного средства

С учетом необходимости хранить данные в базе данных, а также принимая во внимание необходимость обеспечивать одновременный доступ к такой базе данных нескольких пользователей, оптимальным вариантом архитектуры разрабатываемого программного средства является архитектура «клиент-сервер». Кроме того, на сервере базы данных проще организовать контроль за полномочиями пользователей, сервер, как правило, обладает более мощной защитой, по сравнению с рабочими станциями. Сервер также обладает встроенными средствами резервного копирования, использование которых снижает вероятность утраты данных.

Схема ресурсов программного средства приведена на листе графической части РТДП 006032.121.02.

# 4 Разработка и реализация основных алгоритмов

## 4.1 Общая схема работы программной системы для расчета химического состава образцов

Схема работы программы приведена на листах графической части РТДП 006032.121.01.

Работа приложения начинается с авторизации. Пользователь вводит в диалоговом окне авторизации логин и пароль, система предпринимает попытку провести авторизацию по алгоритму, описанному в 4.3 Авторизация пользователя в системе». При этом ведется подсчет неудачных попыток, после третьей попытки система выводит сообщение пользователю и инициализирует закрытие приложения.

После успешной авторизации происходит инициализация главного окна приложения. В главном окне пользователь выбирает необходимую функцию посредством меню либо комбинации горячих клавиш.

Работа с настройками исходных данных осуществляется в диалоговом окне настроек. При загрузке диалогового окна система считывает сохраненные настройки из файла конфигурации приложения и устанавливает соответствующие значения текстовых полей и переключателей. Закрытие диалогового окна по нажатию «OK» к сохранению значений текстовых полей и переключателей в файле конфигурации приложения.

Из меню файл при выборе доступна функция смены пользователя, которая осуществляется аналогично начальной авторизации, за тем лишь исключением, что неудачная авторизация не приводит к закрытию приложения, а оставляет в системе права начального пользователя.

В меню Образец можно вызвать окно списка образцов, предоставляющее основной функционал по работе с образцами и химическими анализами.

Меню Настройки предоставляют возможность пользователю изменить настройки программной системе, применяемые по умолчанию к образцам, анализам и калибровкам.

### 4.2 Старт приложения

На начальном этапе предпринимается попытка создать мьютекс (блок №1 см. Рисунок 7). Затем система ждет в течение 1 с (блок №2) сигнального состояния мьютекса (для того, чтобы избежать ситуации, когда другая копия приложения, создавшая мьютекс, завершила работу, а наша копия не стала работать из-за обнаруженного мьютекса).

Если сигнальное состояние мьютекса не наступает в течение 1 с (блок №3), то система информирует (MessageBox блок №4) пользователя о том, что на данной машине уже работает запущенная копия приложения, и затем производит широковещательную рассылку (HWND=0xFFFF блок №5) при помощи функции PostMessage пользовательского системного сообщения, зарегистрированного предыдущей копией программы (системная функция RegisterWindowMessage) при ее успешном старте. Получив такое сообщение предыдущая копия приложения восстанавливает состояние главного окна в нормальное и помещает его поверх всех окон (topmost). После этого система инициирует закрытие приложения (блок №6).

Если же сигнальное состояние мьютекса наступает в течение 1 с, система предпринимает попытку считать файл конфигурации системы (блок №7) и, при отсутствии ошибок (блок №8), передает управление окну-заставке (splashscreen блок №9). Возникновение же ошибки приводит к тому, что система инициирует закрытие приложения (блок №6).

Организация подобного приложения-одиночки (singleton) необходима для того, чтобы избежать ситуации, когда пользователь изменяет начальные настройки системы в одной из копий приложения, а работу продолжает в другой, полагая, что настройки изменены. В подобной ситуации возможно возникновение трудно отслеживаемых ошибок.

Окно-заставка запускает таймер (2 с), создает экземпляр главного окна, но не отображает его сразу, а регистрирует его в качестве интерфейса пользователя (StartUpURI (7)), автоматически загружаемого системой, т.е. после закрытия окна-заставки система передает управление главному окну приложения.

Рисунок 7 – Схема алгоритма старта системы

### 4.3 Авторизация пользователя в системе

Аутентификация пользователя в системе производится при возникновении события главного окна SourceInitialized (8), а также при выборе пользователем пункта меню «Сменить пользователя» посредством вызова метода главного окна Authorize. При этом методу передается логический параметр, указывающий необходимость смены пользователя.

Схема алгоритма аутентификации приведена на Рисунок 8. Метод Authorize() вызывает диалоговое окно, дающее пользователю возможность ввести логин и пароль (блок №1). Затем осуществляется проверка (блок №2) не нажал ли пользователь кнопку «Отмена». При положительном результате система выводит сообщение о невозможности работать без аутентификации (блок №3) и инициирует завершение приложения (блок №4).

На следующем этапе предпринимается попытка прочесть строку подключения из файла конфигурации приложения (блок №5), при возникновении ошибки, связанной как с отсутствием файла конфигурации, так и с отсутствием самой именованной строки подключения (блок №6), выводится сообщение (блок №21) и инициируется завершение приложения (блок №4).

Затем предпринимается попытка расшифровать, составляющие части строки подключения («data source» — блок №7; «UserID» — блок №8 и «password» — блок №9) при помощи установленного в системе ключа, название которого также хранится в файле конфигурации. Расшифрованная в случае успеха строка подключения передается в качестве параметра базовому конструктору класса DBContext (блок №10), в случае ошибки при расшифровке, связанной как с отсутствием названия ключа, так и отсутствием самого ключа, в качестве параметра передается нерасшифрованная строка подключения.

Проверяется значение параметра relogin (блок №12) и внутреннего поля класса (блок №13). В случае, если это не смена пользователя и значения имени пользователя и его пароля были установлены в предыдущей попытке, происходит возврат экземпляра объекта контекста базы данных (блок №18), а также предпринимается попытка прочесть строку в таблице CalibrationType (блок №19), так как в ней по умолчанию содержится хотя бы одна строка. При возникновении ошибки (блок №20) система инициирует закрытие приложения.

В случае необходимости смены пользователя или это первая попытка соединение с новым именем пользователя и паролем, система создает новый экземпляр класса DbConnectionApplicationRoleInterceptor (реализует интерфейс IDbConnectionInterceptor (9), блок №14), в котором осуществляется перехват событий, генерируемых Entity Framework. В качестве параметров конструктору класса передаются имя пользователя и пароль. В теле обработчика события Opened, генерируемого Entity Framework при открытии соединения, производится попытка выполнить системную хранимую процедуру sp\_setapprole ( (10) блок №15), в качестве параметров которой передаются имя пользователя и пароль, сохраненные в приватных полях класса, предназначенных только для чтения. Если выполнение запроса прошло без ошибок и значения параметра cookie, устанавливаемое базой данных отлично от 0xFFFFFFFF (блок №16), происходит сохранение имени пользователя и пароля (блок №17), значение свойства IsUserIDAndPwdSet устанавливается в значение «истина» (блок №22); вызывается обработчик события AppRoleTreatment (блок №23), позволяющего передать в экземпляр объекта контекста базы данных информацию об успешности выполнения запроса по установлению роли приложения, а также информацию о полученных дополнительных правах, позволяющих просматривать образцы и анализы, созданные другими пользователями. Экземпляр объекта контекста базы данных возвращается в главное окно приложения (блок №18), где также предпринимается попытка прочесть строку в таблице CalibrationType (блок №19). При возникновении ошибки (блок №20) система инициирует закрытие приложения.

Рисунок 8 – Алгоритм аутентификации пользователя

### 4.4 Определение расчетной схемы

По умолчанию расчетная схема для каждого анализа установлена в «Хлоридный». Расчет коэффициентов, позволяющих определить расчетную схему проводится в соответствии с методикой, изложенной в разделе 2.1 Методика определения принадлежности солевых образцов к соответствующему типу» настоящей пояснительной записки.

Алгоритм определения расчетной схемы приведен на листе графической части РТДП 006032.121.03.

В начале происходит вычисление коэффициента по формуле (5); значение коэффициента сравнивается с 1. Если оно больше или равно 1, схема принимается равной «Карбонатная», происходит возврат из метода.

Если значение коэффициента меньше 1, вычисляются значения коэффициентов и по формулам (6) и (8) соответственно; значение сравнивается затем с 1. Равенство значения 1 или большее означает тип расчетной схемы «Сульфатнонатриевая», сравнение значения с 1, позволяет определить подтип: < 1 – подтип I, в ином случае – подтип II.

Если же значение< 1, дополнительно рассчитывается по формуле (7); равенство его значения 1 либо большее, означает тип расчетной схемы «Сульфатномагниевая», дополнительное же сравнение с 1, позволяет определить подтип: < 1 – подтип I, в ином случае – подтип II.

Значение , меньшее 1, означает, что тип расчетной схемы – «Хлоридная».

### 4.5 Рекурсивное определение скорректированного сухого веса образца

На начальном этапе происходит вычисление скорректированных значений сухого магния и сухого веса образца в соответствии с формулами, приведенными в 2.3 Метод определения сухого веса образца». После этого происходит определение качества проведенной корректировки путем сравнения разности между скорректированным весом сухого магния и его весом до коррекции с величиной, определяющей толеранс (0.0001). Если качество корректировки оказывается достаточным, т.е. вышеупомянутая разность не превышает значения толеранса, осуществляется выход из метода с возвратом значения скорректированного сухого магния. Если же качество корректировки оказывается неудовлетворительным, осуществляется рекурсивный вызов метода, при этом в качестве параметров передаются скорректированные сухие веса образца и магния. Значение, которое возвращает рекурсивный метод, возвращается при выходе из родительского метода.

Рисунок 9 – Алгоритм определения скорректированного сухого веса образца

### 4.6 Определение коэффициентов корреляции калибровочной прямой

В цикле (блок №2) для каждой точки калибровки подсчитываются суммы концентрации (блок №3), показаний прибора (блок №4), квадратов концентрации (блок №5), произведения концентрации на показание прибора (блок №6), а также квадрата разности между значением показания прибора для текущей точки и средним значением показаний прибора (блок №7). Дельта определяется как разность между произведением суммы квадратов концентраций на количество точек и квадратом суммы концентраций (блок №8). Проверяется условие равенства значения дельты 0 (блок №9). При равенстве значения и устанавливаются равными максимальному значению типа decimal (тип decimal в C# не поддерживает работу с бесконечностью), а значение квадрата коэффициента корреляции принимается равным 0 (блок №12), работа метода завершается.

В случае, если значение дельта не равно нулю рассчитываются коэффициенты и (блоки №№10 и 11). Проверяется также равенство суммы, определенной в блоке №7, нулю (блок №13), при равенстве значение коэффициента корреляции устанавливается равным 0 (блок №4), работа метода завершается. После этого в цикле (блок №16) для каждой точки определяется сумма квадратов разностей между действительным показанием прибора для текущей точки и показанием прибора, определенным по рассчитанным параметра калибровочной прямой (блок №17). Близость отношения последней суммы к первой и есть показатель качества корреляции. Квадрат коэффициента корреляции рассчитывается как разность между 1 и отношением суммы, рассчитанной в блоке №17, к сумме рассчитанной в блоке № 7. Работа метода завершается.

Рисунок 10 – Алгоритм расчета параметров калибровочной кривой

### 4.7 Фильтрация образцов по дате отбора и лабораторным номерам

Фильтрация образцов необходима для сужения круга образцов, с которыми в данный момент работает пользователь, а также для осуществления поиска образца или нескольких образцов, находящихся в определенном временном интервале и имеющих определенный / определенные лабораторные номера.

Схема алгоритма фильтрации приведена на .Рисунок 11.

На первом этапе определяется наличие символов в части фильтра, отвечающей за отбор по лабораторному номеру. Предполагается, что пользователь будет вводить один или несколько лабораторных номеров, разделяя их точкой с запятой. При этом точка с запятой является «запрещенным» символом при вводе лабораторного номера – отсутствие точки с запятой есть одно из обязательных условий прохождения валидации при вводе лабораторного номера.

Если часть фильтра не пуста, то проверяется наличие в строке точки с запятой, как признака наличия нескольких лабораторных номеров. Если точка с запятой присутствует, производится «разделение» (Regex.Split) строки на несколько подстрок, в результате получается массив строк, содержащий лабораторные номера для поиска. После чего в цикле производится отбор образцов, попадающих в заданный временной интервал, лабораторные номера которых совпадают с лабораторными номерами в массиве. Отобранные образцы попадают в список для отображения и дальнейшей работы.

При отсутствии точки с запятой, в список для отображения отбираются образцы с лабораторным номером равным введенному в фильтре и попадающие в заданный временной интервал.

Если часть фильтра, отвечающего за лабораторный номер, пуста, в список для отображения отбираются образцы, попадающие в заданный временной интервал, без контроля лабораторного номера.

Введенный пользователем фильтр сохраняется в файле конфигурации системы, откуда он извлекается при старте программы.

Рисунок 11 – Алгоритм фильтрации образцов

### 4.8 Обновление данных калибровочной прямой

Для обновления калибровочной прямой в базе данных написана хранимая процедура «UpdateCalibrationData» принимающая в качестве параметра переменную табличного типа, атрибуты в которой полностью совпадают с атрибутами таблицы, хранящей данные калибровочной прямой. Однако поскольку Entity Framework не поддерживает работу с функциями и хранимыми процедурами, работающими с переменными табличного типа, определенными пользователем, выход был найден в использовании пакета расширения Entity Framework Extras (11).

Текс хранимой процедуры приведен в Приложение А  
Текст программы» в разделе. Алгоритм хранимой процедуры приведен на Рисунок 12. Хранимая процедура использует операцию MERGE языка T-SQL (12), предоставляющую возможность осуществить обновление набора строк посредством одной атомарной операции. Суть ее заключается в нахождении строк в целевой таблице и таблице-источнике (переменная табличного типа), совпадающих либо отличающихся по указанным параметрам. В данном случае в качестве параметра выступает первичный ключ IDCalibrationData в таблице CalibrationData и соответствующий ему атрибут в переменной табличного типа. При нахождении совпадающих строк осуществляется проверка на отличие в значениях атрибутов, хранящих концентрацию и показания прибора, а также идентификатора калибровки, при нахождении таких отличий производится обновление данных в соответствующей строке целевой таблицы, если же обновление не требуется — поиск продолжается дальше.

При нахождении строки целевой таблицы, для значения первичного ключа которой в таблице-источнике отсутствует «пара», производится проверка на совпадение идентификаторов калибровки у строк целевой таблицы и таблицы-источника, при совпадении значений идентификаторов такая строка удаляется из целевой таблицы.

При нахождении строки в таблице-источнике, для значения атрибута IDCalibrationData, соответствующего первичному ключу в целевой таблице, которой в целевой таблице отсутствует «пара», в целевую таблицу добавляется новая строка со значениями атрибутов, соответствующими значениям атрибутов строки в таблице источнике.

 Рисунок 12 – Алгоритм обновления данных калибровочной прямой

### 4.9 Сравнение результатов расчета

Для оценки сходимости двух «параллельных» анализов, либо для проверки результатов интеркалибровки проводится сравнение результатов расчета по установленной автоматически, либо выбранной пользователем схеме. Исходным условием для осуществления сравнения является наличие результатов расчета, а также выбор двух (и только двух) строк с результатами анализов.

Алгоритм метода, осуществляющего сравнение, приведен на Рисунок 13. Перед началом производится проверка на возможность проведения сравнения. Далее для каждого параметра в текущей расчетной схеме осуществляется проверка на предмет того, установлен пользователем в текущих настройках одно общее значение толеранса для всех параметров текущей схемы, либо для каждого параметра задано свое значение толеранса. Если для каждого параметра должно быть свое значение толеранса, осуществляется попытка его извлечения из настроек, сохраняемых в фале конфигурации приложения, при неудачной попытке толерансу присваивается значение по умолчанию.

Далее проводится работа с результатами расчета. Так как значения некоторых параметров могут отсутствовать (NULL), проводится проверка на наличие значений. При отсутствии значения хотя бы у одного из параметров, в «стек» сообщений добавляется сообщение о равенстве одного из параметров NULL. Сообщения из стека выводятся в качестве информации о результатах сравнения в диалоговом окне.

Затем результаты расчета по текущему параметру проверяются на равенство 0. В случае, если оба параметра равно 0, в стек добавляется сообщение об удовлетворительном результате сравнения цикл сравнения продолжается Если только один из параметров оказывает равным 0, а второй нет, в стек добавляется сообщение о превышении толеранса по данному параметру, цикл сравнения продолжается для дургих параметров. Далее вычисляется дельта как отношение модуля разности значений параметров к большему из них и значение дельты сравнивается с толерансом, при превышении дельты над значением толеранса, в стек добавляется сообщение о превышении, в противном случае – об удовлетворительном результате сравнения.

По окончании цикла пользователю в диалоговом окне выводится информация, содержащая сообщения о результатах сравнения по всем параметрам в текущей расчетной схеме.

Рисунок 13 – Алгоритм сравнения результатов расчета

# 5 Тестирование приложения

### 5.1 Подготовка к тестированию приложения

Тестирование программного средства осуществлялось по методу серого ящика с использованием фреймворка TestStack.White (13) и встроенных в Visual Studio средств для создания модельных тестов. Осуществлялась проверка работы приложения на соответствие функциональным требованиям. Проверке подвергались следующие функции:

- авторизация;

- вывод списка образцов;

- добавление нового образца;

- просмотр параметров образца;

- редактирование параметров образца;

- удаление образца;

- просмотр списка анализов;

- добавление анализа;

- просмотр данных анализа;

- удаление анализа;

- изменение данных анализа;

- задание расчетной схемы анализа;

- расчет данных анализа;

- сравнение результатов расчета;

- печать результатов расчета;

- оценка качестве результатов расчета;

- экспорт результатов расчета;

- просмотр списка калибровок;

- создание калибровки;

- удаление калибровки;

- изменение параметров калибровки;

- изменение данных калибровки;

- просмотр графика калибровочной прямой;

- определение концентрации по показаниям приборов;

- просмотр начальных параметров;

- изменение начальных параметров;

Разработаны тест-кейсы, приведенные в Таблица 5.

Таблица 5 – Тест-кейсы для тестирования программного средства

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Модуль | № Функции | Название и шаги воспроизведения | Ожидаемый результат | Статус |
| CA-01-01 | Авторизация | 5.1 | **Авт\_позитивый**   1. Запустить приложение 2. Ввести действительный логин 3. Ввести действительный пароль 4. Нажать кнопку «ОК» 5. Закрыть приложение | 1. После окна-заставки появится окно авторизации 2. Логин отобразится в поле «Имя пользователя» 3. Пароль отобразится в поле «Пароль» символами •. Станет доступной кнопка «ОК» Исчезнет текст «Пароль от 3 символов» 4. Загрузится главное окно программы 5. Приложение закроется | ОК |
| CA-01-02 | Авторизация | 5.1 | **Авт\_неверный\_пароль**   1. Запустить приложение 2. Ввести действительный логин 3. Ввести три любых символа 4. Нажать кнопку «ОК» 5. Закрыть приложение | 1. После окна-заставки появится окно авторизации 2. Логин отобразится в поле «Имя пользователя» 3. Пароль отобразится в поле «Пароль» символами •. Станет доступной кнопка «ОК» Исчезнет текст «Пароль от 3 символов» 4. Сообщение об ошибке в диалоговом окне 5. Приложение закроется | ОК |

Продолжение Таблицы 5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Модуль | № Функции | Название и шаги воспроизведения | Ожидаемый результат | Статус |
| CA-01-03 | Авторизация | 5.1 | **Авт\_3\_неверных\_пароля**   1. Запустить приложение 2. Ввести действительный логин 3. Ввести три любых символа 4. Нажать кнопку «ОК» 5. На предложение продолжить ответить «Да» 6. Шаги 2, 3 и 4 повторить 2 раза 7. Нажать «ОК» | 1. После окна-заставки появится окно авторизации 2. Логин отобразится в поле «Имя пользователя» 3. Пароль отобразится в поле «Пароль» символами •. Станет доступной кнопка «ОК» Исчезнет текст «Пароль от 3 символов» 4. Сообщение об ошибке в диалоговом окне 5. Снова появится окно авторизации 6. После третьей попытки появится сообщение об ошибке и невозможности продолжать работ без авторизации 7. Приложение закроется | ОК |
| CA-02-01 | Образцы | 1.1 | **Обр\_список\_позитивый**   1. Шаги 1-4 из CA-01-01 2. Выбрать меню «Образец» 3. Выбрать подпункт «Список…» 4. Нажать кнопку «Загрузить…» 5. Закрыть приложение | 1. Результаты шагов 1-4 из CA-01-01 2. Раскроется меню «Список» 3. Отобразится окно «Список образцов» 4. Будут загружены образцы 5. Приложение закроется | ОК |

Продолжение Таблицы 5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Модуль | № Ф-ции | Название и шаги воспроизведения | Ожидаемый результат | Статус |
| CA-02-02 | Образцы | 1.1 | **Обр\_список\_негативый**   1. Шаги 1-4 из CA-01-01[[1]](#footnote-1) 2. Выбрать меню «Образец» 3. Выбрать подпункт «Список…» 4. Нажать кнопку «Загрузить…» 5. Нажать «ОК» 6. Закрыть приложение | 1. Результаты шагов 1-4 из CA-01-01 2. Раскроется меню «Список» 3. Отобразится окно «Список образцов» 4. Отобразится сообщение об ошибке 5. Диалоговое окно закроется 6. Приложение закроется | ОК |
| CA-02-03 | Образцы | 1.2 | **Обр\_фильтр\_списка**   1. Шаги 1-3 из CA-02-01 2. Вызвать контекстное меню в поле с параметрами фильтра 3. Выбрать подпункт «Сбросить фильтр» 4. Нажать кнопку «Фильтр…» 5. Установить поле «по:» минус месяц от текущей даты 6. Нажать кнопку «ОК» 7. Закрыть приложение | 1. Результаты шагов 1-3 из CA-02-01 2. Раскроется контекстное меню 3. Значение фильтра будет установлено от текущей даты минус один год; фильтрации по лабораторному номеру нет. Загрузится список образцов. 4. Отобразится окно установки параметров фильтра 5. Значение поля будет установлено 6. Окно установки параметров фильтра закроется. Кол-во образцов уменьшится. 7. Приложение закроется | ОК |

Продолжение Таблицы 5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Модуль | № Ф-ции | Название и шаги воспроизведения | Ожидаемый результат | Статус |
| CA-02-04 | Образцы | 1.3 | **Обр\_добавить**   1. Шаги 1-4 из CA-02-01 2. Нажать кнопку «Добавить…» 3. В поле «лабораторный номер» внести случайное текстовое значение 4. В поле «Описание» ввести случайное текстовое значение 5. Нажать кнопку «ОК» 6. Закрыть приложение | 1. Результаты шагов 1-4 CA-02-01 2. Отобразится окно «Внести новый образец» 3. В поле отобразится введенное значение 4. В поле отобразится введенное значение 5. В список образцов будет добавлен текущий образец 6. Приложение закроется | OK |
| CA-02-05 | Образцы | 1.6 | **Обр\_удалить**   1. Шаги 1-4 из CA-02-01 2. Выбрать последний в списке образец 3. Нажать кнопку «Удалить» 4. Нажать «ОК» 5. Закрыть приложение | 1. Результаты шагов 1-4 CA-02-01 2. Образец будет выделен 3. Запрос на подтверждение удаления 4. Образец будет удален 5. Приложение закроется | OK |

Продолжение Таблицы 5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Модуль | № Ф-ции | Название и шаги воспроизведения | Ожидаемый результат | Статус |
| CA-02-06 | Образцы | 1.6 | **Обр\_просмотр\_параметров**   1. Шаги 1-4 из CA-02-01 2. Выбрать последний образец 3. Дважды кликнуть на образце 4. Закрыть приложение | 1. Результаты шагов 1-4 CA-02-01 2. Образец будет выделен 3. Откроется окно с параметрами выбранного образца 4. Приложение закроется | OK |
| CA-02-07 | Образцы | 1.5 | **Обр\_изменить\_параметры**   1. Шаги 1-4 из CA-02-01 2. Выбрать последний образец 3. Дважды кликнуть на образце 4. В поле «Описание» ввести случайную строку 5. В поле «Дата отбора» ввести позавчерашнее число 6. Нажать кнопку «ОК» 7. Закрыть приложение | 1. Результаты шагов 1-4 CA-02-01 2. Образец будет выделен 3. Откроется окно с параметрами 4. Строка отобразится в поле «Описание» 5. Дата отбора установится на позавчерашнее число 6. В списке образцов отобразится образец с измененными данными 7. Приложение закроется | OK |

Продолжение Таблицы 5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Модуль | № Ф-ции | Название и шаги воспроизведения | Ожидаемый результат | Статус |
| CA-02-08 | Образцы | 5.2 | **Обр\_фильтр\_данных**   1. Шаги 1-4 из CA-02-01 2. Выбрать последний образец 3. Дважды кликнуть на образце 4. Запомнить параметры. 5. Закрыть окно с параметрами 6. Закрыть окно «Список образцов» 7. В меню «Файл» выбрать подпункт «Сменить пользователя» 8. Ввести логин и пароль для другого пользователя 9. В меню «Образец» выбрать подпункт «Список…» 10. Нажать кнопку «Загрузить» 11. Закрыть приложение | 1. Результаты шагов 1-4 CA-02-01 2. Образец будет выделен 3. Откроется окно с параметрами. 4. Параметры будут сохранены 5. Окно закроется 6. Окно закроется 7. Откроется окно авторизации 8. Окно авторизации закроется. Отсутствуют сообщения об ошибках. 9. Откроется окно «Список образцов» 10. Список образцов будет загружен; образец, выбранный в п.3 будет отсутствует в списке образцов 11. Приложение закроется | OK |

Продолжение Таблицы 5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Модуль | № Ф-ции | Название и шаги воспроизведения | Ожидаемый результат | Статус |
| CA-03-01 | Анализы | 2.1 | **Анализ\_добавить**   1. Шаги 1-2 из CA-02-08 2. На образце вызвать контекстное меню 3. Запомнить доступность подпункта «Редактировать данные анализов». 4. Выбрать подпункт «Ввести данные новых анализов» 5. Дважды кликнуть на строке анализа 6. Ввести случайную строку 7. Нажать «ОК» 8. Нажать «ОК» 9. Выбрать тот же образец 10. Вызвать контекстное меню 11. Выбрать подпункт «Редактировать данные анализов» 12. Нажать кнопку «Отмена» 13. Закрыть приложение | 1. Результаты шагов 1-2 CA-02-08 2. Контекстное окно откроется 3. Доступность подпункта будет сохранена 4. Откроется окно добавления анализов к образцу 5. Откроется окно ввода описания анализа. 6. Строка отобразится в поле описания 7. Окно ввода описания закроется 8. Окно редактирования закроется 9. Образец будет выделен 10. Контекстное меню отобразится; подпункт «Редактировать данные анализов» будет доступен. 11. Откроется окно редактирования анализов 12. Окно редактирования закроется 13. Приложение закроется |  |

Продолжение Таблицы 5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Модуль | № Ф-ции | Название и шаги воспроизведения | Ожидаемый результат | Статус |
| CA-03-02[[2]](#footnote-2) | Анализы | 2.2 | **Анализ\_просмотреть**   1. Шаги 1-2 из CA-02-08 2. На образце вызвать контекстное меню 3. Выбрать подпункт «Редактировать данные анализов» 4. Нажать кнопку «Отмена» 5. Закрыть приложение | 1. Результаты шагов 1-2 CA-02-08 2. Контекстное окно откроется 3. Откроется окно редактирования анализов 4. Окно редактирования закроется 5. Приложение закроется |  |
| CA-03-03 | Анализы | 2.3 | **Анализ\_изменить**   1. Шаги 1-3 из CA-03-02 2. Выбрать последний анализ 3. Изменить значение даты анализа на два дня назад 4. Нажать кнопку «ОК» 5. Выбрать последний образец 6. На образце вызвать контекстное меню 7. Выбрать подпункт «Редактировать данные анализов» 8. Выбрать дату у последнего анализа 9. Закрыть приложение | 1. Результаты шагов 1-3 CA-03-02 2. Строка анализа выделится 3. Значение даты изменится 4. Окно редактирования закроется 5. Образец выделится 6. Контекстное окно откроется 7. Откроется окно редактирования анализов 8. Дата будет равна установленной в пункте 3 9. Приложение закроется |  |

Продолжение Таблицы 5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Модуль | № Ф-ции | Название и шаги воспроизведения | Ожидаемый результат | Статус |
| CA-03-04 | Анализы | 2.4 | **Анализ\_удалить**   1. Шаги 1-3 из CA-03-02 2. Выбрать последний анализ 3. Нажать DELETE 4. Подтвердить удаление 5. Нажать ESC 6. Выбрать последний образец 7. На образце вызвать контекстное меню 8. Выбрать подпункт «Редактировать данные анализов» 9. Закрыть приложение | 1. Результаты шагов 1-3 CA-03-02 2. Строка анализа выделится 3. Появится окно подтверждения удаления 4. Анализ будет удален 5. Окно редактирования закроется 6. Образец выделится 7. Контекстное окно откроется 8. Откроется окно редактирования; анализ, удаленного в пункте 3 не будет в списке 9. Приложение закроется | ОК |
| CA-03-05 | Анализы | 2.5 | **Анализ\_рассчитать**   1. Шаги 1-3 из CA-03-02 2. Выбрать последний анализ 3. Нажать ALT + C 4. Закрыть приложение | 1. Результаты шагов 1-3 CA-03-02 2. Строка анализа выделится 3. Появится окно, сообщающее о расчете анализа 4. Приложение закроется | ОК |
| CA-03-06 | Анализы | 2.6 | **Анализ\_изменить\_схему**   1. Шаги 1-3 из CA-03-02 2. Выбрать последний анализ 3. Выбрать схему расчета «Карбонатная» 4. Закрыть приложение | 1. Результаты шагов 1-3 CA-03-02 2. Строка анализа выделится 3. Схема будет установлена 4. Приложение закроется | ОК |

Продолжение Таблицы 5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Модуль | № Ф-ции | Название и шаги воспроизведения | Ожидаемый результат | Статус |
| CA-03-07 | Анализы | 2.7 | **Анализ\_сравнить\_расчет**   1. Шаги 1-2 из CA-03-02 2. Выбрать подпункт «Ввести данные новых анализов» - 2 шт. 3. Выделить оба анализа 4. Нажать ALT + C 5. Закрыть сообщение 6. Выделить оба анализа 7. Открыть контекстное меню 8. Выбрать пункт «Сравнить» 9. Закрыть приложение | 1. Результаты шагов 1-2 CA-03-02 2. Откроется окно редактирования с 2 анализами 3. Оба анализа выделятся 4. Сообщение о расчете 2 анализов 5. Сообщение закроется 6. Оба анализа выделятся 7. Контекстное меню откроется 8. Сообщение о результатах сравнения 9. Приложение закроется | ОК |
| CA-03-08 | Анализы | 2.8 | **Анализ\_печать\_результатов**   1. Шаги 1-6 из CA-03-07 2. Нажать кнопку «Печать» 3. Закрыть приложение | 1. Результаты шагов 1-6 CA-03-07 2. Отобразится окно предварительного просмотра 3. Приложение закроется | ОК |
| CA-03-09 | Анализы | 2.9 | **Анализ\_экспорт\_результатов**   1. Шаги 1-2 из CA-03-08 2. Вызвать контекстное меню 3. Выбрать пункт «Сохранить как» 4. Задать имя файла 5. Нажать «Сохранить» 6. Закрыть приложение | 1. Результаты шагов 1-2 CA-03-08 2. Отобразится контекстное меню 3. Диалоговое окно для задания имени файла отобразится 4. Имя файла отобразится 5. Диалог закроется 6. Приложение закроется | ОК |

Продолжение Таблицы 5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Модуль | № Ф-ции | Название и шаги воспроизведения | Ожидаемый результат | Статус |
| CA-03-10 | Анализы | 2.10 | **Анализ\_оценка\_результатов**   1. Шаги 1-4 из CA-03-07 2. Закрыть сообщение 3. Закрыть приложение | 1. Результаты шагов 1-4 CA-03-07 2. Сообщение закроется. В поле дополнительной информации появятся результаты оценки качества расчетов 3. Приложение закроется | ОК |
| CA-04-01 | Калибровки | 3.1 | **Калибровки\_список**   1. Шаги 1-4 из CA-01-01 2. Выбрать пункт меню «Настройки» 3. Выбрать подпункт «Калибровки» 4. Выбрать подпункт «Калий» 5. Закрыть приложение | 1. Результаты шагов 1-4 CA-01-01 2. Пункт меню раскроется 3. Подпункт раскроется 4. Отобразится окно выбора калибровок 5. Приложение закроется | ОК |
| CA-04-02 | Калибровки | 3.2 | **Калибровки\_новая**   1. Шаги 1-4 из CA-04-01 2. Нажать кнопку «Новая» 3. Ввести по две отличающиеся точки в каждый диапазон 4. Ввести случайное описание 5. Нажать кнопку «ОК» 6. Раскрыть список калибровок 7. Закрыть приложение | 1. Результаты шагов 1-4 CA-04-01 2. Отобразится окно данных калибровки 3. Данные точек отобразятся в таблицах соответствующих диапазонов 4. Описание отобразится в поле описания 5. Окно данных закроется 6. Список раскроется; созданная калибровка присутствует в списке 7. Приложение закроется | ОК |

Продолжение Таблицы 5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Модуль | № Ф-ции | Название и шаги воспроизведения | Ожидаемый результат | Статус |
| CA-04-03 | Калибровки | 3.3 | **Калибровки\_удалить**   1. Шаги 1-4 из CA-04-01 2. Из выпадающего списка выбрать калибровку 3. Нажать кнопку «Удалить» 4. Подтвердить удаление 5. Закрыть приложение | 1. Результаты шагов 1-4 CA-04-01 2. Калибровка отобразится в выпадающем списке 3. Отобразится окно подтверждения 4. Калибровка пропадет из выпадающего списка 5. Приложение закроется | ОК |
| CA-04-04 | Калибровки | 3.4 | **Калибр\_изменить\_описание**   1. Шаги 1-4 из CA-04-01 2. Из выпадающего списка выбрать калибровку 3. Нажать кнопку «Изменить» 4. Изменить описание на случайное 5. Нажать кнопку «ОК» 6. Закрыть приложение | 1. Результаты шагов 1-4 CA-04-01 2. Калибровка отобразится в выпадающем списке 3. Отобразится окно редактирования калибровки 4. Описание отобразится в поле 5. Окно редактирования закроется. Калибровка с новым описанием отобразится в раскрывающемся списке. 6. Приложение закроется | ОК |
| CA-04-05 | Калибровки | 3.5 | **Калибр\_данные\_просмотр**   1. Шаги 1-3 из CA-04-04 2. Закрыть приложение | 1. Результаты шагов 1-3 CA-04-04 2. Приложение закроется | ОК |

Продолжение Таблицы 5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Модуль | № Ф-ции | Название и шаги воспроизведения | Ожидаемый результат | Статус |
| CA-04-06 | Калибровки | 3.6 | **Калибр\_изменить\_данные**   1. Шаги 1-4 из CA-04-01 2. Из выпадающего списка выбрать калибровку 3. Нажать кнопку «Изменить» 4. Изменить значение концентрации в первом диапазоне 5. Нажать кнопку «ОК» 6. Закрыть приложение | 1. Результаты шагов 1-4 CA-04-01 2. Калибровка отобразится в выпадающем списке 3. Отобразится окно редактирования калибровки 4. Описание отобразится в поле 5. Окно редактирования закроется 6. Приложение закроется | ОК |
| CA-04-07 | Калибровки | 3.7 | **Калибр\_просмотр\_графика**   1. Шаги 1-4 из CA-04-01 2. Из выпадающего списка выбрать калибровку 3. Нажать кнопку «Просмотр» 4. Закрыть приложение | 1. Результаты шагов 1-4 CA-04-01 2. Калибровка отобразится в выпадающем списке 3. Отобразится окно графического представления калибровочных прямых 4. Приложение закроется | ОК |

Продолжение Таблицы 5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Модуль | № Ф-ции | Название и шаги воспроизведения | Ожидаемый результат | Статус |
| CA-05-01 | Параметры | 4.1 | **Параметры\_просмотр**   1. Шаги 1-4 из CA-01-01 2. Выбрать пункт меню «Настройки» 3. Выбрать подпункт «Исходные данные…» 4. Выбрать подпункт «Для анализа солей» 5. Закрыть приложение | 1. Результаты шагов 1-4 CA-01-01 2. Пункт меню раскроется 3. Подпункт раскроется 4. Отобразится окно параметров 5. Приложение закроется | ОК |
| CA-05-02 | Параметры | 4.2 | **Параметры\_изменить**   1. Шаги 1-4 из CA-01-01 2. Выбрать пункт меню «Настройки» 3. Выбрать подпункт «Исходные данные…» 4. Выбрать подпункт «Для анализа солей» 5. В поле «Холостой Br» ввести положительное число 6. Закрыть приложение | 1. Результаты шагов 1-4 CA-01-01 2. Пункт меню раскроется 3. Подпункт раскроется 4. Отобразится окно параметров 5. Число отобразится в поле «Холостой Br» 6. Приложение закроется | ОК |

### Оценка результатов тестирования

По результатам тестирования можно заключить, что программная система отвечает предъявляемым требованиям и реализует необходимые для работы функции.

# Заключение

В ходе дипломного проектирования разработано программное средство расчета результатов химических анализов.

В дальнейшем разработанное программное средство планируется использовать в работе Филиала «Центральная лаборатория» ГП «НПЦ по геологии». При необходимости функционал средства будет расширен за счет добавления новых типов анализа, либо за счет введения модифицированных схем расчета.

# Список использованных источников

1. **Microsoft Corp.** Row-Level Security. *MSDN.* [В Интернете] [Цитировано: 29 August 2018 г.] https://docs.microsoft.com/en-us/sql/relational-databases/security/row-level-security.

2. **Морачевский Ю.В., Петрова Е.М. (ред.).** *Методы анализа рассолов и солей.* Москва-Ленинград : Химия, 1964.

3. **Microsoft Corp.** SQL Server 2016 Express LocalDB. *MSDN.* [В Интернете] [Цитировано: 26 July 2018 г.] https://docs.microsoft.com/en-us/sql/database-engine/configure-windows/sql-server-2016-express-localdb.

4. —. Database First. *MSDN.* [В Интернете] [Цитировано: 1 09 2018 г.] https://docs.microsoft.com/en-us/ef/ef6/modeling/designer/workflows/database-first.

5. —. Code First to a New Database. *MSDN.* [В Интернете] [Цитировано: 1 September 2018 г.] https://docs.microsoft.com/en-us/ef/ef6/modeling/code-first/workflows/new-database.

6. —. ALTER TABLE computed\_column\_definition (Transact-SQL). *MSDN.* [В Интернете] [Цитировано: 03 September 2018 г.] https://docs.microsoft.com/en-us/sql/t-sql/statements/alter-table-computed-column-definition-transact-sql.

7. —. Свойство Application.StartupUri. *MSDN.* [В Интернете] [Цитировано: 3 September 2018 г.] https://msdn.microsoft.com/ru-ru/library/system.windows.application.startupuri.

8. —. Событие Window.SourceInitialized. *MSDN.* [В Интернете] [Цитировано: 04 September 2018 г.] https://msdn.microsoft.com/ru-ru/library/system.windows.window.sourceinitialized.

9. —. Logging and intercepting database operations. *MSDN.* [В Интернете] [Цитировано: 04 September 2018 г.] https://docs.microsoft.com/en-us/ef/ef6/fundamentals/logging-and-interception.

10. —. sp\_setapprole (Transact-SQL). *MSDN.* [В Интернете] [Цитировано: 04 September 2018 г.] https://docs.microsoft.com/en-us/sql/relational-databases/system-stored-procedures/sp-setapprole-transact-sql.

11. **Fodsuk.** EntityFrameworkExtras.EF6. *NuGet.* [В Интернете] [Цитировано: 06 September 2018 г.] https://www.nuget.org/packages/EntityFrameworkExtras.EF6/.

12. **Microsoft Corp.** MERGE (Transact-SQL). *MSDN.* [В Интернете] [Цитировано: 6 September 2018 г.] https://docs.microsoft.com/en-us/sql/t-sql/statements/merge-transact-sql.

13. **White.** TestStack.White. *ReadTheDocs.* [В Интернете] [Цитировано: 9 September 2018 г.] https://teststackwhite.readthedocs.io/en/latest/.

# Приложение А (обязательное) Листинги

A.1 Класс приложения, отвечающий за запуск главного окна, а также содержащий ресурсы, используемые в других окнах приложения

using System.Windows;

using System.Diagnostics;

using System.Threading;

using System;

using System.Configuration;

using System.IO;

using System.Reflection;

using System.Data.SqlClient;

using System.Reflection;

using System.Runtime.InteropServices;

namespace ChemicalAnalyses

{public partial class App : Application

{

private const int MINIMUM\_SPLASH\_TIME = 1500; // Miliseconds

// Use a name unique to the application (including GUID)

private static Mutex mutex = new Mutex(false,

@"ZVV\_Diploma\_Mutex/{DF776A4B-389C-4A4F-AD0B-1BE989F11ED9}", out mutexIsCreated);

private static bool mutexIsCreated;

protected override void OnStartup(StartupEventArgs e)

{

if (!mutex.WaitOne(TimeSpan.FromSeconds(1), false))

{

MessageBox.Show("Запущена еще одна копия приложения.", "Ошибка",

MessageBoxButton.OK, MessageBoxImage.Exclamation);

//And post message (WM\_SHOWME) to the running instance

NativeMethods.PostMessage((IntPtr)NativeMethods.HWND\_BROADCAST,

NativeMethods.WM\_SHOWME, IntPtr.Zero, IntPtr.Zero);

Shutdown(-2);

return;

}

// check for config file presence

Configuration config;

try

{

Uri UriAssemblyFolder = new Uri(Path.GetDirectoryName(Assembly.GetExecutingAssembly().GetName().CodeBase));

string appPath = UriAssemblyFolder.LocalPath;

config = ConfigurationManager.OpenExeConfiguration(appPath + @"\ChemicalAnalyses.exe");

ClientSettingsSection elementsWeights = (ClientSettingsSection)config.SectionGroups["applicationSettings"].Sections[0];

elementsWeights.Settings.Get("Mg");

StartupUri = new Uri("MainWindow.xaml", UriKind.Relative);

catch

{config file doesn't exist...

MessageBox.Show("Файл конфигурации не найден. \n Программа не может работать!",

"Ошибка файла конфигурации!", MessageBoxButton.OK, MessageBoxImage.Exclamation);

Shutdown(-15);

return;

SplashScreen splash = new SplashScreen("SASplashScreen.png");

splash.Show(false);

// Step 2 - Start a stop watch

Stopwatch timer = new Stopwatch();

timer.Start();

// Step 3 - Load your windows but don't show it yet

base.OnStartup(e);

MainWindow main = new MainWindow();

timer.Stop();

int remainingTimeToShowSplash = MINIMUM\_SPLASH\_TIME - (int)timer.ElapsedMilliseconds;

if (remainingTimeToShowSplash > 0) //if the loading took less time than was planned to show splash

Thread.Sleep(remainingTimeToShowSplash); //sleep a little bit more

splash.Close(TimeSpan.FromSeconds(1)); //one second fade away

protected override void OnExit(ExitEventArgs e)

{

base.OnExit(e);

try { if (mutexIsCreated) mutex.ReleaseMutex(); }

catch (Exception ex){ }

}

}

internal class NativeMethods

{

public const int HWND\_BROADCAST = 0xffff;

public static readonly int WM\_SHOWME = RegisterWindowMessage("WM\_SHOWME");

[DllImport("user32")]

public static extern bool PostMessage(IntPtr hwnd, int msg, IntPtr wparam, IntPtr lparam);

[DllImport("user32")]

public static extern int RegisterWindowMessage(string message);

[DllImport("user32.dll")]

public static extern bool ShowWindow(IntPtr hWnd, int cmdShow);}}

<Application x:Class="ChemicalAnalyses.App"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" ShutdownMode="OnExplicitShutdown">

<Application.Resources>

<ControlTemplate x:Key="ErrorTemplate1">

<StackPanel Orientation="Horizontal">

<Border BorderThickness="2" BorderBrush="Red"/>

<AdornedElementPlaceholder x:Name="placeholder"/>

<TextBlock Foreground="Red"

FontSize="{Binding RelativeSource={x:Static RelativeSource.Self}, Path=FontSize}">\*</TextBlock>

</StackPanel>

</ControlTemplate>

<Style TargetType="{x:Type TextBlock}">

<Setter Property="Validation.ErrorTemplate" Value="{StaticResource ErrorTemplate1}"/>

<Style.Triggers>

<Trigger Property="Validation.HasError" Value="True">

<Setter Property="ToolTip"

Value="{Binding (Validation.Errors)[0].ErrorContent, RelativeSource={x:Static RelativeSource.Self}}"/>

<Setter Property="Background" Value="Pink"/>

</Trigger>

</Style.Triggers>

</Style>

<Style TargetType="{x:Type TextBox}">

<Setter Property="Validation.ErrorTemplate">

<Setter.Value>

<ControlTemplate>

<Grid>

<Border BorderBrush="#FFCB2E2E" BorderThickness="1" Background="#11FF0000" IsHitTestVisible="False" x:Name="errorBorder"/>

<AdornedElementPlaceholder x:Name="placeholder" />

<Popup AllowsTransparency="True" HorizontalAlignment="Right" HorizontalOffset="0" VerticalOffset="0" PopupAnimation="Fade" Placement="Right"

PlacementTarget="{Binding ElementName=errorBorder}" IsOpen="{Binding ElementName=placeholder, Path=AdornedElement.IsFocused, Mode=OneWay}">

<StackPanel Orientation="Horizontal">

<Polygon VerticalAlignment="Center" Points="0,4 4,0 4,8" Fill="#FFCB2E2E" Stretch="Fill" Stroke="#FFCB2E2E" StrokeThickness="2" />

<Border Background="#FFCB2E2E" CornerRadius="4" Padding="4">

<TextBlock HorizontalAlignment="Center" Foreground="White" FontWeight="Bold" Margin="2,0,0,0"

Text="{Binding ElementName=placeholder, Path=AdornedElement.ToolTip, Mode=OneWay}" />

</Border>

</StackPanel>

</Popup>

</Grid>

</ControlTemplate>

</Setter.Value>

</Setter>

<Style.Triggers>

<Trigger Property="Validation.HasError" Value="True">

<Setter Property="ToolTip" Value="{Binding RelativeSource={RelativeSource Self}, Path=(Validation.Errors)[0].ErrorContent}" />

</Trigger>

</Style.Triggers>

</Style>

<!-- Pictures -->

<Viewbox x:Key="icHelpApplication" Width="16" Height="16">

<Rectangle Width="16" Height="16">

<Rectangle.Fill>

<DrawingBrush>

<DrawingBrush.Drawing>

<DrawingGroup>

<DrawingGroup.Children>

<GeometryDrawing Brush="#00FFFFFF" Geometry="F1M16,16L0,16 0,0 16,0z" />

<GeometryDrawing Brush="#FFF6F6F6" Geometry="F1M16,1L0,1 0,15 1.703,15C2.473,15.616 3.437,16 4.5,16 5.563,16 6.527,15.616 7.297,15L16,15z" />

<GeometryDrawing Brush="#FF414141" Geometry="F1M1,2L1,8.703C1.288,8.343,1.616,8.017,2,7.76L2,5 14,5 14,13 8.725,13C8.599,13.354,8.445,13.693,8.24,14L15,14 15,2z" />

<GeometryDrawing Brush="#FF414141" Geometry="F1M6,11C6,11.55,5.55,12,5,12L5,12.5 4,12.5 4,12C4,11.449,4.45,11,5,11L5,10C4.92,9.96 4.748,9.898 4.5,9.898 4.252,9.898 4.08,9.956 4,10L4,10.5 3,10.5 3,10C3,9.42 3.631,9 4.5,9 5.369,9 6,9.42 6,10z M5,14L4,14 4,13 5,13z M4.5,8C2.567,8 1,9.567 1,11.5 1,13.433 2.567,15 4.5,15 6.433,15 8,13.433 8,11.5 8,9.567 6.433,8 4.5,8" />

<GeometryDrawing Brush="#FFF0EFF1" Geometry="F1M2,5L2,7.76C2.715,7.281 3.575,7 4.5,7 6.985,7 9,9.015 9,11.5 9,12.029 8.9,12.528 8.725,13L14,13 14,5z" />

<GeometryDrawing Brush="#FFF0EFF1" Geometry="F1M4.5,9C3.631,9,3,9.42,3,10L3,10.5 4,10.5 4,10C4.08,9.956 4.252,9.9 4.5,9.9 4.748,9.898 4.92,9.956 5,10L5,11C4.449,11,4,11.449,4,12L4,12.5 5,12.5 5,12C5.551,12,6,11.551,6,11L6,10C6,9.42,5.369,9,4.5,9" />

<GeometryDrawing Brush="#FFF0EFF1" Geometry="F1M4,14L5,14 5,13 4,13z" />

</DrawingGroup.Children>

</DrawingGroup>

</DrawingBrush.Drawing>

</DrawingBrush>

</Rectangle.Fill>

</Rectangle>

</Viewbox>

<Viewbox x:Key="icExit2" Stretch="Uniform">

<Canvas Canvas.Left="0" Canvas.Top="0" Width="1000" Height="1000">

<Canvas >

<Canvas>

<Canvas.RenderTransform>

<MatrixTransform Matrix="1 0 0 -1 0 1952"/>

</Canvas.RenderTransform>

<Path Fill="#000000">

<Path.Data>

<PathGeometry Figures="M745.6 1788.6c0 33.8-27.4 61.3-61.3 61.3c-33.8 0-61.3-27.4-61.3-61.3c0-33.9 27.4-61.3 61.3-61.3S745.6 1754.8 745.6 1788.6z M928.6 1134.5l0 719.4c0 89.3-93 88.2-93 88.2l-524.4-0.5c-96.9 0-85.9-90.2-85.9-90.2v-552.9h60.4v550.4c0 33.2 28.1 31.7 28.1 31.7h523c31.3 0 31.6-29.2 31.6-29.2l-0.5-308l-31.8-0.2c0 0-85.5 49.5-118.5 97.4c0 0-27.9 55.6-132.8 55.6l-143.4 0.2c0 0-16.7-0.8-33.9-30.8c-2.4-3.7-85-139-85-139s-20.5-44.4 17.3-44.4c0 0 23-1.9 37.3 22.2l66.8 88.3c0 0 5.8 13.6 29.7 27.3l19.4 0.9c0 0 28.9 0.9 23.4-15.9c0 0-65.3-88-76.7-160.4l-0.2-152.7c0 0-1.8-20.6-23.4-24c-70.6 0-142.9 0.5-142.9 0.5s-48.2-0.5-48.2-47.8c0 0 1.6-44.3 47.5-44.3l214.7 0.8c34.1 0 27.8 86.2 27.8 86.2s15.6 120.2 36.1 89.4c0 0 100.8-188.8 135-249c0 0 12.7-34.2 53-34.2c0 0 36.5 11.4 36.5 38.7l-152.5 296.4v34.2l45.4 105.7c0 0 20.5 25.4 34.2 4.9c0 0 28.2-26.9 102.5-66.3l62-0.6l1.4-317.4c0-35.7-30.1-65.2-30.1-65.2l-45.3-57.2c-29-25.2-51-19.5-51-19.5l-502.9-0.9c-32.8 0 1.9 30 1.9 30l44.3 44v49.6h-66.3c-3.6-4-136-158.2-136-158.2c-33.2-28.1 10.2-25.5 10.2-25.5l692.1 0c15.3 0 45.4 26.6 45.4 26.6l49.4 61.9C937.2 1106.2 928.6 1134.5 928.6 1134.5L928.6 1134.5z" FillRule="NonZero"/>

</Path.Data>

</Path>

</Canvas>

</Canvas>

</Canvas>

</Viewbox>

<Viewbox x:Key="icnExitApplication" Width="16" Height="16">

<Rectangle Width="16" Height="16">

<Rectangle.Fill>

<DrawingBrush>

<DrawingBrush.Drawing>

<DrawingGroup>

<DrawingGroup.Children>

<GeometryDrawing Brush="#00FFFFFF" Geometry="F1M16,16L0,16 0,0 16,0z" />

<GeometryDrawing Brush="#FFF6F6F6" Geometry="F1M9.59,10L9,10.586 9,11.445C8.41,11.789 7.7319,12 7,12 4.8,12 3,10.21 3,8 3,5.79 4.7909,4 7,4 7.7319,4 8.41,4.211 9,4.555L9,5.414 9.59,6 8,6 8,10z M13.2279,4.813C12.07,2.551 9.7169,1 7,1 3.134,1 -10E-05,4.134 -10E-05,8 -10E-05,11.866 3.1339,15 7,15 9.7169,15 12.07,13.45 13.2279,11.187L16,8.414 16,7.586z" />

<GeometryDrawing Brush="#FF414141" Geometry="F1M7,13C4.238,13 2,10.762 2,8 2,5.238 4.238,3 7,3 8.12,3 9.14,3.38 9.973,4L11.463,4C10.365,2.775 8.775,2 7,2 3.686,2 1,4.687 1,8 1,11.313 3.7,14 7,14 8.8,14 10.365,13.225 11.463,12L9.973,12C9.14,12.62,8.118,13,7,13" />

<GeometryDrawing Brush="#FF414141" Geometry="F1M12,5L10,5 12,7 9,7 9,9 12,9 10,11 12,11 15,8z" />

</DrawingGroup.Children>

</DrawingGroup>

</DrawingBrush.Drawing>

</DrawingBrush>

</Rectangle.Fill>

</Rectangle>

</Viewbox>

<Viewbox x:Key="icnListOfSamples" Width="16" Height="16">

<Rectangle Width="16" Height="16">

<Rectangle.Fill>

<DrawingBrush>

<DrawingBrush.Drawing>

<DrawingGroup>

<DrawingGroup.Children>

<GeometryDrawing Brush="#00FFFFFF" Geometry="F1M16,16L0,16 0,0 16,0z" />

<GeometryDrawing Brush="#FFF6F6F6" Geometry="F1M16,14L16,0 1,0 1,14 9.703,14z" />

<GeometryDrawing Brush="#FFF6F6F6" Geometry="F1M1,0L1,5 -10E-05,5 -10E-05,16 12,16 12,14 16,14 16,0z" />

<GeometryDrawing Brush="#FF414141" Geometry="F1M2,1L2,5 3,5 3,4 6,4 6,5 7,5 7,4 10,4 10,5 11,5 11,4 14,4 14,12 12,12 12,13 15,13 15,1z" />

<GeometryDrawing Brush="#FF414141" Geometry="F1M10,14L2,14 2,7 10,7z M1,15L11,15 11,6 1,6z" />

<GeometryDrawing Brush="#FF414141" Geometry="F1M3,9L6,9 6,8 3,8z" />

<GeometryDrawing Brush="#FF414141" Geometry="F1M3,13L7,13 7,12 3,12z" />

<GeometryDrawing Brush="#FF414141" Geometry="F1M3,11L8,11 8,10 3,10z" />

<GeometryDrawing Brush="#FFF0EFF1" Geometry="F1M7,5L10,5 10,4 7,4z" />

<GeometryDrawing Brush="#FFF0EFF1" Geometry="F1M3,5L6,5 6,4 3,4z" />

<GeometryDrawing Brush="#FFF0EFF1" Geometry="F1M11,4L11,5 12,5 12,12 14,12 14,4z" />

<GeometryDrawing Brush="#FFF0EFF1" Geometry="F1M3,12L7,12 7,13 3,13z M3,10L8,10 8,11 3,11z M3,8L6,8 6,9 3,9z M2,14L10,14 10,7 2,7z" />

</DrawingGroup.Children>

</DrawingGroup>

</DrawingBrush.Drawing>

</DrawingBrush>

</Rectangle.Fill>

</Rectangle>

</Viewbox>

<Viewbox x:Key="icnKalium" Stretch="Uniform">

<Canvas Width="8.47" Height="8.47">

<Canvas Opacity="1">

<Canvas.RenderTransform>

<MatrixTransform Matrix="0.96 0 0 0.99 -30.77 -33"/>

</Canvas.RenderTransform>

<Path Fill="#FF000000" StrokeThickness="0.11">

<Path.Data>

<PathGeometry Figures="m 41 35.4 h -1.17 v 1.175 h -0.36 v -1.17 h -1.17 v -0.35 h 1.17 v -1.17 h 0.355 v 1.17 h 1.17 z m -3 7.1 h -1.3 l -3 -3.6 -0.7 0.85 v 2.7 h -1 v -8 h 1 v 4.18 l 3.59 -4.2 h 1.19 l -3.3 3.77 z" FillRule="NonZero"/>

</Path.Data>

</Path>

</Canvas>

</Canvas>

</Viewbox>

<Viewbox x:Key="icnNatrium" Stretch="Uniform">

<Canvas Width="32" Height="32">

<Canvas Opacity="1">

<Canvas.RenderTransform>

<MatrixTransform Matrix="0.59 0 0 1 -4 21.5"/>

</Canvas.RenderTransform>

<Path Fill="#FF000000">

<Path.Data>

<PathGeometry Figures="M 29.9 9.1 H 25 L 11.3 -16.9 V 9.1 H 7.7 V -20 H 13.7 26.3 3.8 V -20 h 3.6 z" FillRule="NonZero"/>

</Path.Data>

</Path>

<Path Fill="#FF000000">

<Path.Data>

<PathGeometry Figures="M 54.3 9.1 H 50.65 V 6.8 q -0.5 0.33 -1.3 0.9375 -0.82 0.6 -1.6 0.9375 -0.92 0.45 -2.1 0.74 -1.2 0.3 -2.8 0.3 -2.95 0 -5 -1.95 -2.1 -1.95 -2.1 -5 0 -2.5 1 -4 1 -1.5 3 -2.4 2 -0.9 4.8 -1.2 2.8 -0.3 6 -0.5 V -5.9 q 0 -1.25 -0.45 -2 -0.43 -0.8 -1.25 -1.3 -0.8 -0.45 -1.9 -0.6 -1 -0.15 -2.3 -0.15 -1.45 0 -3.2 0.4 -1.78 0.37 -3.7 1.1 h -0.2 v -3.7 q 1 -0.3 3.11 -0.65 2 -0.3 4 -0.35 2.3 0 4 0.4 1.72 0.37 3 1.23 1.23 0.9 1.875 2.3 0.65 1.4 0.65 3.5 z M 50.7 3.75 v -6 q -1.7 0.1 -4 0.3 -2.2 0.2 -3.6 0.6 -1.6 0.45 -2.6 1.4 -1 0.94 -1 2.6 0 1.88 1.1 2.8 1.1 0.9 3.5 0.9 1.9 0 3.5 -0.7 1.6 -0.8 3 -1.8 z" FillRule="NonZero"/>

</Path.Data>

</Path>

</Canvas>

<Canvas Opacity="1">

<Canvas.RenderTransform>

<MatrixTransform Matrix="0.53 0 0 0.445 -3 33"/>

</Canvas.RenderTransform>

<Path Fill="#FF000000" StrokeThickness="0.75">

<Path.Data>

<PathGeometry Figures="m 73 -68.8 h -7.81 v 8.1 h -2.38 v -8.1 h -7.8 v -2.4 h 7.8 v -8.1 h 2.39 v 8.12 h 7.8 z" FillRule="NonZero"/>

</Path.Data>

</Path>

</Canvas>

</Canvas>

</Viewbox>

<Viewbox x:Key="icnCalib" Stretch="Uniform">

<Canvas Width="8.4748697" Height="8.4748688">

<Canvas>

<Canvas.RenderTransform>

<TranslateTransform X="-50.6" Y="-58"/>

</Canvas.RenderTransform>

<Rectangle Canvas.Left="50.6" Canvas.Top="58" Width="8.39" Height="8.39" StrokeThickness="0.087" Stroke="#FF000000"/>

<Path StrokeThickness="0.26" Stroke="#FFFF0000" StrokeLineJoin="Miter" StrokeStartLineCap="Flat" StrokeEndLineCap="Flat">

<Path.Data>

<PathGeometry Figures="m 51.21 66 c 0.57 -1.98 1.14 -3.97 2.37 -5.24 1.24 -1.27 3.14 -1.83 5 -2.39" FillRule="NonZero"/>

</Path.Data>

</Path>

<Ellipse Canvas.Left="56.8" Canvas.Top="59.5" Width="0.4" Height="0.4" StrokeThickness="0.1" Stroke="#FF000000" StrokeMiterLimit="4"/>

<Ellipse Canvas.Left="52.5" Canvas.Top="60.2" Width="0.4" Height="0.4" StrokeThickness="0.1" Stroke="#FF000000" StrokeMiterLimit="4"/>

<Ellipse Canvas.Left="52" Canvas.Top="65" Width="0.4" Height="0.4" StrokeThickness="0.1" Stroke="#FF000000" StrokeMiterLimit="4"/>

<Ellipse Canvas.Left="52.7" Canvas.Top="62.9" Width="0.4" Height="0.4" StrokeThickness="0.1" Stroke="#FF000000" StrokeMiterLimit="4"/>

</Canvas>

</Canvas>

</Viewbox>

<Viewbox x:Key="icnInitData" Stretch="Uniform">

<Canvas Width="8.4748697" Height="8.4748688">

<TextBlock Text="0.1234567890" FontSize="2"/>

<TextBlock Text="0.1234567890" FontSize="2" Canvas.Left="5" Canvas.Top="2">

<TextBlock.RenderTransform>

<TransformGroup>

<ScaleTransform ScaleX="0.5"/>

<RotateTransform Angle="45"/>

</TransformGroup>

</TextBlock.RenderTransform>

</TextBlock>

</Canvas>

</Viewbox>

<Viewbox x:Key="icnInitDataSalts" Stretch="Uniform">

<Canvas Width="8.4748697" Height="8.4748688">

<TextBlock Text="NaCl=12" FontSize="2"/>

<TextBlock Text="Coeff=1" FontSize="2" Canvas.Left="5" Canvas.Top="2">

<TextBlock.RenderTransform>

<TransformGroup>

<ScaleTransform ScaleX="0.5"/>

<RotateTransform Angle="45"/>

</TransformGroup>

</TextBlock.RenderTransform>

</TextBlock>

<TextBlock Text="KCl=2" FontSize="2" Canvas.Left="2" Canvas.Top="7">

<TextBlock.RenderTransform>

<TransformGroup>

<ScaleTransform ScaleX="0.5"/>

<RotateTransform Angle="-45"/>

</TransformGroup>

</TextBlock.RenderTransform>

</TextBlock>

</Canvas>

</Viewbox>

</Application.Resources>

</Application>

A.2 Класс главного окна приложения

using ChemicalAnalyses.Alumni;

using ChemicalAnalyses.Dialogs;

using SA\_EF;

using SettingsHelper;

using System;

using System.Collections;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Configuration;

using System.Data;

using System.Data.Common;

using System.Data.Entity.Infrastructure;

using System.Data.SqlClient;

using System.Linq;

using System.Reflection;

using System.Text.RegularExpressions;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Input;

using System.Windows.Interop;

using System.Windows.Media;

namespace ChemicalAnalyses

{

public partial class MainWindow : Window

{

private Window wndThis = null;

private const int MaxAttempts = 3;

public bool IsAdmin

{

get { return (bool)GetValue(IsAdminProperty); }

set { SetValue(IsAdminProperty, value); }

}

public static readonly DependencyProperty IsAdminProperty =

DependencyProperty.Register("IsAdmin", typeof(bool), typeof(MainWindow), new PropertyMetadata(false));

public string WindowTitle { get; } = "Расчет химсостава образцов";

public MainWindow()

{

InitializeComponent();

wndThis = this;

DataContext = this;

}

private bool Authorize(bool relogin = false)

{

int attemptsCounter = 0;

while (attemptsCounter++ < MaxAttempts)

{

UserNamePwdDlg userDlg = new UserNamePwdDlg();

if (userDlg.ShowDialog() == true)

{

ChemicalAnalysesEntities.UserName = userDlg.UserName;

ChemicalAnalysesEntities.Password = userDlg.pbPassword.Password;

}

else

{

if (attemptsCounter < MaxAttempts)

{

MessageBoxResult res = MessageBox.Show("Неверные логин или пароль!" + Environment.NewLine

+ "Продолжить?", "Ошибка!!!", MessageBoxButton.YesNo, MessageBoxImage.Exclamation,

MessageBoxResult.Yes);

if (res == MessageBoxResult.No) return false;

continue;

}

return false;

}

try

{

using (var context = new ChemicalAnalysesEntities(relogin))

{ var t = context.Samples.FirstOrDefault(); }

}

catch (Exception ex)

{

if (ex.InnerException?.GetType() == typeof(SqlException)

&& ((string)(ex.InnerException?.Data["HelpLink.EvtID"])).Equals("53"))

{

MessageBoxResult res = MessageBox.Show("Невозможно подключиться к серверу БД!",

"Ошибка!!!", MessageBoxButton.OK, MessageBoxImage.Exclamation);

CALogger.WriteToLogFile("Невозможно подключиться к серверу БД!" + Environment.NewLine

+ ex.InnerException?.Message);

return false;

}

CALogger.WriteToLogFile("Не найдена БД" + ex.Message);

}

if (!ChemicalAnalysesEntities.AreUserNameAndPwdSet)

{

if (attemptsCounter < MaxAttempts)

{

MessageBoxResult res = MessageBox.Show("Неверные логин или пароль!" + Environment.NewLine

+ "Продолжить?", "Ошибка!!!", MessageBoxButton.YesNo, MessageBoxImage.Exclamation,

MessageBoxResult.Yes);

if (res == MessageBoxResult.No) return false;

continue;

}

return false;

}

IsAdmin = ChemicalAnalysesEntities.IsAdmin;

return true;

}

return false;

}

#region HoverToolTip Property

public object HoverToolTip

{

get { return (object)GetValue(HoverToolTipProperty); }

set { SetValue(HoverToolTipProperty, value); }

}

public static readonly DependencyProperty HoverToolTipProperty =

DependencyProperty.Register(nameof(HoverToolTip), typeof(object), typeof(MainWindow),

new PropertyMetadata(null));

#endregion HoverToolTip Property

private void Window\_PreviewMouseMove(object sender, MouseEventArgs e)

{

var element = Mouse.DirectlyOver as FrameworkElement;

HoverToolTip = GetTooltip(element);

}

protected static Object GetTooltip(FrameworkElement obj)

{

if (obj == null){ return null; }

else if (obj.ToolTip != null){ return obj.ToolTip; }

else{ return GetTooltip(VisualTreeHelper.GetParent(obj) as FrameworkElement); }

}

#region Commands

private void ExitCommand\_Executed(object sender, ExecutedRoutedEventArgs e)

{

Properties.Settings.Default.Save();

CALogger.WriteToLogFile("Закрытие программы");

Application.Current.Shutdown();

}

private void HelpCommand\_Executed(object sender, ExecutedRoutedEventArgs e)

{

MessageBox.Show("Дипломная работа по теме" +

"\n«Программное средство для расчета химического состава образцов»" +

"\nЗахаренков В.В. группа №60325-2\nВерсия: " +

Assembly.GetExecutingAssembly().GetName().Version.ToString(), "О программе…",

MessageBoxButton.OK, MessageBoxImage.Information);

}

private void ListCommand\_Executed(object sender, ExecutedRoutedEventArgs e)

{

SamplesViewDlg dlg = new SamplesViewDlg();

if (dlg.ShowDialog() == true);

}

#endregion Commands

private void CalibrationMenuItem\_Click(object sender, ExecutedRoutedEventArgs e)

{

CalibrationSelectionDlg dlg = null;

switch ((string)(e?.Parameter))

{

case "Kalium":

dlg = new CalibrationSelectionDlg("Kalium", Properties.Settings.Default.KaliumCalibrationNumber);

break;

case "Natrium":

dlg = new CalibrationSelectionDlg("Natrium", Properties.Settings.Default.NatriumCalibrationNumber);

break;

default:

break;

}

dlg.btnSetDefault.Content = "Установить по умолчанию";

dlg.btnSetDefault.ToolTip = "Установить выбранную калибровку по умолчанию для всех новых анализов";

try

{

if (dlg.ShowDialog() == true)

{

switch ((string)(e?.Parameter))

{

case "Kalium":

Properties.Settings.Default.KaliumCalibrationNumber = dlg.CalibrationNumber;

CALogger.WriteToLogFile("Properties.Settings.Default.KaliumCalibrationNumber set to: " + dlg.CalibrationNumber);

break;

case "Natrium":

Properties.Settings.Default.NatriumCalibrationNumber = dlg.CalibrationNumber;

CALogger.WriteToLogFile("Properties.Settings.Default.NatriumCalibrationNumber set to: " + dlg.CalibrationNumber);

break;

default:

break;

}

}

}

catch (Exception ex) { }

}

private void SAOptionsMenuItem\_Click (object sender, RoutedEventArgs e)

{

//use an instance of SaltAnalysis class to check for allowable values

SaltAnalysisData sa = new SaltAnalysisData();

//fill values from the user-defined section of settings

try

{

sa.HgCoefficient = Properties.Settings.Default.HgCoefficient;

sa.BromumStandardTitre = Properties.Settings.Default.BrTitre;

sa.CalciumTrilonTitre = Properties.Settings.Default.CaTrilonB;

sa.MagnesiumTrilonTitre = Properties.Settings.Default.MgTrilonB;

sa.SulfatesBlank = Properties.Settings.Default.SulfatesBlank;

sa.BromumBlank = Properties.Settings.Default.BrBlank;

}

catch

{

CALogger.WriteToLogFile("Ошибка при считывании настроек из файла конфигурации!");

return;

}

Dictionary<SaltCalculationSchemes, SchemeResultsTolerance> schemeDict = new Dictionary<SaltCalculationSchemes, SchemeResultsTolerance>(SchemeCompareOptionsHelper.GetSchemeCompareOptions());

SaltAnalysisOptionsDlg saDlg = new SaltAnalysisOptionsDlg(sa as SaltAnalysisData);

saDlg.SumTolerance = Properties.Settings.Default.SumTolerance;

saDlg.SchemesDictionary = schemeDict;

for (int j = 0; j < saDlg.SchemesDictionary.Count; j++)

{

ColumnDefinition cd = new ColumnDefinition() { };

saDlg.spSchemeTolerances.ColumnDefinitions.Add(cd);

}

//For each scheme put options into Dialog window

int i = 0;

foreach (KeyValuePair<SaltCalculationSchemes, SchemeResultsTolerance> item in saDlg.SchemesDictionary)

{

StackPanel sp = new StackPanel()

{

Name = "sp\_" + item.Key,

Orientation = Orientation.Vertical,

Width = 170,

Height = 430,

VerticalAlignment = VerticalAlignment.Top

};

sp.Children.Add(new TextBlock { Text = item.Key.ToName() });

StackPanel sp1 = new StackPanel()

{

Orientation = Orientation.Horizontal

};

CheckBox cb = new CheckBox()

{

Name = "cb\_" + item.Key,

ToolTip ="Использовать одно значение "+ Environment.NewLine +

"для проверки сходимости в данной схеме?"

};

Binding bindcb = new Binding();

bindcb.Path = new PropertyPath("SchemesDictionary[" + item.Key + "].IsUniversalTolerance");

bindcb.Mode = BindingMode.TwoWay;

bindcb.NotifyOnSourceUpdated = true;

bindcb.UpdateSourceTrigger = UpdateSourceTrigger.PropertyChanged;

BindingOperations.SetBinding(cb, CheckBox.IsCheckedProperty, bindcb);

sp1.Children.Add(cb);

TextBox tb = new TextBox()

{

Name = "tb\_" + item.Key,

ToolTip = "Единое значение толеранса для данной схемы"

};

Binding bindtb = new Binding();

bindtb.Path = new PropertyPath("SchemesDictionary[" + item.Key + "].UniversalTolerance");

bindtb.Mode = BindingMode.TwoWay;

bindtb.NotifyOnSourceUpdated = true;

bindtb.UpdateSourceTrigger = UpdateSourceTrigger.PropertyChanged;

bindtb.ValidatesOnExceptions = true;

BindingOperations.SetBinding(tb, TextBox.TextProperty, bindtb);

Binding tbvisibility = new Binding();

tbvisibility.Path = new PropertyPath("SchemesDictionary[" + item.Key + "].IsUniversalTolerance");

tbvisibility.NotifyOnSourceUpdated = true;

tbvisibility.UpdateSourceTrigger = UpdateSourceTrigger.PropertyChanged;

BindingOperations.SetBinding(tb, TextBox.IsEnabledProperty, tbvisibility);

sp1.Children.Add(tb);

sp.Children.Add(sp1);

DataGrid dgr = new DataGrid()

{

Name = "dgrd\_" + item.Key,

CanUserAddRows = false,

AutoGenerateColumns = false

};

Binding binddgr = new Binding();

binddgr.Path = new PropertyPath("SchemesDictionary[" + item.Key + "].SchemeTolerances");

binddgr.Mode = BindingMode.TwoWay;

binddgr.NotifyOnSourceUpdated = true;

binddgr.UpdateSourceTrigger = UpdateSourceTrigger.PropertyChanged;

binddgr.ValidatesOnExceptions = true;

BindingOperations.SetBinding(dgr, DataGrid.ItemsSourceProperty, binddgr);

Binding dgrdvisibility = new Binding();

dgrdvisibility.Path = new PropertyPath("SchemesDictionary[" + item.Key + "].IsUniversalTolerance");

dgrdvisibility.NotifyOnSourceUpdated = true;

dgrdvisibility.Converter = new BooleanToNegatedBooleanConverter();

dgrdvisibility.UpdateSourceTrigger = UpdateSourceTrigger.PropertyChanged;

BindingOperations.SetBinding(dgr, DataGrid.IsEnabledProperty, dgrdvisibility);

DataGridTextColumn textColumn = new DataGridTextColumn() { IsReadOnly = true, Header = "Свойство" };

textColumn.Binding = new Binding("Item1");

dgr.Columns.Add(textColumn);

TextBlock tbl = new TextBlock() { };

Binding bindtbl = new Binding("Item2");

DataGridTemplateColumn column = new DataGridTemplateColumn()

{

Header = "Толеранс",

IsReadOnly = false,

CellTemplate = (DataTemplate)saDlg.Resources["tmpCellViewTemplate"],

CellEditingTemplate = (DataTemplate)saDlg.Resources["tmpCellEditingTemplate"]

};

dgr.Columns.Add(column);

sp.Children.Add(dgr);

Grid.SetColumn(sp, i++);

saDlg.spSchemeTolerances.Children.Add(sp);

}

if (saDlg.ShowDialog() == true)

{//if OK save settings back to user.config

Properties.Settings.Default.HgCoefficient = sa.HgCoefficient;

CALogger.WriteToLogFile("Properties.Settings.Default.HgCoefficient set to:" + sa.HgCoefficient);

Properties.Settings.Default.BrTitre = sa.BromumStandardTitre;

CALogger.WriteToLogFile("Properties.Settings.Default.BrTitre set to:" + sa.BromumStandardTitre);

Properties.Settings.Default.CaTrilonB = sa.CalciumTrilonTitre;

CALogger.WriteToLogFile("Properties.Settings.Default.CaTrilonB set to:" + sa.CalciumTrilonTitre);

Properties.Settings.Default.MgTrilonB = sa.MagnesiumTrilonTitre;

CALogger.WriteToLogFile(" Properties.Settings.Default.MgTrilonB set to:" + sa.MagnesiumTrilonTitre);

Properties.Settings.Default.SulfatesBlank = sa.SulfatesBlank;

CALogger.WriteToLogFile("Properties.Settings.Default.SulfatesBlank:" + sa.SulfatesBlank);

Properties.Settings.Default.BrBlank = sa.BromumBlank;

CALogger.WriteToLogFile("Properties.Settings.Default.SulfatesBlank:" + sa.BromumBlank);

Properties.Settings.Default.SumTolerance = saDlg.SumTolerance;

CALogger.WriteToLogFile("Properties.Settings.Default.SumTolerance:" + saDlg.SumTolerance);

foreach(KeyValuePair<SaltCalculationSchemes, SchemeResultsTolerance> kvp in schemeDict)

{

try {Properties.Settings.Default[kvp.Key + "\_SchemeToleranceValues"] = kvp.Value.ToString();}

catch (Exception ex){}

}

}

}

protected override void OnClosed(EventArgs e)

{

base.OnClosed(e);

Application.Current.Shutdown();

}

/// <summary>

/// A private delegate processing user-level WM\_SHOWME message already

/// registered either by the current or the previous application instance

/// </summary>

private IntPtr WndProc(IntPtr hwnd, int msg, IntPtr wParam, IntPtr lParam, ref bool handled)

{

if (msg == NativeMethods.WM\_SHOWME)

{

wndThis.WindowState = WindowState.Normal;//return to the normal window size

bool b = wndThis.Topmost; // save state

wndThis.Topmost = true; // put on the top

wndThis.Topmost = b; //recover the state

handled = true; // mark message as handled

}

return IntPtr.Zero;

}

protected override void OnSourceInitialized(EventArgs e)

{

base.OnSourceInitialized(e);

//get handle of the current window

HwndSource source = PresentationSource.FromVisual(this) as HwndSource;

//and add an hook to process WM\_SHOWME messages

source?.AddHook(new HwndSourceHook(WndProc));

}

private void Window\_SourceInitialized(object sender, EventArgs e)

{

try

{

CALogger.InitLogFile(Properties.Settings.Default.LogFile);

}

catch

{

MessageBox.Show("Ошибка в файле конфигурации");

Close();

}

CALogger.WriteToLogFile("Программа запущена. Версия:"

+ Assembly.GetExecutingAssembly().GetName().Version.ToString());

CALogger.WriteToLogFile("Подключение к БД…");

if (!Authorize())

{

CALogger.WriteToLogFile("Ошибка авторизации!");

MessageBox.Show("Без авторизации работа невозможна!","Ошибка авторизации");

Close();

}

}

private void Relogin\_Click(object sender, ExecutedRoutedEventArgs e) => Authorize(true);

}

}

<Window x:Class="ChemicalAnalyses.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:commands="clr-namespace:ChemicalAnalyses.Commands"

xmlns:alumni="clr-namespace:ChemicalAnalyses.Alumni"

mc:Ignorable="d"

Title="{Binding WindowTitle}" Height="250" Width="425"

PreviewMouseMove="Window\_PreviewMouseMove"

SourceInitialized="Window\_SourceInitialized">

<Window.Resources>

<RoutedUICommand x:Key="ReloginCmd" Text="Сменить пользователя">

<RoutedUICommand.InputGestures>

<KeyGesture>ALT+R</KeyGesture>

</RoutedUICommand.InputGestures>

</RoutedUICommand>

<alumni:BooleanToUserTypeConverter x:Key="bln2UserTypeConv"/>

<RoutedUICommand x:Key="Calibration" Text="Работа с калибровками">

</RoutedUICommand>

</Window.Resources>

<Window.CommandBindings>

<CommandBinding Command="commands:CustomCommands.Exit" Executed="ExitCommand\_Executed"/>

<CommandBinding Command="Help" Executed="HelpCommand\_Executed"/>

<CommandBinding Command="commands:CustomCommands.List" Executed="ListCommand\_Executed"/>

<CommandBinding Command="commands:CustomCommands.Exit" Executed="ExitCommand\_Executed"/>

<CommandBinding Command="{StaticResource ReloginCmd}" Executed="Relogin\_Click"/>

<CommandBinding Command="{StaticResource Calibration}" Executed="CalibrationMenuItem\_Click"/>

</Window.CommandBindings>

<Grid ToolTip="">

<Grid.RowDefinitions>

<RowDefinition Height="25"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="25"/>

</Grid.RowDefinitions>

<!--\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Menu \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-->

<Menu x:Name="mnMainMenu" Grid.Row="0" ToolTip="Главное меню программы">

<MenuItem Header="Файл" ToolTip="Работа с файлами…">

<MenuItem Command="{StaticResource ReloginCmd}" ToolTip="Войти в систему под другими логином и паролем"

Icon="{Binding Source={StaticResource icnExitApplication}}" x:Name="miChangeUser"/>

<Separator/>

<MenuItem Command="commands:CustomCommands.Exit" ToolTip="Выход из программы"

Icon="{Binding Source={StaticResource icExit2}}"/>

</MenuItem>

<MenuItem Header="Образец" ToolTip="Выбор образцов для просмотра и изменения">

<MenuItem Header="Список…" Command ="commands:CustomCommands.List"

ToolTip="Диалоговое окно выбора образца из списка имеющихся"

Icon="{Binding Source={StaticResource icnListOfSamples}}"/>

</MenuItem>

<MenuItem Header="Настройки" ToolTip="Различные настройки программы…" x:Name="miMainOptions">

<MenuItem Header="Калибровки" ToolTip="Настройки калибровок…" Icon="{Binding Source={StaticResource icnCalib}}"

x:Name="miMainCalibrations">

<MenuItem Header="Калий" ToolTip="Настройки калибровочных данных для калия" Command="{StaticResource Calibration}"

Icon="{Binding Source={StaticResource icnKalium}}" CommandParameter="Kalium"

x:Name="miMainCalibrationsKalium"/>

<MenuItem Header="Натрий" Command="{StaticResource Calibration}" ToolTip="Настройки калибровочных данных для натрия"

Icon="{Binding Source={StaticResource icnNatrium}}" CommandParameter="Natrium"

x:Name="miMainCalibrationsNatrium"/>

</MenuItem>

<MenuItem Header="Исходные данные…" ToolTip="Разлиные исходные данные, используемые при расчете анализов"

Icon="{Binding Source={StaticResource icnInitData}}" x:Name="miInitialParameters">

<MenuItem Header="для анализа солей" Click="SAOptionsMenuItem\_Click"

ToolTip="Титры, значения холостых анализов и проч." x:Name="miSaltAnalysisOptions"

Icon="{Binding Source={StaticResource icnInitDataSalts}}"/>

</MenuItem>

</MenuItem>

<MenuItem Header="Помощь" ToolTip="Помощь…">

<Separator/>

<MenuItem Command="Help" ToolTip="О программе…" Header="О программе…"

Icon="{Binding Source={StaticResource icHelpApplication}}"/>

</MenuItem>

</Menu>

<!--\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* StatusBar \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-->

<StatusBar Grid.Row="3" VerticalAlignment="Stretch" x:Name="sbMain" ToolTip="Строка состояния">

<StatusBar.ItemsPanel>

<ItemsPanelTemplate>

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="5\*"></ColumnDefinition>

<ColumnDefinition Width="2\*"></ColumnDefinition>

</Grid.ColumnDefinitions>

</Grid>

</ItemsPanelTemplate>

</StatusBar.ItemsPanel>

<StatusBarItem Grid.Column="0">

<TextBlock x:Name="stbFirstSection" Text="{Binding Path=HoverToolTip,

RelativeSource={RelativeSource AncestorType=Window}}"/>

</StatusBarItem>

<StatusBarItem Grid.Column="1">

<StackPanel Orientation="Horizontal">

<TextBlock x:Name="stbSecondSection" Text="{Binding Path=IsAdmin,

RelativeSource={RelativeSource AncestorType=Window}, Converter={StaticResource bln2UserTypeConv}}"

ToolTip="Отображает статус текущего пользователя в системе"/>

</StackPanel>

</StatusBarItem>

</StatusBar>

</Grid>

</Window>

A.3 Класс диалогового окна для ввода логина и пароля

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace ChemicalAnalyses.Dialogs

{

public partial class UserNamePwdDlg : Window

{

private int errorCount = 0;

public string UserName

{

get { return (string)GetValue(UserNameProperty); }

set { if (value != null) SetValue(UserNameProperty, value); }

}

public static readonly DependencyProperty UserNameProperty =

DependencyProperty.Register(nameof(UserName),

typeof(string), typeof(UserNamePwdDlg),

new PropertyMetadata("Введите имя пользователя"),

new ValidateValueCallback(validateUserNameValue));

public bool wrongPwd

{

get { return (bool)GetValue(wrongPwdProperty); }

set { SetValue(wrongPwdProperty, value); }

}

public static readonly DependencyProperty wrongPwdProperty =

DependencyProperty.Register(nameof(wrongPwd),

typeof(bool), typeof(UserNamePwdDlg),

new PropertyMetadata(true));

public UserNamePwdDlg()

{

InitializeComponent();

grdMain.DataContext = this;

}

private void pbPassword\_PasswordChanged(object sender, RoutedEventArgs e)

{wrongPwd = pbPassword.Password.Length < 3;}

static bool validateUserNameValue(object value)

{

if (value == null || ((string)value).Equals(string.Empty)) return false;

return (((string)value).Length >= 3 && ((string)value).Length <= 100);

}

private void SaveCommand\_Executed(object sender, ExecutedRoutedEventArgs e)

{ DialogResult = true; }

private void SaveCommand\_CanExecute(object sender, CanExecuteRoutedEventArgs e)

{ e.CanExecute = errorCount == 0 && !wrongPwd; }

private void Window\_ValidationError(object sender, ValidationErrorEventArgs e)

{

var validationEventArgs = e as ValidationErrorEventArgs;

if (validationEventArgs == null) throw new Exception("Unexpected event args");

switch (validationEventArgs.Action)

{

case ValidationErrorEventAction.Added:

{ errorCount++; break; }

case ValidationErrorEventAction.Removed:

{ errorCount--; break; }

default:

throw new Exception("Unknown action");

}

}

}

}

<Window x:Class="ChemicalAnalyses.Dialogs.UserNamePwdDlg"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:alumni="clr-namespace:ChemicalAnalyses.Alumni"

mc:Ignorable="d"

Title="Авторизация" Height="180" Width="250" MaxHeight="180" MinHeight="180" MinWidth="250"

Validation.Error="Window\_ValidationError">

<Window.CommandBindings>

<CommandBinding Command="Save" Executed="SaveCommand\_Executed" CanExecute="SaveCommand\_CanExecute"/>

</Window.CommandBindings>

<Window.Resources>

<alumni:BooleanToVisibilityConverter x:Key="bln2VisibilityConverter"/>

</Window.Resources>

<Grid x:Name="grdMain">

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="35"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Grid.Resources>

<Style TargetType="{x:Type Button}">

<Setter Property="Width" Value="90"/>

<Setter Property="Height" Value="25"/>

</Style>

</Grid.Resources>

<StackPanel Orientation="Vertical" Grid.Column="0" Grid.ColumnSpan="2" Grid.Row="0">

<Label Content="Введите имя пользователя:" Target="{Binding ElementName=tbUserName}"/>

<TextBox Text="{Binding Path=UserName, NotifyOnValidationError=True, Mode=TwoWay,

ValidatesOnExceptions=True, UpdateSourceTrigger=PropertyChanged}" x:Name="tbUserName"

Margin="3,0,3,0"/>

<Label Content="Введите пароль:" Target="{Binding ElementName=pbPassword}"/>

<PasswordBox x:Name="pbPassword" PasswordChanged="pbPassword\_PasswordChanged" MaxLength="50"

Margin="3,0,3,0"/>

<Label Visibility="{Binding Path=wrongPwd, Converter={StaticResource bln2VisibilityConverter}}">

<TextBlock Text="Пароль от 3 символов!" Foreground="Red"/>

</Label>

</StackPanel>

<Button Content="OK" HorizontalAlignment="Center" Grid.Row="1" Grid.Column="0" Command="Save" VerticalAlignment="Center" IsCancel="False" IsDefault="True" x:Name="btnOK"/>

<Button Content="Cancel" HorizontalAlignment="Center" Grid.Row="1" Grid.Column="1" VerticalAlignment="Center" IsCancel="True" IsDefault="False" x:Name="btnCancels"/>

</Grid>

</Window>

A.4 Класс окна работы с образцами

using ChemicalAnalyses.Alumni;

using SA\_EF;

using SettingsHelper;

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Data.Entity;

using System.Data.Entity.Infrastructure;

using System.Diagnostics;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

using System.Windows.Media;

using TAlex.WPF.Controls;

namespace ChemicalAnalyses.Dialogs

{

public partial class SamplesViewDlg : Window

{

ObservableCollection<Sample> SamplesCollection;

public string FilterText

{

get { return (string)GetValue(FilterTextProperty); }

set { SetValue(FilterTextProperty, value); }

}

public static readonly DependencyProperty FilterTextProperty =

DependencyProperty.Register(nameof(FilterText), typeof(string), typeof(SamplesViewDlg),

new PropertyMetadata(null));

private static SampleFilterFields fFields;

public SampleFilterFields GetFilter() { return fFields; }

public decimal NumberOfAnalysesToAdd

{

get { return (decimal)GetValue(NumberOfAnalysesToAddProperty); }

set { SetValue(NumberOfAnalysesToAddProperty, value); }

}

public static readonly DependencyProperty NumberOfAnalysesToAddProperty =

DependencyProperty.Register(nameof(NumberOfAnalysesToAdd),

typeof(decimal), typeof(SamplesViewDlg), new PropertyMetadata(1M));

public SamplesViewDlg()

{

SamplesCollection = new ObservableCollection<Sample>();

InitializeComponent();

lbSamples.ItemsSource = SamplesCollection;

grdMain.DataContext = this;

try {fFields = Properties.Settings.Default.PreviousFilter;}

catch { }

if (fFields == null) fFields = new SampleFilterFields();

FilterText = GetFilter().ToString();

}

void FillData()

{

SamplesCollection.Clear();

try

{

using (var context = new ChemicalAnalysesEntities())

{

if (context != null)

{

#if DEBUG

context.Database.Log = (s) => { Debug.WriteLine(s); };

#endif

string[] lnArray = null;

if (fFields.LabNumber != null && !fFields.LabNumber.Equals(string.Empty))

{

if (fFields.LabNumber.Contains(';'))

{//a list of samples semicolon-separated

lnArray = Regex.Split(fFields.LabNumber, ";");

// JOIN Samples with a list of filtering labnumbers on Sample.Labnumber and then apply WHERE clause

context.Samples

.Join(inner: lnArray.ToList(),

outerKeySelector: e => e.LabNumber,

innerKeySelector: o => o.Trim(),

resultSelector: (e, o) => e)

.Where(p => p.SamplingDate <= fFields.EndDate && p.SamplingDate >= fFields.StartDate)

.ToList().ForEach(d => SamplesCollection.Add(d));

}

else context.Samples.Where(p => p.LabNumber.Equals(fFields.LabNumber))

.Where(p => p.SamplingDate <= fFields.EndDate && p.SamplingDate >= fFields.StartDate)

.ToList().ForEach(d => SamplesCollection.Add(d));

}

else context.Samples.Where(p => p.SamplingDate <= fFields.EndDate && p.SamplingDate >= fFields.StartDate)

.ToList().ForEach(d => SamplesCollection.Add(d)); ;

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка");

}

}

private void UpdateCommand\_Executed(object sender, ExecutedRoutedEventArgs e)

{

Sample smpl = SamplesCollection[lbSamples.SelectedIndex];

SampleDlg dlg = new SampleDlg(ref smpl);

dlg.Title = "Редактировать информацию об образце";

if (dlg.ShowDialog() == true)

{

try

{

using (var context = new ChemicalAnalysesEntities())

{

context.Entry(smpl).State = EntityState.Modified;

context.SaveChanges();

}

CALogger.WriteToLogFile(string.Format("Изменены данные образца {0}", smpl.ToString()));

FillData();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка при обновлении записи", MessageBoxButton.OK);

}

}

}

private void UpdateCommand\_CanExecute(object sender, CanExecuteRoutedEventArgs e)

{e.CanExecute = lbSamples?.SelectedItems.Count != 0; }

private void FilterCommand\_CanExecute(object sender, CanExecuteRoutedEventArgs e)

{ e.CanExecute = true; } //always

private void FilterCommand\_Executed(object sender, ExecutedRoutedEventArgs e)

{

SamplesFilterDlg dlg = new SamplesFilterDlg();

if (fFields != null) dlg.Filter = fFields;

if (dlg.ShowDialog() == true)

{

fFields = dlg.Filter;

FilterText = GetFilter().ToString();

FillData();

}

}

private void DeleteCommand\_CanExecute(object sender, CanExecuteRoutedEventArgs e)

{

e.CanExecute = (lbSamples?.SelectedItems.Count != 0) &&

lbSamples?.SelectedItems.Cast<Sample>().ToList().Count(p => p.SamplesCount != 0) == 0;

}

private void DeleteCommand\_Executed(object sender, ExecutedRoutedEventArgs e)

{

if (MessageBox.Show("Удаленные данные будет невозможно восстановить\n Продолжить?", "Удаление", MessageBoxButton.YesNo, MessageBoxImage.Question, MessageBoxResult.No) == MessageBoxResult.No)

return;

string sample\_to\_log = SamplesCollection[lbSamples.SelectedIndex].ToString();

using (var context = new ChemicalAnalysesEntities())

{

context.Entry(SamplesCollection[lbSamples.SelectedIndex] as Sample)

.State = EntityState.Deleted;

try

{

context.SaveChanges();

}

catch (DbUpdateException)

{

MessageBox.Show("Удалить образец не удалось. По-видимому, в базе данных для данного образца " + "присутствуют анализы, введенные другими пользователями.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Hand);

return;

}

catch (Exception ex)

{ return; }

}

CALogger.WriteToLogFile(string.Format("Удален образец {0}", sample\_to\_log));

FillData();

}

private void AddCommand\_CanExecute(object sender, CanExecuteRoutedEventArgs e)

{ e.CanExecute = true; } //always

private void AddCommand\_Executed(object sender, ExecutedRoutedEventArgs e)

{

Sample smpl = new Sample();

SampleDlg dlg = new SampleDlg(ref smpl);

dlg.Title = "Внести новый образец";

if (dlg.ShowDialog() == true)

{

try

{

CALogger.WriteToLogFile(string.Format("Внесен новый образец {0}", smpl.ToString()));

using (var context = new ChemicalAnalysesEntities())

{

#if DEBUG

context.Database.Log = s => { Debug.WriteLine(s); };

#endif

context.Samples.Add(smpl);

context.SaveChanges();

}

FillData();

}

catch (Exception ex)

{

if (ex.InnerException?.InnerException?.Data["HelpLink.EvtID"] != null

&& (string)ex.InnerException?.InnerException?.Data["HelpLink.EvtID"] == "2601")

MessageBox.Show("Лабораторный номер должен быть уникален в пределах календарного года!", "Ошибка!", MessageBoxButton.OK);

else MessageBox.Show(ex.Message, "Ошибка при добавлении записи", MessageBoxButton.OK);

}

}

}

private void ReadButton\_Click(object sender, RoutedEventArgs e)

{ FillData(); }

private void EditCommand\_CanExecute (object sender, CanExecuteRoutedEventArgs e)

{

e.CanExecute = (lbSamples?.SelectedItems.Count != 0) &&

lbSamples?.SelectedItems.Cast<Sample>().ToList().Count(p => p.SamplesCount == 0) == 0;

}

private void EditCommand\_Executed (object sender, ExecutedRoutedEventArgs e)

{

List<Sample> lst = lbSamples.SelectedItems.Cast<Sample>().ToList<Sample>();

StringBuilder title = new StringBuilder("образцов №№ ");

lst.ForEach(p => { title.Append(p.LabNumber); title.Append(" "); });

SaltAnalysisDlg saltADlg = new SaltAnalysisDlg(lst, "Edit");

saltADlg.Title = "Редактирование данных анализов для " + ((lbSamples.SelectedItems.Count == 1) ?

"образца №" + ((Sample)lbSamples.SelectedItem).IDSample.ToString() :

title.ToString());

if (saltADlg.ShowDialog() == true);

FillData();

}

private void ClearFilterMenuItem\_Click(object sender, RoutedEventArgs e)

{

fFields = new SampleFilterFields();

FilterText = fFields.ToString();

FillData();

}

private void CancelButton\_Click(object sender, RoutedEventArgs e)

{

try { Properties.Settings.Default.PreviousFilter = fFields; }

catch { }

}

private void Window\_PreviewMouseMove(object sender, MouseEventArgs e)

{

var element = Mouse.DirectlyOver as FrameworkElement;

HoverToolTip2 = GetTooltip(element);

}

protected static Object GetTooltip(FrameworkElement obj)

{

if (obj == null) { return null; }

else if (obj.ToolTip != null) { return obj.ToolTip; }

else { return GetTooltip(VisualTreeHelper.GetParent(obj) as FrameworkElement); }

}

public object HoverToolTip2

{

get { return (object)GetValue(HoverToolTip2Property); }

set { SetValue(HoverToolTip2Property, value); }

}

public static readonly DependencyProperty HoverToolTip2Property =

DependencyProperty.Register(nameof(HoverToolTip2), typeof(object), typeof(SamplesViewDlg),

new PropertyMetadata(null));

private void NumericUpDown\_ValueChanged(object sender, RoutedPropertyChangedEventArgs<decimal> e)

{

NumberOfAnalysesToAdd = ((NumericUpDown)sender).Value;

}

private void AddNewAnalysisCommand\_CanExecute(object sender, CanExecuteRoutedEventArgs e)

{

e.CanExecute = (lbSamples.SelectedItems.Count != 0);

}

private void AddNewAnalysisCommand\_Executed(object sender, ExecutedRoutedEventArgs e)

{

List<Sample> lst = lbSamples.SelectedItems.Cast<Sample>().ToList<Sample>();

StringBuilder title = new StringBuilder("образцов №№ ");

lst.ForEach(p => { title.Append(p.LabNumber); title.Append(" "); });

SaltAnalysisDlg saltADlg = new SaltAnalysisDlg(lst, "Create", (int)NumberOfAnalysesToAdd);

saltADlg.Title = "Новые данные анализов для " + ((lbSamples.SelectedItems.Count == 1) ?

"образца №" + ((Sample)lbSamples.SelectedItem).IDSample.ToString() :

title.ToString());

if (saltADlg.ShowDialog() == true) FillData();

}

private void lbSamples\_MouseDoubleClick(object sender, MouseButtonEventArgs e)

{

if (lbSamples.SelectedIndex != -1) UpdateCommand\_Executed(sender, null);

}

}

}

<Window x:Class="ChemicalAnalyses.Dialogs.SamplesViewDlg"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:commands="clr-namespace:ChemicalAnalyses.Commands"

xmlns:alumni="clr-namespace:ChemicalAnalyses.Alumni"

xmlns:talex="http://schemas.talex-soft.com/2010/xaml/presentation"

mc:Ignorable="d"

Title="Список образцов в базе" Height="350" Width="600" MinHeight="205" MinWidth="500"

PreviewMouseMove="Window\_PreviewMouseMove">

<Window.Resources>

<Style TargetType="{x:Type Button}">

<Setter Property="Button.Margin" Value="5"/>

</Style>

<alumni:StringToBooleanConverter x:Key="str2bool"/>

</Window.Resources>

<Window.CommandBindings>

<CommandBinding Command="Delete" CanExecute="DeleteCommand\_CanExecute" Executed="DeleteCommand\_Executed"/>

<CommandBinding Command="New" CanExecute="AddCommand\_CanExecute" Executed="AddCommand\_Executed"/>

<CommandBinding Command="commands:CustomCommands.Update" Executed="UpdateCommand\_Executed"

CanExecute="UpdateCommand\_CanExecute" />

<CommandBinding Command="commands:CustomCommands.Filter" CanExecute="FilterCommand\_CanExecute"

Executed="FilterCommand\_Executed"/>

<CommandBinding Command="commands:CustomCommands.Edit" CanExecute="EditCommand\_CanExecute"

Executed="EditCommand\_Executed"/>

<CommandBinding Command="commands:CustomCommands.AddNewAnalysis" CanExecute="AddNewAnalysisCommand\_CanExecute"

Executed="AddNewAnalysisCommand\_Executed"/>

</Window.CommandBindings>

<Grid x:Name="grdMain">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="120"/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="auto"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="auto"/>

</Grid.RowDefinitions>

<StackPanel Grid.Column="0" Grid.Row="0" Grid.RowSpan="3" Margin="0,10,0,0">

<Button Content="Загрузить список" Click="ReadButton\_Click" ToolTip="Загрузить список образцов из базы в соответствии с&#x0a;установленным фильтром"/>

<Button Content="Добавить…" Command="New" ToolTip="Добавить новый образец"/>

<Button Content="Редактировать…" Command="commands:CustomCommands.Update" x:Name="btnEditSample"

ToolTip="Редактировать данные образца" ToolTipService.ShowOnDisabled="True"/>

<Button Command="Delete" Content="Удалить" ToolTip="Удалить образец из базы" ToolTipService.ShowOnDisabled="True"/>

<Button Content="Фильтр…" Command="commands:CustomCommands.Filter" x:Name="btnFilter"

ToolTip="Фильтровать записи" ToolTipService.ShowOnDisabled="True"/>

<Button Content="Выход" IsCancel="True" ToolTip="Закрыть окно" Click="CancelButton\_Click"/>

</StackPanel>

<TextBlock Grid.Column="1" Grid.Row="0" x:Name="txtFilters"

Text="{Binding Path=FilterText, TargetNullValue=Фильтр сброшен}"

HorizontalAlignment="Center" TextWrapping="Wrap" ToolTip="Образцы в списке отобраны в соответствии с указанными параметрами&#x0a;Для сброса перейдите в контекстное меню">

<TextBlock.ContextMenu>

<ContextMenu>

<MenuItem Header="Сбросить фильтр" Click="ClearFilterMenuItem\_Click" x:Name="miClearFilter"

IsEnabled="{Binding RelativeSource={RelativeSource AncestorType=ContextMenu, Mode=FindAncestor},

Path=PlacementTarget.Text, Converter={StaticResource str2bool}, NotifyOnSourceUpdated=True}"

ToolTip="Устанавливает фильтр по умолчанию: интервал дат 1 — год с текущего момента&#x0a;без фильтрации по номеру образца"/>

</ContextMenu>

</TextBlock.ContextMenu>

</TextBlock>

<ListBox Grid.Column="1" Grid.Row="1" x:Name="lbSamples" ItemsSource="{Binding}" VerticalContentAlignment="Stretch"

SelectionMode="Extended" MouseDoubleClick="lbSamples\_MouseDoubleClick">

<ListBox.ContextMenu>

<ContextMenu>

<MenuItem Command="commands:CustomCommands.AddNewAnalysis" x:Name="miAddAnalyses"

ToolTip="Добавляет к каждому образцу выбранное количество записей для данных анализа">

<MenuItem.Header>

<StackPanel Orientation="Horizontal">

<TextBlock Text="Ввести данные новых анализов "/>

<talex:NumericUpDown Minimum="1" Maximum="30" ValueChanged="NumericUpDown\_ValueChanged"

ToolTip="Установите количество добавляемых анализов к каждому образцу"

x:Name="taNumUpDown"/>

<TextBlock Text=" штук"/>

</StackPanel>

</MenuItem.Header>

</MenuItem>

<MenuItem Header="Редактировать данные анализов" Command="commands:CustomCommands.Edit" x:Name="miEditAnalyses"/>

<MenuItem Header="Удалить образец" Command="Delete" ToolTip="Удалить образец из базы" x:Name="miDeleteSample"/>

<MenuItem Header="Редактировать…" Command="commands:CustomCommands.Update"

ToolTip="Редактировать данные образца" x:Name="miEditSample"/>

</ContextMenu>

</ListBox.ContextMenu>

</ListBox>

<TextBlock Grid.Column="0" Grid.Row="2" Text="{Binding ElementName=lbSamples, Path=Items.Count,

StringFormat={}{0} записей, UpdateSourceTrigger=PropertyChanged}" HorizontalAlignment="Center"

Margin="7" ToolTip="Отображает количество загруженных на данный момент образцов"/>

<StatusBar Grid.Column="1" Grid.Row="2">

<TextBlock Text="{Binding Path=HoverToolTip2,

RelativeSource={RelativeSource AncestorType=Window}}"/>

</StatusBar>

</Grid>

</Window>

A.5 Класс установки и редактирования фильтра для отбора образцов

using ChemicalAnalyses.Alumni;

using System;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

namespace ChemicalAnalyses.Dialogs

{

public partial class SamplesFilterDlg : Window

{

private int errorCount = 0;

public SampleFilterFields Filter

{

get { return (SampleFilterFields)GetValue(FilterProperty); }

set { SetValue(FilterProperty, value); }

}

public static readonly DependencyProperty FilterProperty =

DependencyProperty.Register(nameof(Filter), typeof(SampleFilterFields), typeof(SamplesFilterDlg),

new PropertyMetadata(null));

public SamplesFilterDlg()

{

InitializeComponent();

this.DataContext = this;

}

private void CommandBinding\_CanExecute(object sender, CanExecuteRoutedEventArgs e)

{ e.CanExecute = errorCount == 0; }

private void CommandBinding\_Executed(object sender, ExecutedRoutedEventArgs e)

{DialogResult = true;}

private void Window\_Error(object sender, ValidationErrorEventArgs e)

{

var validationEventArgs = e as ValidationErrorEventArgs;

if (validationEventArgs == null) throw new Exception("Unexpected event args");

switch (validationEventArgs.Action)

{

case ValidationErrorEventAction.Added:

{ errorCount++; break; }

case ValidationErrorEventAction.Removed:

{ errorCount--; break; }

default:

{ throw new Exception("Unknown action"); }

}

}

}

}

<Window x:Class="ChemicalAnalyses.Dialogs.SamplesFilterDlg"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

mc:Ignorable="d"

Title="Отбор образцов по параметрам"

Height="200" Width="300" MaxHeight="200" MaxWidth="300" MinHeight="200" MinWidth="300" Validation.Error="Window\_Error">

<Window.CommandBindings>

<CommandBinding Command="Save" CanExecute="CommandBinding\_CanExecute" Executed="CommandBinding\_Executed"/>

</Window.CommandBindings>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="30"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<TextBlock Grid.Row="0" Grid.ColumnSpan="2"

Text="Выберите интервал дат для фильтрации образцов:"/>

<StackPanel Orientation="Vertical" Grid.Column="0" Grid.Row="1">

<Label Content="\_c:" Margin="5,-2,5,-2" Target="{Binding ElementName=dpStartDate}"/>

<DatePicker x:Name="dpStartDate" Text="{Binding Path=Filter.StartDate, ValidatesOnExceptions=True, UpdateSourceTrigger=PropertyChanged, StringFormat={}{0:dd/MM/yyyy}, Mode=TwoWay,

NotifyOnValidationError=True}"

SelectedDateFormat="Short" Language="ru-RU">

<DatePicker.Resources>

<Style TargetType="{x:Type DatePickerTextBox}">

<Setter Property="Control.Template">

<Setter.Value>

<ControlTemplate>

<TextBox x:Name="PART\_TextBox" Text="{Binding Path=SelectedDate, RelativeSource={RelativeSource AncestorType={x:Type DatePicker}}, StringFormat={}{0:dd-MM-yyyy}}"/>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

</DatePicker.Resources>

</DatePicker>

</StackPanel>

<StackPanel Orientation="Vertical" Grid.Column="1" Grid.Row="1">

<Label Content="\_по:" Margin="5,-2,5,-2" Target="{Binding ElementName=dpEndDate}"/>

<DatePicker x:Name="dpEndDate" Text="{Binding Path=Filter.EndDate, ValidatesOnExceptions=True,

UpdateSourceTrigger=PropertyChanged, StringFormat={}{0:dd/MM/yyyy}, Mode=TwoWay,

NotifyOnValidationError=True}"

SelectedDateFormat="Short" Language="ru-RU">

<DatePicker.Resources>

<Style TargetType="{x:Type DatePickerTextBox}">

<Setter Property="Control.Template">

<Setter.Value>

<ControlTemplate>

<TextBox x:Name="PART\_TextBox" Text="{Binding Path=SelectedDate, RelativeSource={RelativeSource AncestorType={x:Type DatePicker}}, StringFormat={}{0:dd-MM-yyyy}}"/>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

</DatePicker.Resources>

</DatePicker>

</StackPanel>

<StackPanel Orientation="Vertical" Grid.Row="2" Grid.ColumnSpan="2">

<Label Content="\_Лабораторный номер (точное совпадение):" Target="{Binding ElementName=tbLR}"/>

<TextBox x:Name="tbLR" Text="{Binding Path=Filter.LabNumber, NotifyOnTargetUpdated=True, UpdateSourceTrigger=PropertyChanged}" ToolTip="Введите лабораторный номер для поиска&#x0a;Несколько номеров можно разделить символом'|'&#x0a;Начальные и конечные пробелы игнорируются"/>

</StackPanel>

<Button Content="OK" IsCancel="False" IsDefault="False" Grid.Row="3" Grid.Column="0" Margin="5" Command="Save" Width="90"/>

<Button Content="Cancel" IsCancel="True" IsDefault="True" Grid.Row="3" Grid.Column="1" Margin="5" Width="90"/>

</Grid>

</Window>

A.6 Класс просмотра и редактирования свойств образца

using System.Windows;

using SA\_EF;

namespace ChemicalAnalyses.Dialogs

{

public partial class SampleDlg : Window

{

public Sample smpl { get; set; }

public SampleDlg(ref Sample smpl)

{

this.smpl = smpl;

InitializeComponent();

grdSample.DataContext = this.smpl;

}

private void OKButton\_Click(object sender, RoutedEventArgs e)

{

smpl.LabNumber = smpl.LabNumber.Trim();

this.DialogResult = true;

}

}

}

<Window x:Class="ChemicalAnalyses.Dialogs.SampleDlg"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:alumni="clr-namespace:ChemicalAnalyses.Alumni"

mc:Ignorable="d"

Title="{Binding}" Height="200" Width="300" MaxWidth="350" MaxHeight="270">

<Window.Resources>

<alumni:SampleAvConverter x:Key="avConv"/>

</Window.Resources>

<Grid x:Name="grdSample">

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="30"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<StackPanel Grid.Row="0" Grid.Column="0">

<Label Content="Лабораторный номер:" Grid.RowSpan="2"/>

<TextBox x:Name="tbLabNumber" Text="{Binding Path=LabNumber, ValidatesOnExceptions=True, UpdateSourceTrigger=PropertyChanged}" Grid.RowSpan="2" Height="{Binding ElementName=dpSamplingDate, Path=ActualHeight}" ToolTip="Лабораторный номер. От 2 до 15 символов включительно.&#x0a;Использование символа ';' не допускается."/>

</StackPanel>

<StackPanel Grid.Row="0" Grid.Column="1">

<Label Content="Дата отбора:" Grid.RowSpan="2"/>

<DatePicker x:Name="dpSamplingDate" Text="{Binding Path=SamplingDate, ValidatesOnExceptions=True, UpdateSourceTrigger=PropertyChanged, StringFormat={}{0:dd/MM/yyyy},Mode=TwoWay}" Grid.RowSpan="2" SelectedDateFormat="Short" Language="ru-RU">

<DatePicker.Resources>

<Style TargetType="{x:Type DatePickerTextBox}">

<Setter Property="Control.Template">

<Setter.Value>

<ControlTemplate>

<TextBox x:Name="PART\_TextBox" Text="{Binding Path=SelectedDate, RelativeSource={RelativeSource AncestorType={x:Type DatePicker}}, StringFormat={}{0:dd-MM-yyyy}}"/>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

</DatePicker.Resources>

</DatePicker>

</StackPanel>

<StackPanel Grid.Row="1" Grid.ColumnSpan="2" Grid.Column="0">

<Label Content="Описание:" Grid.RowSpan="2"/>

<TextBox x:Name="tbDescription" Text="{Binding Path=Description, ValidatesOnExceptions=True, UpdateSourceTrigger=PropertyChanged}" TextWrapping="WrapWithOverflow" Height="{Binding ElementName=dpSamplingDate, Path=ActualHeight}" ToolTip="Краткое описание образца"/>

</StackPanel>

<Button Grid.Row="2" Grid.Column="2" Width="90" Height="20" Content="Cancel" HorizontalAlignment="Center" IsCancel="True"/>

<Button Grid.Row="2" Grid.Column="0" Width="90" Height="20" Content="OK" HorizontalAlignment="Center" IsDefault="True" IsCancel="False" Click="OKButton\_Click">

<Button.IsEnabled>

<MultiBinding Converter="{StaticResource avConv}" UpdateSourceTrigger="PropertyChanged">

<Binding ElementName="tbLabNumber" Path="Text"/>

<Binding ElementName="dpSamplingDate" Path="Text"/>

<Binding ElementName="tbDescription" Path="Text"/>

</MultiBinding>

</Button.IsEnabled>

</Button>

</Grid>

</Window>

A.7 Класс просмотра и редактирования начальных параметров для анализов

using SA\_EF;

using System;

using System.Collections.Generic;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

namespace ChemicalAnalyses.Dialogs

{

public partial class SaltAnalysisOptionsDlg : Window

{

public SaltAnalysisData sa\_local { get; set; }

private int errorCount = 0;

public decimal SumTolerance

{

get { return (decimal)GetValue(SumToleranceProperty); }

set { SetValue(SumToleranceProperty, value); }

}

public static readonly DependencyProperty SumToleranceProperty = DependencyProperty.Register(

nameof(SumTolerance), typeof(decimal), typeof(SaltAnalysisOptionsDlg),

new PropertyMetadata(0.02M), new ValidateValueCallback(validateSumToleranceValue));

public Dictionary<SaltCalculationSchemes,SchemeResultsTolerance> SchemesDictionary

{

get { return (Dictionary<SaltCalculationSchemes,SchemeResultsTolerance>)GetValue(SchemesDictionaryProperty); }

set { SetValue(SchemesDictionaryProperty, value); }

}

public static readonly DependencyProperty SchemesDictionaryProperty =

DependencyProperty.Register(nameof(SchemesDictionary),

typeof(Dictionary<SaltCalculationSchemes,SchemeResultsTolerance>), typeof(SaltAnalysisOptionsDlg));

public SaltAnalysisOptionsDlg(SaltAnalysisData sa)

{

sa\_local = sa;

InitializeComponent();

grdMain.DataContext = this;

}

private void Window\_ValidationError(object sender, ValidationErrorEventArgs e)

{

var validationEventArgs = e as ValidationErrorEventArgs;

if (validationEventArgs == null) throw new Exception("Unexpected event args");

switch (validationEventArgs.Action)

{

case ValidationErrorEventAction.Added:

{errorCount++; break;}

case ValidationErrorEventAction.Removed:

{errorCount--; break;}

default:

throw new Exception("Unknown action");

}

}

private void SaveCommand\_Executed(object sender, ExecutedRoutedEventArgs e)

{DialogResult = true;}

private void SaveCommand\_CanExecute(object sender, CanExecuteRoutedEventArgs e)

{e.CanExecute = errorCount == 0;}

static bool validateSumToleranceValue(object value)

{ return (decimal)value > 0.001M && (decimal)value < 0.1M; }

}

}

<Window x:Class="ChemicalAnalyses.Dialogs.SaltAnalysisOptionsDlg"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

mc:Ignorable="d"

Title="Исходные настройки для анализа солей:" Height="250" Width="550" MaxHeight="250" MaxWidth="550" Validation.Error="Window\_ValidationError">

<Window.CommandBindings>

<CommandBinding Command="Save" Executed="SaveCommand\_Executed" CanExecute="SaveCommand\_CanExecute"/>

</Window.CommandBindings>

<Window.Resources>

<DataTemplate x:Key="tmpCellViewTemplate">

<TextBlock Text="{Binding Path=Item2, StringFormat={}{0:0.000 %}}"/>

</DataTemplate>

<DataTemplate x:Key="tmpCellEditingTemplate">

<TextBox Text="{Binding Path=Item2,

UpdateSourceTrigger=PropertyChanged, StringFormat={}{0:0.000}, ValidatesOnExceptions=True,

NotifyOnValidationError=True}"/>

</DataTemplate>

</Window.Resources>

<Grid x:Name="grdMain">

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="35"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<TabControl Grid.Row="0" Grid.ColumnSpan="2" Grid.Column="0" x:Name="tcSaltOptionsTabs">

<TabItem Header="Анализ" x:Name="tabAnalysis">

<AdornerDecorator>

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<StackPanel Orientation="Vertical" Margin="5" Grid.Column="0" Grid.Row="0">

<Label Content="\_ТрилонБ (Ca) мг-экв/л:" Target="{Binding ElementName=tbCaTrilonB}"/>

<TextBox x:Name="tbCaTrilonB" Text="{Binding Path=sa\_local.CalciumTrilonTitre, ValidatesOnExceptions=True, NotifyOnValidationError=True, UpdateSourceTrigger=PropertyChanged}"/>

<Label Content="ТрилонБ (\_Mg) мг-экв/л:" Target="{Binding ElementName=tbMgTrilonB}"/>

<TextBox x:Name="tbMgTrilonB" Text="{Binding Path=sa\_local.MagnesiumTrilonTitre, ValidatesOnExceptions=True, NotifyOnValidationError=True, UpdateSourceTrigger=PropertyChanged}"/>

<Label Content="\_Холостой Br мл:" Target="{Binding ElementName=tbBrBlank}"/>

<TextBox x:Name="tbBrBlank" Text="{Binding Path=sa\_local.BromumBlank, ValidatesOnExceptions=True, NotifyOnValidationError=True, UpdateSourceTrigger=PropertyChanged}"/>

</StackPanel>

<StackPanel Orientation="Vertical" Margin="5" Grid.Column="1" Grid.Row="0">

<Label Content="Эквивалент (Hg) мл:" Target="{Binding ElementName=tbHg, ValidatesOnExceptions=True, NotifyOnValidationError=True, UpdateSourceTrigger=PropertyChanged}"/>

<TextBox x:Name="tbHg" Text="{Binding Path=sa\_local.HgCoefficient, ValidatesOnExceptions=True, NotifyOnValidationError=True, UpdateSourceTrigger=PropertyChanged}"/>

<Label Target="{Binding ElementName=tbSulfatesBlank}" Content="Холостой SO₄²¯"/>

<TextBox x:Name="tbSulfatesBlank" Text="{Binding Path=sa\_local.SulfatesBlank, ValidatesOnExceptions=True, NotifyOnValidationError=True, UpdateSourceTrigger=PropertyChanged}"/>

<Label Content="Титр Br мг-экв/мл:" Target="{Binding ElementName=tbBrTitre, ValidatesOnExceptions=True, NotifyOnValidationError=True, UpdateSourceTrigger=PropertyChanged}"/>

<TextBox x:Name="tbBrTitre" Text="{Binding Path=sa\_local.BromumStandardTitre, ValidatesOnExceptions=True, NotifyOnValidationError=True, UpdateSourceTrigger=PropertyChanged}"/>

</StackPanel>

<StackPanel Grid.Column="2" Orientation="Vertical" Margin="5" >

<StackPanel.ToolTip>

<ToolTip>

<ToolTip.Content>

<StackPanel Orientation="Vertical">

<TextBlock Text="Значение толеранса, с которым оцениваются суммы."/>

<TextBlock Text="Должно находиться в интервале от 0.1% до 10%"/>

</StackPanel>

</ToolTip.Content>

</ToolTip>

</StackPanel.ToolTip>

<Label Content="\_Толеранс сумм:" Target="{Binding ElementName=tbSumTolerance}"/>

<TextBox x:Name="tbSumTolerance" Grid.Column="0" Text="{Binding Path=SumTolerance, ValidatesOnExceptions=True, NotifyOnValidationError=True, UpdateSourceTrigger=PropertyChanged}" MinWidth="60"/>

</StackPanel>

</Grid>

</AdornerDecorator>

</TabItem>

<TabItem Header="Проверка" x:Name="tabControl">

<AdornerDecorator>

<ScrollViewer HorizontalScrollBarVisibility="Auto" VerticalScrollBarVisibility="Auto">

<Grid x:Name="spSchemeTolerances" Height="450"/>

</ScrollViewer>

</AdornerDecorator>

</TabItem>

</TabControl>

<Button Content="OK" Command="Save" IsDefault="False" IsCancel="False" Grid.Column="0" Grid.Row="1" Width="80" Height="25"/>

<Button Content="Cancel" IsDefault="True" IsCancel="True" Grid.Column="1" Grid.Row="1" Width="80" Height="25"/>

</Grid>

</Window>

A.8 Класс диалогового окна работа с анализами

using ChemicalAnalyses.Alumni;

using Microsoft.Office.Interop.Excel;

using Microsoft.Win32;

using PrintHelper;

using SA\_EF;

using SA\_EF.Interfaces;

using SettingsHelper;

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.ComponentModel;

using System.Data.Entity;

using System.Diagnostics;

using System.Linq;

using System.Reflection;

using System.Text;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

namespace ChemicalAnalyses.Dialogs

{

public partial class SaltAnalysisDlg : System.Windows.Window

{

private int \_validationErrorCount = 0;

private int \_validationRowErrorCount = 0;

public ObservableCollection<SaltAnalysisData> saltAnalysisDatas { get; set; }

public string TypeOfWork { get; set; }

public List<Sample> Labnumbers { get; set; }

private bool \_all\_selected = false;

private List<SchemesPrintingGrid> pGrids;

private bool AutoScroll = true;

public static ObservableCollection<KeyValuePair<SaltCalculationSchemes, string>> SchemesNames { get; set; }

public SaltAnalysisDlg(List<Sample> lst, string typeOfWork = "Create", int qToAdd = 1)

{

TypeOfWork = typeOfWork;

decimal HgCoeff = Properties.Settings.Default.HgCoefficient;

decimal BrBlank = Properties.Settings.Default.BrBlank;

decimal BrTitre = Properties.Settings.Default.BrTitre;

decimal SulfatesBlank = Properties.Settings.Default.SulfatesBlank;

decimal CaTrilonB = Properties.Settings.Default.CaTrilonB;

decimal MgTrilonB = Properties.Settings.Default.MgTrilonB;

int KaliumCalibrationNumber = Properties.Settings.Default.KaliumCalibrationNumber;

Labnumbers = lst;

if (SchemesNames == null) SchemesNames = new ObservableCollection<KeyValuePair< SaltCalculationSchemes, string>> (Enum.GetValues(typeof(SaltCalculationSchemes)).OfType<SaltCalculationSchemes>()

.Select(p => new KeyValuePair<SaltCalculationSchemes, string>( p, p.ToName())));

saltAnalysisDatas = new ObservableCollection<SaltAnalysisData>();

InitializeComponent();

//Add handler to process cell editing events

dgrdSA.CellEditEnding += dgrdSA\_CellEditEnding;

if (TypeOfWork == "Create")

{//Creating new analyses 'data with on-default settings

Labnumbers.ForEach(p => {

for (int i = 1; i <= qToAdd; i++) {

saltAnalysisDatas.Add(

new SaltAnalysisData

{

IDSample = p.IDSample,

LabNumber = p.LabNumber,

HgCoefficient = HgCoeff,

BromumBlank = BrBlank,

BromumStandardTitre = BrTitre,

CalciumTrilonTitre = CaTrilonB,

MagnesiumTrilonTitre = MgTrilonB,

KaliumCalibration = KaliumCalibrationNumber

}); //of Add

}});//of for and Foreach

//Change OK button tooltip

btnOK.ToolTip = "Сохранить новые введенные данные для анализов";

btnDelete.Visibility = Visibility.Collapsed;

btnPrint.Visibility = Visibility.Collapsed;

}

else

{

FillData();

//Change OK button tooltip

btnOK.ToolTip = "Сохранить измененные данные для анализов";

btnDelete.Visibility = Visibility.Visible;

btnPrint.Visibility = Visibility.Collapsed;

}

grdMain.DataContext = this;

}

private void FillData()

{

saltAnalysisDatas.Clear();

using (var context = new ChemicalAnalysesEntities())

{

#if DEBUG

context.Database.Log = s => { Debug.WriteLine(s); };

#endif

if (Labnumbers?.Count == 1)

{

try

{

var smpl = context.Samples

.Find(Labnumbers[0].IDSample).SaltAnalysisDatas;

foreach (SaltAnalysisData sad in smpl)

{

sad.LabNumber = Labnumbers[0].LabNumber;

saltAnalysisDatas.Add(sad);

}

}

catch (Exception ex) { }

}

else

{

var sampl\_list = (from Labnumber in Labnumbers

join smpl in context.SaltAnalysisDatas on Labnumber.IDSample equals smpl.IDSample

select smpl);

foreach (SaltAnalysisData sad in sampl\_list)

{

sad.LabNumber = context.Samples.Where(p=>p.IDSample == sad.IDSample)

.FirstOrDefault().LabNumber;

saltAnalysisDatas.Add(sad);

}

}

}

}

private void OnErrorEvent(object sender, RoutedEventArgs e)

{

var validationEventArgs = e as ValidationErrorEventArgs;

if (validationEventArgs == null) throw new Exception("Unexpected event args");

switch (validationEventArgs.Action)

{

case ValidationErrorEventAction.Added:

{\_validationErrorCount++; break;}

case ValidationErrorEventAction.Removed:

{\_validationErrorCount--; break;}

default:

throw new Exception("Unknown action");

}

}

private void OnRowErrorEvent(object sender, RoutedEventArgs e)

{

var validationEventArgs = e as ValidationErrorEventArgs;

if (validationEventArgs == null) throw new Exception("Unexpected event args");

switch (validationEventArgs.Action)

{

case ValidationErrorEventAction.Added:

{\_validationRowErrorCount++; break;}

case ValidationErrorEventAction.Removed:

{\_validationRowErrorCount--; break;}

default:

throw new Exception("Unknown action");

}

}

private void SaveCommand\_CanExecute (object sender, CanExecuteRoutedEventArgs e)

{ e.CanExecute = (\_validationErrorCount == 0) && (\_validationRowErrorCount == 0) && (dgrdSA.Items.Count >= 1);}

private void SaveCommand\_Executed(object sender, ExecutedRoutedEventArgs e)

{

StringBuilder s = new StringBuilder();

try

{

if (TypeOfWork == "Create")

{

using (var context = new ChemicalAnalysesEntities())

{

dgrdSA.Items.Cast<SaltAnalysisData>().ToList().ForEach(p =>

{

context.Entry(p).State = EntityState.Added;

s.AppendLine(p.LabNumber);

});

context.SaveChanges();

}

CALogger.WriteToLogFile("Созданы новые анализы для следующих образцов: " + s);

}

else if (TypeOfWork == "Edit")

{

using (var context = new ChemicalAnalysesEntities())

{

dgrdSA.Items.Cast<SaltAnalysisData>().ToList().ForEach(p =>

{

context.Entry(p).State = EntityState.Modified;

s.AppendLine(p.LabNumber);

});

context.SaveChanges();

}

CALogger.WriteToLogFile("Изменены данные анализов для следующих образцов: " + s);

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message + " в " + ex.Source, "Ошибка");

}

DialogResult = true;

}

private void DeleteCommand\_CanExecute (object sender, CanExecuteRoutedEventArgs e)

{ e.CanExecute = dgrdSA.SelectedItems.Count != 0; }

private void DeleteCommand\_Executed (object sender, ExecutedRoutedEventArgs e)

{

if (MessageBox.Show("Удаленные данные будет невозможно восстановить\n Продолжить?", "Удаление", MessageBoxButton.YesNo, MessageBoxImage.Question, MessageBoxResult.No) == MessageBoxResult.No)

return;

StringBuilder s = new StringBuilder();

try

{

using (var context = new ChemicalAnalysesEntities())

{

dgrdSA.SelectedItems.Cast<SaltAnalysisData>().ToList().ForEach(p =>

{

s.AppendLine(p.LabNumber);

context.Entry(p).State=EntityState.Deleted;

});

context.SaveChanges();

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message + " в " + ex.Source, "Ошибка");

}

finally

{

CALogger.WriteToLogFile("Удалены данные анализов для следующих образцов: " + s);

FillData();

}

}

private void CalibrationButton\_Click (object sender, RoutedEventArgs e)

{

if (dgrdSA.SelectedItems.Count > 1)

{

MessageBox.Show("Выбрано несколько образцов"+Environment.NewLine +"Выберите только один!","Ошибка", MessageBoxButton.OK,MessageBoxImage.Error);

return;

}

CalibrationSelectionDlg dlg = null;

dlg = new CalibrationSelectionDlg("Kalium", ((SaltAnalysisData)dgrdSA.SelectedItem).KaliumCalibration);

dlg.btnSetDefault.Content = "Установить для №" + ((ISaltAnalysisCalcResults)dgrdSA.SelectedItem).LabNumber;

dlg.btnSetDefault.ToolTip = "Установить выбранную калибровку только для анализа образца с лабораторным № " +

((ISaltAnalysisCalcResults)dgrdSA.SelectedItem).LabNumber;

if (dlg.ShowDialog() == true)

{

((SaltAnalysisData)dgrdSA.SelectedItem).KaliumCalibration = dlg.CalibrationNumber;

}

}

private void CalculateCommand\_CanExecute(object sender, CanExecuteRoutedEventArgs e)

{e.CanExecute = dgrdSA.SelectedItems.Count != 0 && \_validationErrorCount == 0;}

private void CalculateCommand\_Executed(object sender, ExecutedRoutedEventArgs e)

{

if (!ConfirmWhenSchemesDiffer()) return;

dgrdSA.SelectedItems.Cast<SaltAnalysisData>().ToList().ForEach(p => {

p.CalcDryValues(p.DefaultCalculationScheme);

p.KDry = p.CalcKaliumValue();

//calculate using the user-selected scheme

p.CalcSchemeResults(p, p.DefaultCalculationScheme);

//set recommended scheme to the calculated one

p.RecommendedCalculationScheme = p.CalcRecommendedScheme(p);

p.IonSumColor = p.CalcSchemeTolerance(p).IonSumColor;

p.SaltSumColor = p.CalcSchemeTolerance(p).SaltSumColor;

});

MessageBox.Show(dgrdSA.SelectedItems.Count.ToString() + " образцов были расчитаны", "Результаты расчета");

dgrdSA.SelectedItems.Clear();

btnPrint.Visibility = Visibility.Visible;

}

private void PrintCommand\_Executed(object sender, ExecutedRoutedEventArgs e)

{

if (!ConfirmWhenSchemesDiffer()) return;

SAPrintPreview sAPrintPreview = new SAPrintPreview

{

Owner = this,

Title = @"Предварительный просмотр результатов расчета"

};

ContextMenu contextMenu = new ContextMenu();

System.Windows.Controls.MenuItem menuItem

= new System.Windows.Controls.MenuItem() { Header = "Сохранить как…" };

menuItem.Click += new RoutedEventHandler(\_setWorkbook);

pGrids = new List<SchemesPrintingGrid>();

IEnumerable<ISaltAnalysisCalcResults> res = dgrdSA.SelectedItems.Cast<ISaltAnalysisCalcResults>();

foreach (SaltCalculationSchemes schem in Enum.GetValues(typeof(SaltCalculationSchemes))

.Cast<SaltCalculationSchemes>())

{

var tmp = res.Where(p => p.DefaultCalculationScheme == schem);

if (tmp.Count() > 0 )

{

if (tmp.Count() <= 30)

{

SchemesPrintingGrid p = new SchemesPrintingGrid(tmp)

{

Name = "pg" + schem.ToString(),

ResultsType = schem

};

pGrids.Add(p);

}

else

{

int j = 0;

foreach(var item in tmp.Chunk(30))

{

SchemesPrintingGrid p = new SchemesPrintingGrid(item)

{

Name = "pg" + j++ + schem.ToString(),

ResultsType = schem

};

pGrids.Add(p);

}

}

}

}

foreach (SchemesPrintingGrid pgrd in pGrids)

{

System.Windows.Controls.TextBox tbTitle

= new System.Windows.Controls.TextBox()

{

Text = string.Format( "Химический состав, % (схема: {0})", pgrd.ResultsType.ToName()),

FontSize = 24,

FontWeight = FontWeights.Bold,

Width = 1000,

HorizontalContentAlignment = HorizontalAlignment.Center,

BorderThickness = new Thickness(0),

IsReadOnly = true

};

sAPrintPreview.fdSA.Pages.Add(

new PageContent()

{

Name = "pk1\_" + pgrd.Name,

Child = new FixedPage()

{

Name = "fp1\_"+pgrd.Name,

Width = 1056,

Height = 890,

Children = { new Canvas() { Children = { pgrd, tbTitle } } }

}

});

Canvas.SetLeft(pgrd, 10);

Canvas.SetTop(pgrd, 50);

Canvas.SetTop(tbTitle, 10);

Canvas.SetLeft(tbTitle, 10);

}

StackPanel spHygro = new StackPanel() { Orientation = Orientation.Horizontal };

sAPrintPreview.grdOptions.Children.Add(spHygro);

contextMenu.Items.Add(menuItem);

sAPrintPreview.fdSA.ContextMenu = contextMenu;

Grid.SetColumn(spHygro, 0);

Grid.SetColumnSpan(spHygro, 2);

System.Windows.Controls.Label lbHygro =

new System.Windows.Controls.Label() { Content = "Выводить гигроскопическую влагу для всех образцов? " };

spHygro.Children.Add(lbHygro);

System.Windows.Controls.CheckBox cbHygro = new System.Windows.Controls.CheckBox();

Binding bdHygro = new Binding("ShowHygroscopicWaterForAll") {

Source = pGrids.Where(p=>p.ResultsType == SaltCalculationSchemes.Chloride).FirstOrDefault()};

cbHygro.SetBinding(System.Windows.Controls.CheckBox.IsCheckedProperty, bdHygro);

spHygro.Children.Add(cbHygro);

sAPrintPreview.ShowDialog();

}

private void \_setWorkbook(object sender, EventArgs e)

{

\_Application exApp = null;

Workbook wb = null;

try

{

exApp = new Microsoft.Office.Interop.Excel.Application();

wb = exApp.Workbooks.Add();

SaveFileDialog saveDialog = new SaveFileDialog();

saveDialog.Filter = "Excel files (\*.xlsx)|\*.xlsx|All files (\*.\*)|\*.\*";

saveDialog.FilterIndex = 1;

if (saveDialog.ShowDialog() == true)

{

foreach (SchemesPrintingGrid p in pGrids) p.ExportToExcel(ref wb);

wb.SaveAs(saveDialog.FileName);

MessageBox.Show(string.Format($"Эскпорт в файл {0} осуществлен", saveDialog.FileName),"Экспорт");

}

}

catch { }

finally

{

wb?.Close(XlSaveAction.xlSaveChanges);

exApp?.Quit();

}

}

private void dgrdSA\_LoadingRow(object sender, DataGridRowEventArgs e)

{

e.Row.AddHandler(System.Windows.Controls.Validation.ErrorEvent,

new RoutedEventHandler(OnRowErrorEvent));

}

private void PrintCommand\_CanExecute(object sender, CanExecuteRoutedEventArgs e)

{

e.CanExecute = dgrdSA.SelectedItems.Count != 0

&& \_validationErrorCount == 0

&& dgrdSA.SelectedItems.OfType<ISaltAnalysisCalcResults>().All(p=>p.IsCalculated);

}

private void SelectAllCommand\_CanExecute(object sender, CanExecuteRoutedEventArgs e)

{

e.CanExecute = dgrdSA.Items.Count != 0;

}

private void SelectAllCommand\_Executed(object sender, ExecutedRoutedEventArgs e)

{

if (!\_all\_selected) dgrdSA.SelectAll();

else dgrdSA.UnselectAll();

\_all\_selected = !\_all\_selected;

}

private bool ConfirmWhenSchemesDiffer()

{

if (dgrdSA.SelectedItems.Cast<ISaltAnalysisCalcResults>()

.Any(p => p.DefaultCalculationScheme != p.RecommendedCalculationScheme))

{

if (MessageBox.Show("В одном из выбранных для расчета образцов рекомендуемая схема расчета" +

" не совпадает с выбранной.\nПродолжить?", "Внимание", MessageBoxButton.YesNo,

MessageBoxImage.Hand, MessageBoxResult.No) == MessageBoxResult.No) return false;

}

return true;

}

private void ComboBox\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

((ISaltAnalysisCalcResults)((ComboBox)sender)?.DataContext).IsCalculated = false;

}

void dgrdSA\_CellEditEnding(object sender, DataGridCellEditEndingEventArgs e)

{

if (e?.EditAction == DataGridEditAction.Commit)

{

DataGridRow row = e.Row as DataGridRow;

if (row != null) (row.DataContext as ISaltAnalysisCalcResults).IsCalculated = false;

}

}

private void DataGridRow\_MouseDoubleClick(object sender, MouseButtonEventArgs e)

{

DataGridRow row = sender as DataGridRow;

if (row != null)

{

SADescriptionDialog sADescription = new SADescriptionDialog();

sADescription.Title = "Описание анализа";

ISaltAnalysisCalcResults analysisData = row.DataContext as ISaltAnalysisCalcResults;

sADescription.lblDescr.Content = "Введите описание анализа №: " + analysisData.LabNumber;

sADescription.Description = analysisData.AnalysisDescription;

sADescription.Topmost = true;

if (sADescription.ShowDialog() == true)

{

analysisData.AnalysisDescription = sADescription.Description;

CALogger.WriteToLogFile(string.Format("Введено описание для образца №{0} — {1}",

analysisData.LabNumber, analysisData.AnalysisDescription));

}

}

}

private void dgrdSA\_SourceUpdated(object sender, DataTransferEventArgs e)

{

if((dgrdSA?.CurrentItem as ISaltAnalysisCalcResults)!=null)

(dgrdSA?.CurrentItem as ISaltAnalysisCalcResults).IsCalculated = false;

}

private void CompareCommand\_CanExecute(object sender, CanExecuteRoutedEventArgs e)

{

e.CanExecute = dgrdSA.SelectedItems.Count == 2 &&

dgrdSA.SelectedItems.Cast<ISaltAnalysisCalcResults>().All(p => p.IsCalculated) &&

dgrdSA.SelectedItems.Cast<SaltAnalysisData>().Select(p=>p.DefaultCalculationScheme).Distinct().Count() == 1;

}

private void CompareCommand\_Executed(object sender, ExecutedRoutedEventArgs e)

{

decimal tolerance = 1;

SchemeResultsTolerance res = SchemeCompareOptionsHelper

.GetSchemeCompareOptions()[(dgrdSA.SelectedItem as ISaltAnalysisCalcResults).DefaultCalculationScheme];

string delimiter = string.Empty;

SaltAnalysisData res1 = dgrdSA.SelectedItems[0] as SaltAnalysisData;

SaltAnalysisData res2 = dgrdSA.SelectedItems[1] as SaltAnalysisData;

StringBuilder stringBuilder = new StringBuilder();

foreach (string prpName in SchemesHelper.GetPropertiesToCheck(res1.DefaultCalculationScheme))

{

string desc = string.Empty;

if (res.IsUniversalTolerance && res.UniversalTolerance.HasValue) tolerance = res.UniversalTolerance.Value;

else

{

try { tolerance = res.SchemeTolerances.Where(p => p.Item1 == prpName).FirstOrDefault().Item2.Value; }

catch { tolerance = 0.005M; }

}

decimal? v1 = (decimal?)res1.GetType().GetProperty(prpName).GetValue(res1);

decimal? v2 = (decimal?)res1.GetType().GetProperty(prpName).GetValue(res2);

if (v1.HasValue && v2.HasValue)

{

try

{

PropertyInfo pi = typeof(ISaltAnalysisCalcResults).GetRuntimeProperty(prpName);

if (pi == null) pi = typeof(ISaltAnalysisDryData).GetRuntimeProperty(prpName);

desc = ((CustomDescriptionAttribute)pi.GetCustomAttribute(typeof(CustomDescriptionAttribute)))

.Description;

}

catch (Exception ex)

{ desc = prpName; }

if (!(v1.Value == 0 && v2.Value == 0))

{

if(Math.Abs((v1.Value - v2.Value) / Math.Max(v1.Value, v2.Value)) > tolerance)

{

stringBuilder.Append(delimiter);

stringBuilder.Append($"Разница значений по параметру {desc} превышает толеранс.");

delimiter = ";" + Environment.NewLine;

}

else

{

stringBuilder.Append(delimiter);

stringBuilder.Append($"Параметр {desc} OK.");

delimiter = ";" + Environment.NewLine;

}

}

}

else if (v1.HasValue || v2.HasValue)

{

stringBuilder.Append(delimiter);

stringBuilder.Append($"Для параметра {desc} одно из значений NULL.");

delimiter = ";" + Environment.NewLine;

}

}

MessageBox.Show(stringBuilder.ToString(), "Результаты сравнения");

}

private void DuplicateCommand\_CanExecute(object sender, CanExecuteRoutedEventArgs e)

{ e.CanExecute = dgrdSA.SelectedItems.Count >= 1;}

private void DuplicateCommand\_Executed(object sender, ExecutedRoutedEventArgs e)

{

using (var context = new ChemicalAnalysesEntities())

{

foreach (SaltAnalysisData item in dgrdSA.SelectedItems)

{

var newItem = context.SaltAnalysisDatas.AsNoTracking()

.FirstOrDefault(p => p.IDSaltAnalysis == item.IDSaltAnalysis);

context.SaltAnalysisDatas.Add(newItem);

context.Entry(newItem).State = EntityState.Added;

}

context.SaveChanges();

}

FillData();

}

}

}

<Window x:Class="ChemicalAnalyses.Dialogs.SaltAnalysisDlg"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:alumni="clr-namespace:ChemicalAnalyses.Alumni"

xmlns:commands="clr-namespace:ChemicalAnalyses.Commands"

mc:Ignorable="d"

Title="{Binding}" Height="500" Width="1150" MinHeight="500" MinWidth="1050"

x:Name="wSAWindow" Validation.Error="OnErrorEvent">

<Window.CommandBindings>

<CommandBinding Command="Save" CanExecute="SaveCommand\_CanExecute" Executed="SaveCommand\_Executed"/>

<CommandBinding Command="Delete" CanExecute="DeleteCommand\_CanExecute" Executed="DeleteCommand\_Executed"/>

<CommandBinding Command="commands:CustomCommands.Calculate" CanExecute="CalculateCommand\_CanExecute" Executed="CalculateCommand\_Executed"/>

<CommandBinding Command="Print" Executed="PrintCommand\_Executed" CanExecute="PrintCommand\_CanExecute"/>

<CommandBinding Command="SelectAll" CanExecute="SelectAllCommand\_CanExecute" Executed="SelectAllCommand\_Executed"/>

</Window.CommandBindings>

<Window.Resources>

<alumni:SchemeToSchemeDescriptionConverter x:Key="sch2descConv"/>

<alumni:SchemeToSchemeDescriptionKVPairConverter x:Key="sch2descKVPairConv"/>

<alumni:BooleanToVisibilityConverter x:Key="bln2VisibilityConv"/>

<Style TargetType="{x:Type ToolTip}">

<Setter Property="ToolTipService.ShowOnDisabled" Value="True"/>

</Style>

</Window.Resources>

<Grid x:Name="grdMain">

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="30"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Grid.Resources>

<Style TargetType="{x:Type Button}">

<Setter Property="Width" Value="90"/>

<Setter Property="Height" Value="25"/>

</Style>

</Grid.Resources>

<ScrollViewer HorizontalScrollBarVisibility="Auto" Grid.Column="0" Grid.ColumnSpan="4" Grid.Row="0" x:Name="scroller"

VerticalScrollBarVisibility="Auto" CanContentScroll="True">

<DataGrid x:Name="dgrdSA" CanUserAddRows="False" TargetUpdated="dgrdSA\_SourceUpdated"

ItemsSource="{Binding Path=saltAnalysisDatas, UpdateSourceTrigger=PropertyChanged, ValidatesOnExceptions=True}"

AutoGenerateColumns="False" LoadingRow="dgrdSA\_LoadingRow">

<DataGrid.ColumnHeaderStyle>

<Style TargetType="{x:Type DataGridColumnHeader}">

<Setter Property="LayoutTransform">

<Setter.Value>

<RotateTransform Angle="270" />

</Setter.Value>

</Setter>

<Setter Property="MinHeight" Value="20"/>

</Style>

</DataGrid.ColumnHeaderStyle>

<DataGrid.RowValidationErrorTemplate>

<ControlTemplate>

<Grid Margin="0,-2,0,-2"

ToolTip="{Binding RelativeSource={RelativeSource FindAncestor,

AncestorType={x:Type DataGridRow}},

Path=(Validation.Errors)[0].ErrorContent}">

<Ellipse StrokeThickness="0" Fill="Red" Width="{TemplateBinding FontSize}" Height="{TemplateBinding FontSize}"/>

</Grid>

</ControlTemplate>

</DataGrid.RowValidationErrorTemplate>

<DataGrid.RowValidationRules>

<alumni:SaltAnalysisValidationRule ValidatesOnTargetUpdated="True" ValidationStep="RawProposedValue"/>

</DataGrid.RowValidationRules>

<DataGrid.Resources>

<Style TargetType="DataGridRow">

<EventSetter Event="MouseDoubleClick" Handler="DataGridRow\_MouseDoubleClick"/>

<Setter Property="ToolTip">

<Setter.Value>

<StackPanel Orientation="Vertical">

<TextBlock Text="{Binding Path=AnalysisDescription, UpdateSourceTrigger=PropertyChanged, NotifyOnSourceUpdated=True, TargetNullValue=Описание отсутствует}"/>

<TextBlock Text="Для изменения дважды кликните мышью"/>

</StackPanel>

</Setter.Value>

</Setter>

</Style>

</DataGrid.Resources>

<DataGrid.Columns>

<DataGridTextColumn Header="Лабораторный&#10;номер" Binding="{Binding

Path=LabNumber}" IsReadOnly="True"/>

<DataGridTemplateColumn Header="Дата&#10;анализа">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<AdornerDecorator>

<DatePicker Text="{Binding AnalysisDate, UpdateSourceTrigger=PropertyChanged,

Mode=TwoWay, NotifyOnValidationError=True, ValidatesOnExceptions=True,

NotifyOnTargetUpdated=True}" SelectedDateFormat="Short" Language="ru-RU"

ToolTip="Ввести дату анализа"/>

</AdornerDecorator>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

<DataGridTemplateColumn Header="Масса сырой&#10;навески, г">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<AdornerDecorator>

<TextBox Text="{Binding Path=WetWeight, NotifyOnValidationError=True, UpdateSourceTrigger=PropertyChanged, Mode=TwoWay, StringFormat={}{0:0.0000}, ValidatesOnExceptions=True, NotifyOnTargetUpdated=True}"/>

</AdornerDecorator>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

<DataGridTemplateColumn Header="Mg (титр),&#10;мл">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<AdornerDecorator>

<TextBox Text="{Binding Path=MagnesiumTitre, NotifyOnValidationError=True, UpdateSourceTrigger=PropertyChanged, Mode=TwoWay, StringFormat={}{0:0.00}, ValidatesOnExceptions=True, NotifyOnTargetUpdated=True}"/>

</AdornerDecorator>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

<DataGridTemplateColumn Header="(аликвота)&#10;Mg, мл">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<AdornerDecorator>

<TextBox Text="{Binding Path=MagnesiumAliquote, NotifyOnValidationError=True, UpdateSourceTrigger=PropertyChanged, Mode=TwoWay, StringFormat={}{0:0.0}, ValidatesOnExceptions=True, NotifyOnTargetUpdated=True}"/>

</AdornerDecorator>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

<DataGridTemplateColumn Header="Ca (титр),&#10;мл">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<AdornerDecorator>

<TextBox Text="{Binding Path=CalciumTitre, NotifyOnValidationError=True, UpdateSourceTrigger=PropertyChanged, Mode=TwoWay, StringFormat={}{0:0.00}, ValidatesOnExceptions=True, NotifyOnTargetUpdated=True}"/>

</AdornerDecorator>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

<DataGridTemplateColumn Header="(аликвота)&#10;Ca, мл">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<AdornerDecorator>

<TextBox Text="{Binding Path=CalciumAliquote, NotifyOnValidationError=True, UpdateSourceTrigger=PropertyChanged, Mode=TwoWay, StringFormat={}{0:0.0}, ValidatesOnExceptions=True, NotifyOnTargetUpdated=True}"/>

</AdornerDecorator>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

<DataGridTemplateColumn Header="K,&#10;показание">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<AdornerDecorator>

<TextBox Text="{Binding Path=KaliumValue, NotifyOnValidationError=True, UpdateSourceTrigger=PropertyChanged, Mode=TwoWay, StringFormat={}{0:0.0}, ValidatesOnExceptions=True, NotifyOnTargetUpdated=True}"/>

</AdornerDecorator>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

<DataGridTemplateColumn Header="K объём,&#10;мл">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<AdornerDecorator>

<TextBox Text="{Binding Path=KaliumVolume, NotifyOnValidationError=True, UpdateSourceTrigger=PropertyChanged, Mode=TwoWay, StringFormat={}{0:0.0}, ValidatesOnExceptions=True, NotifyOnTargetUpdated=True}"/>

</AdornerDecorator>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

<DataGridTemplateColumn Header="K, диапазон">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<AdornerDecorator>

<TextBox Text="{Binding Path=KaliumDiapason, NotifyOnValidationError=True, UpdateSourceTrigger=PropertyChanged, Mode=TwoWay, StringFormat={}{0:0}, ValidatesOnExceptions=True, NotifyOnTargetUpdated=True}"/>

</AdornerDecorator>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

<DataGridTemplateColumn Header="K, калибровка">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<Button Content="{Binding Path=KaliumCalibration, UpdateSourceTrigger=PropertyChanged, StringFormat={}{0:0}, NotifyOnTargetUpdated=True}" ToolTip="Нажмите, чтобы изменить/создать новую" Click="CalibrationButton\_Click" Width="25"/>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

<DataGridTemplateColumn Header="Г.Влажность,&#10;пустой бюкс,&#10;г">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<AdornerDecorator>

<TextBox Text="{Binding Path=HumidityCrucibleEmptyWeight, NotifyOnValidationError=True, UpdateSourceTrigger=PropertyChanged, Mode=TwoWay, StringFormat={}{0:0.0000}, ValidatesOnExceptions=True, NotifyOnTargetUpdated=True}"/>

</AdornerDecorator>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

<DataGridTemplateColumn Header="Г.Влажность,&#10;сырая навеска,&#10;г">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<AdornerDecorator>

<TextBox Text="{Binding Path=HumidityCrucibleWetSampleWeight, NotifyOnValidationError=True, UpdateSourceTrigger=PropertyChanged, Mode=TwoWay, StringFormat={}{0:0.0000}, ValidatesOnExceptions=True, NotifyOnTargetUpdated=True}"/>

</AdornerDecorator>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

<DataGridTemplateColumn Header="Г.Влажность,&#10;сухая навеска,&#10;110°, г">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<AdornerDecorator>

<TextBox Text="{Binding Path=HumidityCrucibleDry110SampleWeight, NotifyOnValidationError=True, UpdateSourceTrigger=PropertyChanged, Mode=TwoWay, StringFormat={}{0:0.0000}, ValidatesOnExceptions=True, NotifyOnTargetUpdated=True}"/>

</AdornerDecorator>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

<DataGridTemplateColumn Header="Г.Влажность,&#10;сухая навеска,&#10;180°, г">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<AdornerDecorator>

<TextBox Text="{Binding Path=HumidityCrucibleDry180SampleWeight, NotifyOnValidationError=True, UpdateSourceTrigger=PropertyChanged, Mode=TwoWay, StringFormat={}{0:0.0000}, ValidatesOnExceptions=True, NotifyOnTargetUpdated=True}"/>

</AdornerDecorator>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

<DataGridTemplateColumn Header="Cl титр,&#10;мл">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<AdornerDecorator>

<TextBox Text="{Binding Path=ChlorumTitre, NotifyOnValidationError=True, UpdateSourceTrigger=PropertyChanged, Mode=TwoWay, StringFormat={}{0:0.00}, ValidatesOnExceptions=True, NotifyOnTargetUpdated=True}"/>

</AdornerDecorator>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

<DataGridTemplateColumn Header="Cl (аликвота),&#10;мл">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<AdornerDecorator>

<TextBox Text="{Binding Path=ChlorumAliquote, NotifyOnValidationError=True, UpdateSourceTrigger=PropertyChanged, Mode=TwoWay, StringFormat={}{0:0.0}, ValidatesOnExceptions=True, NotifyOnTargetUpdated=True}"/>

</AdornerDecorator>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

<DataGridTemplateColumn Header="Br титр,&#10;мл">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<AdornerDecorator>

<TextBox Text="{Binding Path=BromumTitre, NotifyOnValidationError=True, UpdateSourceTrigger=PropertyChanged, Mode=TwoWay, StringFormat={}{0:0.0}, ValidatesOnExceptions=True, NotifyOnTargetUpdated=True}"/>

</AdornerDecorator>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

<DataGridTemplateColumn Header="Br (аликвота),&#10;мл">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<AdornerDecorator>

<TextBox Text="{Binding Path=BromumAliquote, NotifyOnValidationError=True, UpdateSourceTrigger=PropertyChanged, Mode=TwoWay, StringFormat={}{0:0.0}, ValidatesOnExceptions=True, NotifyOnTargetUpdated=True}"/>

</AdornerDecorator>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

<DataGridTemplateColumn Header="Н.О,&#10;бюкс, г">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<AdornerDecorator>

<TextBox Text="{Binding Path=ResiduumCrucibleEmptyWeight, NotifyOnValidationError=True, UpdateSourceTrigger=PropertyChanged, Mode=TwoWay, StringFormat={}{0:0.0000}, ValidatesOnExceptions=True, NotifyOnTargetUpdated=True}"/>

</AdornerDecorator>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

<DataGridTemplateColumn Header="Н.О,&#10;бюкс c&#10;навеской, г">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<AdornerDecorator>

<TextBox Text="{Binding Path=ResiduumCrucibleFullWeight, NotifyOnValidationError=True, UpdateSourceTrigger=PropertyChanged, Mode=TwoWay, StringFormat={}{0:0.0000}, ValidatesOnExceptions=True, NotifyOnTargetUpdated=True}"/>

</AdornerDecorator>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

<DataGridTemplateColumn>

<DataGridTemplateColumn.Header>

<TextBlock>

<Run>SO₄²¯</Run>

<LineBreak/>

<Run>вес тигля, г</Run>

</TextBlock>

</DataGridTemplateColumn.Header>

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<AdornerDecorator>

<TextBox Text="{Binding Path=SulfatesCrucibleEmptyWeight, NotifyOnValidationError=True, UpdateSourceTrigger=PropertyChanged, Mode=TwoWay, StringFormat={}{0:0.0000}, ValidatesOnExceptions=True, NotifyOnTargetUpdated=True}"/>

</AdornerDecorator>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

<DataGridTemplateColumn>

<DataGridTemplateColumn.Header>

<TextBlock>

<Run>SO₄²¯, тигель с</Run>

<LineBreak/>

<Run>осадком, г</Run>

</TextBlock>

</DataGridTemplateColumn.Header>

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<AdornerDecorator>

<TextBox Text="{Binding Path=SulfatesCrucibleFullWeight, NotifyOnValidationError=True, UpdateSourceTrigger=PropertyChanged, Mode=TwoWay, StringFormat={}{0:0.0000}, ValidatesOnExceptions=True, NotifyOnTargetUpdated=True}"/>

</AdornerDecorator>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

<DataGridTemplateColumn>

<DataGridTemplateColumn.Header>

<TextBlock >

<Run>SO₄²¯</Run><LineBreak/><Run>аликвота, мл</Run>

</TextBlock>

</DataGridTemplateColumn.Header>

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<AdornerDecorator>

<TextBox Text="{Binding Path=SulfatesAliquote, NotifyOnValidationError=True, UpdateSourceTrigger=PropertyChanged, Mode=TwoWay, StringFormat={}{0:0.0}, ValidatesOnExceptions=True, NotifyOnTargetUpdated=True}"/>

</AdornerDecorator>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

</DataGrid.Columns>

<DataGrid.RowDetailsTemplate>

<DataTemplate>

<AdornerDecorator> <!--Wrap in adoernerdecorator to preserve validation errors-->

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="auto"/>

<ColumnDefinition Width="auto"/>

</Grid.ColumnDefinitions>

<StackPanel Orientation="Horizontal" Grid.Row="0" Grid.Column="0">

<Label Content="Коэффициент Hg:"/>

<TextBox Text="{Binding Path=HgCoefficient, StringFormat={}{0:0.0000}, ValidatesOnExceptions=True, NotifyOnValidationError=True, UpdateSourceTrigger=PropertyChanged, NotifyOnTargetUpdated=True}"/>

<Label Content="мг-экв/мл; Холостой Br:"/>

<TextBox Text="{Binding Path=BromumBlank, StringFormat={}{0:0.0}, ValidatesOnExceptions=True, NotifyOnTargetUpdated=True, NotifyOnValidationError=True, UpdateSourceTrigger=PropertyChanged}"/>

<Label Content="мл; Стандартный титр Br:"/>

<TextBox Text="{Binding Path=BromumStandardTitre, StringFormat={}{0:0.0000}, NotifyOnValidationError=True, ValidatesOnExceptions=True, UpdateSourceTrigger=PropertyChanged , NotifyOnTargetUpdated=True}"/>

<Label Content=" мг-экв/мл"/>

<Label Content=" Рекомендуемая схема расчета: "/>

<TextBlock Text="{Binding Path=RecommendedCalculationScheme, UpdateSourceTrigger=PropertyChanged, NotifyOnTargetUpdated=True, Converter={StaticResource sch2descConv}}"/>

</StackPanel>

<StackPanel Orientation="Horizontal" Grid.Row="1" Grid.Column="0">

<Label Content="Холостой сульфаты:"/>

<TextBox Text="{Binding Path=SulfatesBlank, StringFormat={}{0:0.0000}, ValidatesOnNotifyDataErrors=True, UpdateSourceTrigger=PropertyChanged, ValidatesOnExceptions=True, NotifyOnTargetUpdated=True}"/>

<Label Content="мг; Трилон Б (Ca):"/>

<TextBox Text="{Binding Path=CalciumTrilonTitre, StringFormat={}{0:0.0000}, ValidatesOnExceptions=True, NotifyOnTargetUpdated=True, UpdateSourceTrigger=PropertyChanged, NotifyOnValidationError=True}"/>

<Label Content=" мг-экв/мл; Трилон Б (Mg):"/>

<TextBox Text="{Binding Path=MagnesiumTrilonTitre, StringFormat={}{0:0.0000}, ValidatesOnExceptions=True, NotifyOnTargetUpdated=True, UpdateSourceTrigger=PropertyChanged, NotifyOnValidationError=True}"/>

<Label Content=" мг-экв/мл; Предполагаемая схема расчета:"/>

<ComboBox SelectedItem="{Binding Path=DefaultCalculationScheme, ValidatesOnExceptions=True, UpdateSourceTrigger=PropertyChanged, NotifyOnValidationError=True, Mode=TwoWay, NotifyOnTargetUpdated=True, NotifyOnSourceUpdated=True, Converter={StaticResource sch2descKVPairConv}}" ItemsSource="{Binding RelativeSource={RelativeSource Mode=FindAncestor, AncestorType=Window}, Path=SchemesNames, Mode=TwoWay}" DisplayMemberPath="Value" SelectedValuePath="Key" SelectionChanged="ComboBox\_SelectionChanged" ToolTip="Выберите требуемую схему расчета, принимая во внимание предполагаемую.&#x0a;Данные буду обсчитаны с использованием выбранной вами схемы!"/>

<Label Content="\_Рассчитано:" Target="{Binding ElementName=cbIsCalculated}"/>

<CheckBox IsChecked="{Binding Path=IsCalculated, NotifyOnSourceUpdated=True}" x:Name="cbIsCalculated" IsEnabled="False"/>

</StackPanel>

<StackPanel Visibility="{Binding Path=IsCalculated, Converter={StaticResource bln2VisibilityConv}}" x:Name="spIonForm" Orientation="Horizontal" Grid.Row="1" Grid.Column="1">

<StackPanel.ToolTip>

<ToolTip>

<ToolTip.Content>

<StackPanel>

<StackPanel Orientation="Horizontal" Margin="1">

<Ellipse Width="16" Height="16" Fill="Green"/>

<TextBlock Text="— Сумма находится в пределах нормы"/>

</StackPanel>

<StackPanel Orientation="Horizontal">

<Ellipse Width="16" Height="16" Fill="Blue"/>

<TextBlock Text="— Значение суммы ниже допустимого предела"/>

</StackPanel>

<StackPanel Orientation="Horizontal">

<Ellipse Width="16" Height="16" Fill="Red"/>

<TextBlock Text="— Значение суммы выше допустимого предела"/>

</StackPanel>

</StackPanel>

</ToolTip.Content>

</ToolTip>

</StackPanel.ToolTip>

<Label Content="Ионная форма:" x:Name="lblIonicForm"/>

<Ellipse Width="16" Height="16" Fill="{Binding Path=IonSumColor, NotifyOnSourceUpdated=True}"/>

</StackPanel>

<StackPanel Visibility="{Binding Path=IsCalculated, Converter={StaticResource bln2VisibilityConv}}" x:Name="spSaltForm" Orientation="Horizontal" ToolTip="{Binding ElementName=spIonForm, Path=ToolTip}" Grid.Column="1" Grid.Row="0">

<Label Content="Солевая форма:" x:Name="lblSaltForm"/>

<Ellipse Width="16" Height="16" Fill="{Binding Path=SaltSumColor, NotifyOnSourceUpdated=True}"/>

</StackPanel>

</Grid>

</AdornerDecorator>

</DataTemplate>

</DataGrid.RowDetailsTemplate>

<DataGrid.ContextMenu>

<ContextMenu>

<ContextMenu.Resources>

<RoutedUICommand x:Key="cmdCompare" Text="Сравнить">

<RoutedUICommand.InputGestures>

<KeyGesture>SHIFT+ALT+C</KeyGesture>

</RoutedUICommand.InputGestures>

</RoutedUICommand>

<RoutedUICommand x:Key="cdmDuplicate" Text="Дублировать">

<RoutedUICommand.InputGestures>

<KeyGesture>SHIFT+ALT+D</KeyGesture>

</RoutedUICommand.InputGestures>

</RoutedUICommand>

</ContextMenu.Resources>

<MenuItem Header="Рассчитать результаты" Command="commands:CustomCommands.Calculate" ToolTip="Производит расчет по схеме, выбранной из выпадающего списка" x:Name="miCalculateResults"/>

<MenuItem Header="Удалить выбранные" Command="Delete" x:Name="miDeleteAnalyses"/>

<MenuItem Command="{StaticResource cmdCompare}" x:Name="imCompareResults">

<MenuItem.ToolTip>

<StackPanel Orientation="Vertical">

<TextBlock Text="Сравнить результаты расчета двух образцов." FontWeight="Black"/>

<TextBlock Text="Сравнение возможно только если:"/>

<TextBlock Text="1) выбрано 2 образа;"/>

<TextBlock Text="2) у выбранных образцов совпадают схемы расчета;"/>

<TextBlock Text="3) для выбранных образцов произведен расчет."/>

</StackPanel>

</MenuItem.ToolTip>

</MenuItem>

<MenuItem Command="{StaticResource cdmDuplicate}"/>

<ContextMenu.CommandBindings>

<CommandBinding Command="Delete" CanExecute="DeleteCommand\_CanExecute" Executed="DeleteCommand\_Executed"/>

<CommandBinding Command="{StaticResource cmdCompare}" CanExecute="CompareCommand\_CanExecute" Executed="CompareCommand\_Executed"/>

<CommandBinding Command="{StaticResource cdmDuplicate}" CanExecute="DuplicateCommand\_CanExecute" Executed="DuplicateCommand\_Executed"/>

</ContextMenu.CommandBindings>

</ContextMenu>

</DataGrid.ContextMenu>

</DataGrid>

</ScrollViewer>

<Button x:Name="btnOK" Content="OK" Grid.Column="0" Grid.Row="1" Command="Save" IsDefault="False"/>

<Button Content="Отмена" Grid.Column="1" Grid.Row="1" IsDefault="True" IsCancel="True"/>

<Button x:Name="btnDelete" Grid.Row="1" Grid.Column="2" Command="Delete" Content="Удалить" Visibility="{Binding}"/>

<Button x:Name="btnPrint" Grid.Row="1" Grid.Column="3" Command="Print" Content="Печать" Visibility="{Binding}"/>

</Grid>

</Window>

A.9 Класс диалогового окна ввода и редактирования описания анализа

using System;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

namespace ChemicalAnalyses.Dialogs

{

public partial class SADescriptionDialog : Window

{

private int errorCount = 0;

public string Description

{

get { return (string)GetValue(DescriptionProperty); }

set { if (value != null) SetValue(DescriptionProperty, value); }

}

public static readonly DependencyProperty DescriptionProperty =

DependencyProperty.Register(nameof(Description),

typeof(string), typeof(SADescriptionDialog),

new PropertyMetadata("Введите описание анализа не длинее 100 символов!"),

new ValidateValueCallback(validateDescriptionValue));

public SADescriptionDialog()

{

InitializeComponent();

grdMain.DataContext = this;

}

static bool validateDescriptionValue(object value)

{

if (value == null) return false;

return ((string)value).Length <= 100;

}

private void SaveCommand\_Executed(object sender, ExecutedRoutedEventArgs e)

{ DialogResult = true; }

private void SaveCommand\_CanExecute(object sender, CanExecuteRoutedEventArgs e)

{ e.CanExecute = errorCount == 0; }

private void Window\_ValidationError(object sender, ValidationErrorEventArgs e)

{

var validationEventArgs = e as ValidationErrorEventArgs;

if (validationEventArgs == null) throw new Exception("Unexpected event args");

switch (validationEventArgs.Action)

{

case ValidationErrorEventAction.Added:

{ errorCount++; break; }

case ValidationErrorEventAction.Removed:

{ errorCount--; break; }

default:

throw new Exception("Unknown action");

}

}

}

}

<Window x:Class="ChemicalAnalyses.Dialogs.SADescriptionDialog"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

mc:Ignorable="d"

Title="{Binding}" Height="150" Width="300" MaxWidth="350" MaxHeight="200"

Validation.Error="Window\_ValidationError">

<Window.CommandBindings>

<CommandBinding Command="Save" Executed="SaveCommand\_Executed" CanExecute="SaveCommand\_CanExecute"/>

</Window.CommandBindings>

<Grid x:Name="grdMain" Margin="2">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="35"/>

</Grid.RowDefinitions>

<Grid.Resources>

<Style TargetType="{x:Type Button}">

<Setter Property="Width" Value="90"/>

<Setter Property="Height" Value="25"/>

</Style>

</Grid.Resources>

<StackPanel Orientation="Vertical" Grid.Column="0" Grid.ColumnSpan="2" Grid.Row="0">

<Label Content="{Binding}" Target="{Binding ElementName=tbDescription}" x:Name="lblDescr"/>

<TextBox Text="{Binding Path=Description, NotifyOnValidationError=True, Mode=TwoWay, ValidatesOnExceptions=True, UpdateSourceTrigger=PropertyChanged}" x:Name="tbDescription" TextWrapping="WrapWithOverflow" VerticalScrollBarVisibility="Auto"/>

</StackPanel>

<Button Content="OK" HorizontalAlignment="Center" Grid.Row="1" Grid.Column="0" Command="Save" VerticalAlignment="Center" IsCancel="False" IsDefault="True"/>

<Button Content="Cancel" HorizontalAlignment="Center" Grid.Row="1" Grid.Column="1" VerticalAlignment="Center" IsCancel="True" IsDefault="False"/>

</Grid>

</Window>

A.10 Класс диалогового окна просмотра графика калибровочных прямых

using SA\_EF;

using System;

using System.Collections.ObjectModel;

using System.Linq;

using System.Windows;

namespace ChemicalAnalyses.Dialogs

{

public partial class CalibrationViewDialog : Window

{

public LinearCalibration lcCalibration

{

get { return (LinearCalibration)GetValue(lcCalibrationProperty); }

set { SetValue(lcCalibrationProperty, value); }

}

public static readonly DependencyProperty lcCalibrationProperty =

DependencyProperty.Register("lcCalibration",

typeof(LinearCalibration), typeof(CalibrationViewDialog),

new PropertyMetadata());

public CalibrationViewDialog(ref LinearCalibration lc)

{

InitializeComponent();

lcCalibration = lc;

grdMain.DataContext = this;

try

{lnSeries1.ItemsSource = new ObservableCollection<Tuple<decimal, decimal>>()

{

new Tuple<decimal, decimal>(0,lc.Intercept[0]),

new Tuple<decimal, decimal>(lc.LinearCalibrationData[0].Max(p=>p.Concentration),

lc.LinearCalibrationData[0].Max(p=>p.Concentration)\*lc.Slope[0]+

lc.Intercept[0])

};

lnSeries2.ItemsSource = new ObservableCollection<Tuple<decimal, decimal>>()

{

new Tuple<decimal, decimal>(0,lc.Intercept[1]),

new Tuple<decimal, decimal>(lc.LinearCalibrationData[1].Max(p=>p.Concentration),

lc.LinearCalibrationData[1].Max(p=>p.Concentration)\*lc.Slope[1]+

lc.Intercept[1])

};

}

catch { }

}

}

}

<Window x:Class="ChemicalAnalyses.Dialogs.CalibrationViewDialog"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:chartingToolkit="clr-namespace:System.Windows.Controls.DataVisualization.Charting;assembly=System.Windows.Controls.DataVisualization.Toolkit"

xmlns:chHlpr="clr-namespace:ChemicalAnalyses.Alumni"

mc:Ignorable="d"

Title="Просмотр параметров калибровки" Height="750" Width="625">

<Window.Resources>

<Style TargetType="{x:Type chartingToolkit:NumericAxisLabel}" x:Key="nmStyle">

<Setter Property="StringFormat" Value="{}{0:0.####}"/>

</Style>

<Style TargetType="{x:Type TextBlock}">

<Setter Property="Margin" Value="5"/>

</Style>

<Style TargetType="{x:Type chartingToolkit:LineDataPoint}" x:Key="pointStyle">

<Setter Property="Width" Value="0"/>

</Style>

</Window.Resources>

<Grid x:Name="grdMain">

<Grid.ColumnDefinitions>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="auto"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="auto"/>

</Grid.RowDefinitions>

<Grid Grid.Row="1">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="150"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<chartingToolkit:Chart Grid.Row="0" x:Name="chrChart1" chHlpr:ChartHelpers.IsLegendHidden="True" MinHeight="350">

<chartingToolkit:Chart.Title>

Первый диапазон

</chartingToolkit:Chart.Title>

<chartingToolkit:Chart.Axes>

<chartingToolkit:LinearAxis Orientation="X" x:Name="axConcentration1" AxisLabelStyle="{StaticResource nmStyle}" Title="Концентрация, мг/л"/>

<chartingToolkit:LinearAxis Orientation="Y" x:Name="axValue1" Title="Показания прибора"/>

</chartingToolkit:Chart.Axes>

<chartingToolkit:ScatterSeries IndependentValueBinding="{Binding Path=Concentration}" DependentValueBinding="{Binding Path=Value}" ItemsSource="{Binding Path=lcCalibration.LinearCalibrationData[0]}" x:Name="serScatterSeries1"/>

<chartingToolkit:LineSeries ItemsSource="{Binding}" x:Name="lnSeries1" DependentValuePath="Item2" IndependentValuePath="Item1" DataPointStyle="{StaticResource pointStyle}">

</chartingToolkit:LineSeries>

<chartingToolkit:Chart.ToolTip>

<TextBlock Text="Пример подсказки для графика"/>

</chartingToolkit:Chart.ToolTip>

</chartingToolkit:Chart>

<chartingToolkit:Chart Grid.Row="1" x:Name="chrChart2" chHlpr:ChartHelpers.IsLegendHidden="True" MinHeight="350" Title="Второй диапазон">

<chartingToolkit:Chart.Axes>

<chartingToolkit:LinearAxis Orientation="X" x:Name="axConcentration2" AxisLabelStyle="{StaticResource nmStyle}" Title="Концентрация, мг/л"/>

<chartingToolkit:LinearAxis Orientation="Y" x:Name="axValue2" Title="Показания прибора"/>

</chartingToolkit:Chart.Axes>

<chartingToolkit:ScatterSeries IndependentValueBinding="{Binding Path=Concentration}" DependentValueBinding="{Binding Path=Value}" ItemsSource="{Binding Path=lcCalibration.LinearCalibrationData[1]}" x:Name="serScatterSeries2"/>

<chartingToolkit:LineSeries ItemsSource="{Binding}" x:Name="lnSeries2" DependentValuePath="Item2" IndependentValuePath="Item1" DataPointStyle="{StaticResource pointStyle}">

</chartingToolkit:LineSeries>

</chartingToolkit:Chart>

<StackPanel Grid.Column="1" Grid.Row="0">

<TextBlock>

<Run>Y=</Run>

<Run Text="{Binding ElementName=grdMain, Path=DataContext.lcCalibration.Slope[0], StringFormat={}{0:0.0}}"/>

<Run>\*x+</Run>

<Run Text="{Binding ElementName=grdMain, Path=DataContext.lcCalibration.Intercept[0], StringFormat={}{0:0.00}}"/>

</TextBlock>

<TextBlock>

<Run>R</Run>

<Run Typography.Variants="Superscript">2</Run>

<Run>=</Run>

<Run Text="{Binding ElementName=grdMain, Path=DataContext.lcCalibration.RSquared[0], StringFormat={}{0:0.0000}}"/>

</TextBlock>

<TextBlock Text="Химический элемент:"/>

<TextBlock Text="{Binding Path=lcCalibration.CalibrationType}"/>

<TextBlock Text="Дата:"/>

<TextBlock Text="{Binding Path=lcCalibration.CalibrationDate, StringFormat={}{0:dd/MM/yyyy},

UpdateSourceTrigger=PropertyChanged}"/>

<TextBlock Text="Описание:"/>

<TextBlock Text="{Binding Path=lcCalibration.Description}" TextWrapping="WrapWithOverflow"/>

</StackPanel>

<StackPanel Grid.Column="1" Grid.Row="1">

<TextBlock>

<Run>Y=</Run>

<Run Text="{Binding ElementName=grdMain, Path=DataContext.lcCalibration.Slope[1], StringFormat={}{0:0.0}}"/>

<Run>\*x+</Run>

<Run Text="{Binding ElementName=grdMain, Path=DataContext.lcCalibration.Intercept[1], StringFormat={}{0:0.00}}"/>

</TextBlock>

<TextBlock>

<Run>R</Run>

<Run Typography.Variants="Superscript">2</Run>

<Run>=</Run>

<Run Text="{Binding ElementName=grdMain, Path=DataContext.lcCalibration.RSquared[1],

StringFormat={}{0:0.0000}}"/>

</TextBlock>

<TextBlock Text="Химический элемент:"/>

<TextBlock Text="{Binding Path=lcCalibration.CalibrationType}"/>

</StackPanel>

</Grid>

</Grid>

</Window>

A.11 Класс окна работы с калибровками

using EntityFrameworkExtras.EF6;

using SA\_EF;

using SettingsHelper;

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.ComponentModel;

using System.Data;

using System.Data.Entity;

using System.Diagnostics;

using System.Linq;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

namespace ChemicalAnalyses.Dialogs

{

public partial class CalibrationSelectionDlg : Window

{

public ObservableCollection<LinearCalibration> lcList { get; set; }

public int CalibrationNumber { get; set; }

string type { get; set; }

private ChemicalAnalysesEntities context;

public CalibrationSelectionDlg(string type="Kalium", int number = 0)

{

lcList = new ObservableCollection<LinearCalibration>();

InitializeComponent();

this.type = type;

CalibrationNumber = number;

context = new ChemicalAnalysesEntities();

#if DEBUG

context.Database.Log = s => { Debug.WriteLine(s); };

#endif

FillData();

Title = "Выбор калибровки для: " + type;

grdMain.DataContext = this;

//let tooltips be shown on disabled controls as well

try

{

ToolTipService.ShowOnDisabledProperty.OverrideMetadata(

typeof(Control),

new FrameworkPropertyMetadata(true));

}

catch { }//if it's already overridden

}

private void FillData()

{

lcList.Clear();

try

{

foreach (LinearCalibration clbr in context.LineaCalibrations

.Where(p => p.CalibrationType.Trim() == type))

lcList.Add(clbr);

}

catch(Exception ex) { }

try

{

cbLCSelection.SelectedIndex = lcList.IndexOf(lcList.First(p => p.CalibrationID == CalibrationNumber));

}

catch

{

cbLCSelection.SelectedIndex = 0;

}

}

private void EditCommand\_CanExecute(object sender, CanExecuteRoutedEventArgs e)

{ e.CanExecute = cbLCSelection?.SelectedItem != null;}

private void EditCommand\_Executed(object sender, ExecutedRoutedEventArgs e)

{

LinearCalibration lc = context.LineaCalibrations

.Find(((LinearCalibration)cbLCSelection.SelectedItem).CalibrationID);

if (lc != null) lc.GetLinearCoefficients();

else return;

CalibrationDataDialog cldDlg = new CalibrationDataDialog(ref lc);

if (cldDlg.ShowDialog() == true)

{

try

{

context.Database.BeginTransaction(IsolationLevel.Serializable);

context.Entry(lc).State = EntityState.Modified;

List<LCData> t = new List<LCData>();

for (int i = 0; i <= lc.LinearCalibrationData.Rank; i++)

{

lc.LinearCalibrationData[i].ToList().ForEach(p =>

{

t.Add(new LCData()

{

IDCalibration = lc.CalibrationID,

IDCalibrationData = p.IDCalibrationData,

Diapason = i + 1,

Concentration = p.Concentration,

Value = p.Value

});

});

}

context.Database.ExecuteStoredProcedure(new UpdateLCWithSP() { tmp = t });

context.SaveChanges();

context.Database.CurrentTransaction.Commit();

CALogger.WriteToLogFile(string.Format("Изменена калибровка ID{0};{1} - {2}", lc.CalibrationID, lc.Description, lc.CalibrationType)); }

catch (Exception ex)

{

context.Database.CurrentTransaction.Rollback();

MessageBox.Show(ex.Message + " в " + ex.Source, "Ошибка");

}

}

FillData();

}

private void ViewCommand\_CanExecute (object sender, CanExecuteRoutedEventArgs e)

{ e.CanExecute = cbLCSelection?.SelectedItem != null; }

private void ViewCommand\_Executed (object sender, ExecutedRoutedEventArgs e)

{

LinearCalibration lc = context.LineaCalibrations

.Find(((LinearCalibration)cbLCSelection.SelectedItem).CalibrationID);

if (lc != null) lc.GetLinearCoefficients();

else return;

CalibrationViewDialog cvDlg = new CalibrationViewDialog(ref lc);

cvDlg.Show(); //just to show it, no results are necessary

}

private void NewCommand\_CanExecute(object sender, CanExecuteRoutedEventArgs e)

{e.CanExecute = true;}

private void NewCommand\_Executed(object sender, ExecutedRoutedEventArgs e)

{

LinearCalibration lc = new LinearCalibration() { CalibrationType = type};

CalibrationDataDialog cldDlg = new CalibrationDataDialog(ref lc);

if (cldDlg.ShowDialog() == true)

{

CALogger.WriteToLogFile(string.Format("Создана калибровка {0} - {1}",

lc.Description , lc.CalibrationType.ToString()));

for (int i = 0; i <= lc.LinearCalibrationData.Rank; i++)

{

lc.LinearCalibrationData[i].ToList().ForEach(p =>

{

lc.CalibrationData.Add(new DataPoint

{

Concentration = p.Concentration,

Value = p.Value,

Diapason = i + 1

});

});

}

LinearCalibration newlc = context.LineaCalibrations.Add(lc);

context.SaveChanges();

FillData();

}

}

private void SetDefaultCommand\_CanExecute (object sender, CanExecuteRoutedEventArgs e)

{ e.CanExecute = cbLCSelection?.SelectedItem != null; }

private void SetDefaultCommand\_Executed(object sender, ExecutedRoutedEventArgs e)

{

CalibrationNumber = ((LinearCalibration)cbLCSelection.SelectedItem).CalibrationID;

CALogger.WriteToLogFile(string.Format("Установлена по умолчанию калибровка для {0} - {1}", type, ((LinearCalibration)cbLCSelection.SelectedItem).ToString()));}

private void CloseButton\_Click(object sender, RoutedEventArgs e)

{DialogResult = true;}

private void DeleteCommand\_CanExecute(object sender, CanExecuteRoutedEventArgs e)

{

try

{

if (cbLCSelection?.SelectedItem != null)

{

e.CanExecute = context.LineaCalibrations

.Find((cbLCSelection.SelectedItem as LinearCalibration).CalibrationID)?

.SaltAnalysis.Count == 0;

}

else e.CanExecute = false;

}

catch

{ e.CanExecute = false; }

btnDeleteCalibration.ToolTip = (e.CanExecute) ? "Удалить выбранную калибровку"

: "Удаление калибровки невозможно." + Environment.NewLine + "Присутствуют связанные данные.";

private void DeleteCommand\_Executed(object sender, ExecutedRoutedEventArgs e)

{

try

{

#if DEBUG

context.Database.Log = s => { Debug.WriteLine(s); };

#endif

CALogger.WriteToLogFile(string.Format("Удалена калибровка ID{0};{1} - {2}",

((LinearCalibration)cbLCSelection.SelectedItem).CalibrationID,

((LinearCalibration)cbLCSelection.SelectedItem).Description,

((LinearCalibration)cbLCSelection.SelectedItem).CalibrationType.ToString()));

context.Database.ExecuteStoredProcedure(new DeleteCalibrationByID()

{Calibration\_ID = ((LinearCalibration)cbLCSelection.SelectedItem).CalibrationID });

context.SaveChanges();

//}

FillData();

}

catch

{

MessageBox.Show("Не удалось удалить калибровку!" + Environment.NewLine + "Имеются связанные данные.");

}

}

protected override void OnClosing(CancelEventArgs e)

{

context?.Dispose();

base.OnClosing(e);

}

}

}

<Window x:Class="ChemicalAnalyses.Dialogs.CalibrationSelectionDlg"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:commands="clr-namespace:ChemicalAnalyses.Commands"

mc:Ignorable="d"

Title="{Binding}" Height="300" Width="400" MaxWidth="400" MinHeight="250">

<Window.Resources>

<Style TargetType="{x:Type Button}">

<Setter Property="Width" Value="90"/>

<Setter Property="Height" Value="25"/>

</Style>

</Window.Resources>

<Window.CommandBindings>

<CommandBinding Command="commands:CustomCommands.Edit" CanExecute="EditCommand\_CanExecute" Executed="EditCommand\_Executed"/>

<CommandBinding Command="commands:CustomCommands.View" CanExecute="ViewCommand\_CanExecute" Executed="ViewCommand\_Executed"/>

<CommandBinding Command="New" CanExecute="NewCommand\_CanExecute" Executed="NewCommand\_Executed"/>

<CommandBinding Command="commands:CustomCommands.SetDefault" CanExecute="SetDefaultCommand\_CanExecute" Executed="SetDefaultCommand\_Executed"/>

<CommandBinding Command="Delete" CanExecute="DeleteCommand\_CanExecute" Executed="DeleteCommand\_Executed"/>

</Window.CommandBindings>

<Grid x:Name="grdMain">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="35"/>

<RowDefinition Height="35"/>

</Grid.RowDefinitions>

<StackPanel Grid.Column="0" Grid.ColumnSpan="4">

<Label Content="\_Выбор подходящей калибровки из списка:" Target="{Binding ElementName=cbLCSelection}"/>

<ComboBox x:Name="cbLCSelection" Margin="10" Height="25" ItemsSource="{Binding Path=lcList}" ToolTip="Для совершения операций выберите калибровку из списка"/>

</StackPanel>

<Button Content="Сохранить" Grid.Column="0" Grid.Row="1" IsDefault="True" IsCancel="False" Click="CloseButton\_Click" x:Name="btnSaveCalibration"/>

<Button Content="Отмена" Grid.Column="1" Grid.Row="1" IsCancel="True"/>

<Button Content="{Binding}" Grid.Column="2" Grid.Row="1" Grid.ColumnSpan="2" Width="auto" Margin="4,0,4,0" Command="commands:CustomCommands.SetDefault" x:Name="btnSetDefault" ToolTip="{Binding}"/>

<Button Content="Изменить" Grid.Column="0" Grid.Row="2" Command="commands:CustomCommands.Edit" ToolTip="Изменить выбранную калибровку" x:Name="btnChangeCalibration"/>

<Button Content="Просмотр" Grid.Column="1" Grid.Row="2" Command="commands:CustomCommands.View" x:Name="btnViewCalibration" ToolTip="Просмотреть графическое представление и расчитанные параметры для выбранной калибровки"/>

<Button Content="Новая" Grid.Column="2" Grid.Row="2" Command="New" ToolTip="Создать новую калибровку" x:Name="btnNewCalibration"/>

<Button x:Name="btnDeleteCalibration" Content="Удалить" Grid.Column="3" Grid.Row="2" Command="Delete" ToolTip="{Binding}"/>

</Grid>

</Window>

A.12 Класс окна ввода и редактирования данных калибровки

using ChemicalAnalyses.Alumni;

using SA\_EF;

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Linq;

using System.Text;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

namespace ChemicalAnalyses.Dialogs

{

public partial class CalibrationDataDialog : Window

{

private int errorCount = 0;

private bool fdNoEqualDP = true;

private bool sdNoEqualDP = true;

public LinearCalibration lc { get; set; }

public static ObservableCollection<KeyValuePair<string, string>> elems { get; set; }

public CalibrationDataDialog (ref LinearCalibration calibration)

{

InitializeComponent();

lc = calibration;

lc.CalibrationType = lc.CalibrationType.Trim();

grdCalibrationDialog.DataContext = this;

if (elems == null) elems = new ObservableCollection<KeyValuePair<string, string>>

(Enum.GetValues(typeof(ChemicalElemetCalibration)).OfType<ChemicalElemetCalibration>()

.Select(p => new KeyValuePair<string, string>(p.ToString(), p.ToName())));

// if new is being created don't disable type selector

if (lc.CalibrationData.Count != 0) cbChemicalElemets.IsEnabled = false;

}

private void SaveCommand\_CanExecute(object sender, CanExecuteRoutedEventArgs e)

{

e.CanExecute = errorCount == 0 && (dgrdFirstDiapason.Items.Count > 2)

&& (dgrdSecondDiapason.Items.Count > 2) && fdNoEqualDP && sdNoEqualDP;

}

private void SaveCommand\_Executed(object sender, ExecutedRoutedEventArgs e)

{ DialogResult = true; }

private void OnErrorEvent(object sender, RoutedEventArgs e)

{

var validationEventArgs = e as ValidationErrorEventArgs;

if (validationEventArgs == null) throw new Exception("Unexpected event args");

switch (validationEventArgs.Action)

{

case ValidationErrorEventAction.Added:

{ errorCount++; break; }

case ValidationErrorEventAction.Removed:

{ errorCount--; break; }

default:

{ throw new Exception("Unknown action"); }

}

}

private void DataGrids\_CurrentCellChanged(object sender, EventArgs e)

{

if ((sender as DataGrid).Name == "dgrdFirstDiapason")

fdNoEqualDP = !lc.ContainsEqualDataPoints(0);

else sdNoEqualDP = !lc.ContainsEqualDataPoints(1);

}

}

}

<Window x:Class="ChemicalAnalyses.Dialogs.CalibrationDataDialog"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:alumni="clr-namespace:ChemicalAnalyses.Alumni"

mc:Ignorable="d"

Title="Данные калибровки" MinHeight="40" MinWidth="400" Height="400" Width="400"

Validation.Error="OnErrorEvent">

<Window.Resources>

<Style TargetType="{x:Type DataGrid}">

<Setter Property="CanUserAddRows" Value="True"/>

<Setter Property="CanUserReorderColumns" Value="False"/>

<Setter Property="AutoGenerateColumns" Value="False"/>

</Style>

<alumni:ElementToElementDescriptionKVPairConverter x:Key="elem2descKVPairConv"/>

</Window.Resources>

<Window.CommandBindings>

<CommandBinding Command="Save" CanExecute="SaveCommand\_CanExecute" Executed="SaveCommand\_Executed"/>

</Window.CommandBindings>

<Grid x:Name="grdCalibrationDialog">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="20"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

</Grid.RowDefinitions>

<Grid.Resources>

<Style TargetType="{x:Type DataGridColumnHeader}">

<Setter Property="HorizontalContentAlignment" Value="Center" />

</Style>

</Grid.Resources>

<TextBlock Text="Первый диапазон" Grid.Column="0" Grid.Row="0" HorizontalAlignment="Center"/>

<TextBlock Text="Второй диапазон" Grid.Column="1" Grid.Row="0" HorizontalAlignment="Center"/>

<DataGrid Grid.Column="0" Grid.Row="1" x:Name="dgrdFirstDiapason" ItemsSource="{Binding Path=lc.LinearCalibrationData[0], UpdateSourceTrigger=PropertyChanged}" CanUserAddRows="True" CurrentCellChanged="DataGrids\_CurrentCellChanged">

<DataGrid.Columns>

<DataGridTemplateColumn Header="Концентрация" Width="\*">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<TextBlock Text="{Binding Path=Concentration, StringFormat={}{0:0.0000 мг/л}}"/>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

<DataGridTemplateColumn.CellEditingTemplate>

<DataTemplate>

<TextBox Text="{Binding Path=Concentration, UpdateSourceTrigger=PropertyChanged, StringFormat={}{0:0.0000}, ValidatesOnExceptions=True, NotifyOnValidationError=True}"/>

</DataTemplate>

</DataGridTemplateColumn.CellEditingTemplate>

</DataGridTemplateColumn>

<DataGridTemplateColumn Width="\*" Header="Показания">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<TextBlock Text="{Binding Path=Value}"/>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

<DataGridTemplateColumn.CellEditingTemplate>

<DataTemplate>

<AdornerDecorator>

<TextBox Text="{Binding Path= Value, UpdateSourceTrigger=PropertyChanged, StringFormat={}{0:0.0}, ValidatesOnExceptions=True, NotifyOnValidationError=True}"/>

</AdornerDecorator>

</DataTemplate>

</DataGridTemplateColumn.CellEditingTemplate>

</DataGridTemplateColumn>

</DataGrid.Columns>

</DataGrid>

<DataGrid Grid.Column="1" Grid.Row="1" x:Name="dgrdSecondDiapason"

ItemsSource="{Binding Path=lc.LinearCalibrationData[1],

UpdateSourceTrigger=PropertyChanged}" CanUserAddRows="True"

CurrentCellChanged="DataGrids\_CurrentCellChanged">

<DataGrid.Columns>

<DataGridTemplateColumn Header="Концентрация" Width="\*">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<TextBlock Text="{Binding Path=Concentration, StringFormat={}{0:0.0000 мг/л}}"/>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

<DataGridTemplateColumn.CellEditingTemplate>

<DataTemplate>

<TextBox Text="{Binding Path=Concentration, UpdateSourceTrigger=PropertyChanged, StringFormat={}{0:0.0000}, ValidatesOnExceptions=True, NotifyOnValidationError=True}"/>

</DataTemplate>

</DataGridTemplateColumn.CellEditingTemplate>

</DataGridTemplateColumn>

<DataGridTemplateColumn Width="\*" Header="Показания">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<TextBlock Text="{Binding Path=Value}"/>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

<DataGridTemplateColumn.CellEditingTemplate>

<DataTemplate>

<AdornerDecorator>

<TextBox Text="{Binding Path= Value, UpdateSourceTrigger=PropertyChanged, StringFormat={}{0:0.0}, ValidatesOnExceptions=True, NotifyOnValidationError=True}"/>

</AdornerDecorator>

</DataTemplate>

</DataGridTemplateColumn.CellEditingTemplate>

</DataGridTemplateColumn>

</DataGrid.Columns>

</DataGrid>

<StackPanel Grid.Column="0" Grid.Row="2" Orientation="Horizontal" HorizontalAlignment="Center">

<Button Content="OK" IsDefault="True" IsCancel="False" Margin="5" Width="70" x:Name="btnOK" Command="Save"/>

<Button Content="Cancel" IsDefault="False" IsCancel="True" Margin="5" Width="70"/>

</StackPanel>

<ComboBox Margin="15,0,15,0" Grid.Column="1" Grid.Row="2" Height="20" x:Name="cbChemicalElemets" ItemsSource="{Binding RelativeSource={RelativeSource Mode=FindAncestor, AncestorType=Window}, Path=elems, Mode=TwoWay}" SelectedItem="{Binding Path=lc.CalibrationType, Mode=TwoWay, ValidatesOnExceptions=True, UpdateSourceTrigger=PropertyChanged, NotifyOnValidationError=True, Converter={StaticResource elem2descKVPairConv}}" DisplayMemberPath="Value" SelectedValuePath="Key" ToolTip="Тип калибровки можно выбрать из предложенного списка&#x0a;(Только на этапе создания)" ToolTipService.IsEnabled="True"/>

<StackPanel Grid.Row="3" Grid.Column="0" Grid.ColumnSpan="2">

<Label Content="\_Краткое описание: "/>

<TextBox x:Name="tbDescription" TextWrapping="WrapWithOverflow" Text="{Binding Path=lc.Description, UpdateSourceTrigger=PropertyChanged, Mode=TwoWay, NotifyOnTargetUpdated=True, ValidatesOnExceptions=True, NotifyOnValidationError=True}" ToolTip="Введите краткое описание, которое максимально точно характиризует калибровку.&#x0a;(Будет отображаться в списке)"/>

<StackPanel Orientation="Horizontal" Margin="0, 0, 0, 2">

<Label Content="\_Дата: " Target="{Binding ElementName=dpckCalibrationDate}"/>

<AdornerDecorator>

<DatePicker Text="{Binding lc.CalibrationDate, UpdateSourceTrigger=PropertyChanged, Mode=TwoWay, NotifyOnValidationError=True, ValidatesOnExceptions=True}" SelectedDateFormat="Short" Language="ru-RU" ToolTip="Ввести дату калибровки" x:Name="dpckCalibrationDate"/>

</AdornerDecorator>

</StackPanel>

</StackPanel>

</Grid>

</Window>

A.13 Класс, объявляющий пользовательские команды

using System.Windows.Input;

namespace ChemicalAnalyses.Commands

{

public static class CustomCommands

{

public static readonly RoutedUICommand Exit = new RoutedUICommand(

"Выход", Exit",

typeof(CustomCommands),

new InputGestureCollection

{new KeyGesture (Key.F4,ModifierKeys.Alt)});

public static readonly RoutedUICommand Options = new RoutedUICommand(

"Настройки калибровок", "Options",

typeof(CustomCommands),

new InputGestureCollection

{new KeyGesture (Key.O,ModifierKeys.Alt)});

public static readonly RoutedUICommand Edit = new RoutedUICommand(

"Редактировать", "Edit",

typeof(CustomCommands),

new InputGestureCollection

{new KeyGesture (Key.E,ModifierKeys.Control)});

public static readonly RoutedUICommand Update = new RoutedUICommand(

"Update", "Update",

typeof(CustomCommands),

new InputGestureCollection

{new KeyGesture (Key.U,ModifierKeys.Alt)});

public static readonly RoutedUICommand Filter = new RoutedUICommand(

"Filter", "Filter",

typeof(CustomCommands),

new InputGestureCollection

{new KeyGesture (Key.F,ModifierKeys.Alt)});

public static readonly RoutedUICommand List = new RoutedUICommand(

"List", "List",

typeof(CustomCommands),

new InputGestureCollection

{new KeyGesture (Key.L,ModifierKeys.Control)});

public static readonly RoutedUICommand View = new RoutedUICommand(

"View", "View",

typeof(CustomCommands),

new InputGestureCollection

{new KeyGesture (Key.V,ModifierKeys.Alt)});

public static readonly RoutedUICommand SetDefault = new RoutedUICommand(

"SetDefault", "SetDefault",

typeof(CustomCommands),

new InputGestureCollection

{new KeyGesture (Key.D,ModifierKeys.Alt)});

public static readonly RoutedUICommand Calculate = new RoutedUICommand(

"Calculate", "Calculate",

typeof(CustomCommands),

new InputGestureCollection

{new KeyGesture (Key.C,ModifierKeys.Alt)});

public static readonly RoutedUICommand AddNewAnalysis = new RoutedUICommand(

"Добавить новый анализ", "AddNewAnalysis",

typeof(CustomCommands));

}

}

A.14 Класс, реализующий правило валидации для класса, содержащего данныеанализов

using System;

using SA\_EF;

using System.Windows.Controls;

using System.Windows.Data;

namespace ChemicalAnalyses.Alumni

{

public class SaltAnalysisValidationRule : ValidationRule

{

public override ValidationResult Validate(object value,

System.Globalization.CultureInfo cultureInfo)

{

SaltAnalysisData slt = (value as BindingGroup).Items[0] as SaltAnalysisData;

try

{

SaltAnalysisData new\_s = new SaltAnalysisData

{

AnalysisDate = slt.AnalysisDate,

BromumAliquote = slt.BromumAliquote,

BromumBlank = slt.BromumBlank,

BromumStandardTitre = slt.BromumStandardTitre,

BromumTitre = slt.BromumTitre,

CalciumAliquote = slt.CalciumAliquote,

CalciumTitre = slt.CalciumTitre,

CalciumTrilonTitre = slt.CalciumTrilonTitre,

CarbonatesTitre = slt.CarbonatesTitre,

ChlorumAliquote = slt.ChlorumAliquote,

ChlorumTitre = slt.ChlorumTitre,

HgCoefficient = slt.HgCoefficient,

HumidityCrucibleEmptyWeight = slt.HumidityCrucibleEmptyWeight,

HumidityCrucibleWetSampleWeight = slt.HumidityCrucibleWetSampleWeight,

HumidityCrucibleDry110SampleWeight = slt.HumidityCrucibleDry110SampleWeight,

HumidityCrucibleDry180SampleWeight = slt.HumidityCrucibleDry180SampleWeight,

HydrocarbonatesTitre = slt.HydrocarbonatesTitre,

IDSample = slt.IDSample,

KaliumCalibration = slt.KaliumCalibration,

KaliumConcentration = slt.KaliumConcentration,

KaliumDiapason = slt.KaliumDiapason,

KaliumValue = slt.KaliumValue,

KaliumVolume = slt.KaliumVolume,

MagnesiumAliquote = slt.MagnesiumAliquote,

MagnesiumTitre = slt.MagnesiumTitre,

MagnesiumTrilonTitre = slt.MagnesiumTrilonTitre,

ResiduumCrucibleEmptyWeight = slt.ResiduumCrucibleEmptyWeight,

ResiduumCrucibleFullWeight = slt.ResiduumCrucibleFullWeight,

SulfatesAliquote = slt.SulfatesAliquote,

SulfatesBlank = slt.SulfatesBlank,

SulfatesCrucibleEmptyWeight = slt.SulfatesCrucibleEmptyWeight,

SulfatesCrucibleFullWeight = slt.SulfatesCrucibleFullWeight,

WetWeight = slt.WetWeight

};

return ValidationResult.ValidResult;

}

catch (Exception ex)

{

return new ValidationResult(false, ex.Message);

}

}

}

}

A.15 Класс-помощник для получения опций при сравнений результатов расчета

using SA\_EF;

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Linq;

namespace ChemicalAnalyses.Alumni

{

public static class SchemeCompareOptionsHelper

{

/// <summary>

/// Works for each scheme marked as realized (by custom attribute) in the corresponding enum options are grouped in an instance of special class options are got either from user-level settings or created anew

/// </summary>

/// <returns>Dictionary of key-value pairs scheme-scheme comparison options</returns>

public static IDictionary<SaltCalculationSchemes, SchemeResultsTolerance> GetSchemeCompareOptions()

{

Dictionary<SaltCalculationSchemes, SchemeResultsTolerance> dict =

new Dictionary<SaltCalculationSchemes, SchemeResultsTolerance>();

SchemeResultsTolerance sc;

foreach (var p in Enum.GetValues(typeof(SaltCalculationSchemes))

.OfType<SaltCalculationSchemes>().Where(p => p.GetAttribute<SchemeRealizedAttribute>() != null))

{

try

{

sc = new SchemeResultsTolerance(Properties.Settings.Default[p.ToString() + "\_SchemeToleranceValues"].ToString());

}

catch (Exception ex)

{

//No setting is present create new one

sc = new SchemeResultsTolerance()

{

IsUniversalTolerance = true,

UniversalTolerance = 0.005M,

SchemeTolerances = new ObservableCollection<ParameterValuePair>(

SchemesHelper.GetPropertiesToCheck(p)

.Select(r => new ParameterValuePair() { Item1 = r, Item2 = 0.005M }))

};

}

dict.Add( p, sc);

}

return dict;

}

}

}

A.16 Класс фильтра при отборе образцов

using System;

using System.Runtime.CompilerServices;

using System.ComponentModel;

namespace ChemicalAnalyses.Alumni

{

public class SampleFilterFields: INotifyPropertyChanged

{

private DateTime \_endDate = DateTime.Today;

public DateTime EndDate

{

get { return \_endDate; }

set

{

if (value > DateTime.Today) throw new ArgumentOutOfRangeException("EndDate",

"Конечная дата не может лежать в будущем!");

\_endDate = value;

OnPropertyChanged("EndDate");

}

}

private DateTime \_startDate = DateTime.Today.AddYears(-1);

public DateTime StartDate

{

get { return \_startDate; }

set

{

if (value > EndDate) throw new ArgumentOutOfRangeException("StartDate",

"Начальная дата не может находиться позже конечной!");

\_startDate = value;

OnPropertyChanged("StartDate");

}

}

private string \_labNumber;

public string LabNumber

{

get { return \_labNumber; }

set

{

\_labNumber = value;

OnPropertyChanged("LabNumber");

}

}

public event PropertyChangedEventHandler PropertyChanged;

public void OnPropertyChanged([CallerMemberName]string prop = "")

{

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(prop));

}

public override string ToString()

{

return "Начальная дата: " + StartDate.ToShortDateString() +

"\nКонечная дата: " + EndDate.ToShortDateString() +

(((LabNumber == null ) || string.Empty.Equals(LabNumber.Trim())) ? ""

: ("\nЛабораторный номер:" + LabNumber));

}

}

}

A.17 Классы конвертеров, используемых в окнах приложения

using SA\_EF;

using System;

using System.Collections.Generic;

using System.Globalization;

using System.Linq;

using System.Windows;

using System.Windows.Data;

namespace ChemicalAnalyses.Alumni

{

[ValueConversion(typeof(string),typeof(bool))]

public class StringToBooleanConverter : IValueConverter

{

public object ConvertBack(object value, Type targetType,

object parameter, CultureInfo culture)

{throw new NotImplementedException("Backward conversion is not possible");}

public object Convert(object value, Type targetType,

object parameter, CultureInfo culture)

{

if ((string)value == "Not filtered") return false;

return true;

}

}

[ValueConversion(typeof(object[]),typeof(bool))]

public class SampleAvConverter : IMultiValueConverter

{

public object Convert(object[] values, Type targetType, object parameter, CultureInfo culture)

{

string labnm;

string descr;

DateTime smplDate;

NumberFormatInfo nfi = new NumberFormatInfo();

nfi.NumberDecimalSeparator = ".";

try

{

labnm = values[0].ToString().Trim();

smplDate = DateTime.Parse(values[1].ToString());

descr = values[2].ToString().Trim();

Sample smpl = new Sample

{

IDSample =1, //Dummy just to test the other fields

LabNumber = labnm,

SamplingDate = smplDate,

Description = descr

};

return true;

}

catch { return false; }

}

public object[] ConvertBack(object value, Type[] targetTypes, object parameter, CultureInfo culture)

{throw new NotSupportedException("Обратная конвертация невозможна!");}

}

[ValueConversion(typeof(string), typeof(double?))]

public class StringToNullableDoubleConverter : IValueConverter

{

public object ConvertBack(object value, Type targetType,

object parameter, CultureInfo culture)

{

double d;

object o;

string sep = culture.NumberFormat.NumberDecimalSeparator;

if (value.ToString().Trim() == "") return null;

try

{

d = double.Parse(value.ToString());

return o = d;

}

catch

{return null;}

}

public object Convert(object value, Type targetType,

object parameter, CultureInfo culture)

{

if (value == null) return "";

return value.ToString();

}

}

[ValueConversion(typeof(decimal), typeof(decimal))]

public class DoubleToPercentageConverter : IValueConverter

{

public object ConvertBack(object value, Type targetType,

object parameter, CultureInfo culture)

{throw new NotImplementedException("Backward conversion is not possible");}

public object Convert(object value, Type targetType,

object parameter, CultureInfo culture)

{

if (value is null) return null;

return (decimal)value \* 100;

}

}

public class DecimalsToSumConverter : IMultiValueConverter

{

public object Convert(object[] values, Type targetType, object parameter, CultureInfo culture)

{

if (values == null) return null;

decimal res = 0;

for (int i = 0; i < values.Length; i++)

{

if (values[i] != null)

{

try

{res += (decimal)values[i];}

catch { }

}

}

return res \* 100;

}

public object[] ConvertBack(object value, Type[] targetTypes, object parameter, CultureInfo culture)

{throw new NotImplementedException("Backward conversion is not possible");}

}

[ValueConversion(typeof(bool), typeof(Visibility))]

public class BooleanToVisibilityConverter : IValueConverter

{

public object ConvertBack(object value, Type targetType,

object parameter, CultureInfo culture)

{

if (value != null && (Visibility)value == Visibility.Visible) return true;

else return false;

}

public object Convert(object value, Type targetType,

object parameter, CultureInfo culture)

{

if (value is null) return Visibility.Hidden;

return ((bool)value) ? (Visibility.Visible) : (Visibility.Collapsed);

}

}

[ValueConversion(typeof(bool), typeof(bool))]

public class BooleanToNegatedBooleanConverter : IValueConverter

{

public object ConvertBack(object value, Type targetType,

object parameter, CultureInfo culture)

{ return !(bool)value; }

public object Convert(object value, Type targetType,

object parameter, CultureInfo culture)

{ return !(bool)value; }

}

[ValueConversion(typeof(bool), typeof(string))]

public class BooleanToUserTypeConverter : IValueConverter

{

public object ConvertBack(object value, Type targetType,

object parameter, CultureInfo culture)

{return false;}

public object Convert(object value, Type targetType,

object parameter, CultureInfo culture)

{

if (value is null) return "Пользователь";

return ((bool)value) ? ("Администратор") : ("Пользователь");

}

}

[ValueConversion(typeof(bool), typeof(Visibility))]

public class BooleanToVisibilityNegativeConverter : IValueConverter

{

public object ConvertBack(object value, Type targetType,

object parameter, CultureInfo culture)

{

if (value == null || (Visibility)value != Visibility.Visible) return true;

else return false;

}

public object Convert(object value, Type targetType,

object parameter, CultureInfo culture)

{

if (value is null || !(bool)value) return Visibility.Visible;

else return Visibility.Collapsed;

}

}

[ValueConversion(typeof(SaltCalculationSchemes), typeof(Visibility),

ParameterType =typeof(SaltCalculationSchemes))]

public class SchemeToVisibilityConverter : IValueConverter

{

public object ConvertBack(object value, Type targetType,

object parameter, CultureInfo culture)

{throw new NotImplementedException("Backward conversion is not possible");}

public object Convert(object value, Type targetType,

object parameter, CultureInfo culture)

{

if (value == null || parameter == null) return Visibility.Collapsed;

if ((SaltCalculationSchemes)value == (SaltCalculationSchemes)parameter) return Visibility.Visible;

return Visibility.Collapsed;

}

}

[ValueConversion(typeof(object[]),typeof(Visibility))]

public class SchemeToVisibilityMultipleConverter : IMultiValueConverter

{

public object Convert(object[] values, Type targetType, object parameter, CultureInfo culture)

{

if (values == null || values[0] == null) return Visibility.Hidden;

if (values.Skip(1).Any(p => (SaltCalculationSchemes)(values[0]) == (SaltCalculationSchemes)p))

return Visibility.Visible;

return Visibility.Hidden;

}

public object[] ConvertBack(object value, Type[] targetTypes, object parameter, CultureInfo culture)

{throw new NotImplementedException("Backward conversion is not possible");}

}

[ValueConversion(typeof(SaltCalculationSchemes), typeof(string))]

public class SchemeToSchemeDescriptionConverter : IValueConverter

{

public object ConvertBack(object value, Type targetType,

object parameter, CultureInfo culture)

{throw new NotImplementedException("Backward conversion not implemented");}

public object Convert(object value, Type targetType,

object parameter, CultureInfo culture)

{

if (value != null) return ((SaltCalculationSchemes)value).ToName();

return null;

}

}

[ValueConversion(typeof(SaltCalculationSchemes), typeof(KeyValuePair<SaltCalculationSchemes, string>))]

public class SchemeToSchemeDescriptionKVPairConverter : IValueConverter

{

public object ConvertBack(object value, Type targetType,

object parameter, CultureInfo culture)

{

if (value !=null)

return ((KeyValuePair<SaltCalculationSchemes, string>)value).Key;

return null;

}

public object Convert(object value, Type targetType,

object parameter, CultureInfo culture)

{

if (value != null) return new KeyValuePair<SaltCalculationSchemes,string>

((SaltCalculationSchemes)value, ((SaltCalculationSchemes)value).ToName());

return null;

}

}

[ValueConversion(typeof(string), typeof(KeyValuePair<string, string>))]

public class ElementToElementDescriptionKVPairConverter : IValueConverter

{

public object ConvertBack(object value, Type targetType,

object parameter, CultureInfo culture)

{

if (value != null)

return ((KeyValuePair<string, string>)value).Key;

return null;

}

public object Convert(object value, Type targetType,

object parameter, CultureInfo culture)

{

try

{

if (value != null) return new KeyValuePair<string, string>

((string)value,

((ChemicalElemetCalibration)Enum.Parse(typeof(ChemicalElemetCalibration),

((string)value).Trim())).ToName());

}

catch { return null; }

return null;

}

}

}

A.18 Класс-помощник для скрытия легенды на графике калибровки

using System.Windows.Controls.DataVisualization.Charting;

using System.Windows.Controls.DataVisualization;

using System.Windows;

using System.ComponentModel;

namespace ChemicalAnalyses.Alumni

{

public static class ChartHelpers

{

static ChartHelpers()

{

HideLegendStyle = new Style(typeof(Legend));

HideLegendStyle.Setters.Add(new Setter(Legend.WidthProperty, 0.0));

HideLegendStyle.Setters.Add(new Setter(Legend.HeightProperty, 0.0));

HideLegendStyle.Setters.Add(new Setter(Legend.VisibilityProperty, Visibility.Collapsed));}

/// <summary>Gets a <see cref="Style"/> to hide the legend.</summary>

public static readonly Style HideLegendStyle;

#region IsLegendHidden

[Category("Common")]

[AttachedPropertyBrowsableForType(typeof(Chart))]

public static bool GetIsLegendHidden(Chart chart)

{

return (bool)chart.GetValue(IsLegendHiddenProperty);

}

public static void SetIsLegendHidden(Chart chart, bool value)

{

chart.SetValue(IsLegendHiddenProperty, value);

}

public static readonly DependencyProperty IsLegendHiddenProperty =

DependencyProperty.RegisterAttached(

"IsLegendHidden",

typeof(bool), // type

typeof(ChartHelpers), // containing static class

new PropertyMetadata(default(bool), OnIsLegendHiddenChanged)

);

private static void OnIsLegendHiddenChanged(DependencyObject d, DependencyPropertyChangedEventArgs e)

{ OnIsLegendHiddenChanged((Chart)d, (bool)e.NewValue); }

private static void OnIsLegendHiddenChanged(Chart chart, bool isHidden)

{

if (isHidden)

{ chart.LegendStyle = HideLegendStyle; }

}

#endregion IsLegendHidden

}

}

A.19 Класс, служащий посредником при связывании WPF в случае, если контекст данных не наследуется

using System.Windows;

namespace ChemicalAnalyses.Alumni

{

public class BindingProxy : Freezable

{

public static readonly DependencyProperty DataProperty =

DependencyProperty.Register("Data", typeof(object),

typeof(BindingProxy), new UIPropertyMetadata(null));

public object Data

{

get { return GetValue(DataProperty); }

set { SetValue(DataProperty, value); }

}

#region Overrides of Freezable

protected override Freezable CreateInstanceCore()

{ return new BindingProxy(); }

#endregion

}

}

A.20 Класс-расширение функционала стандартного класса Enum в части работы с атрибутами

using System;

using System.ComponentModel;

using System.Collections.Generic;

namespace ChemicalAnalyses.Alumni

{

public static class EnumExtensions

{

//Taken from <https://stackoverflow.com/questions/1799370/getting-attributes-of-enums-value> response of Troy Alford

public static T GetAttribute<T>(this Enum value) where T : Attribute

{

var type = value.GetType();

var memberInfo = type.GetMember(value.ToString());

var attributes = memberInfo[0].GetCustomAttributes(typeof(T), false);

if(attributes.Length!=0) return (T)attributes[0];

return null;

}

// This method creates a specific call to the above method, requesting the

// Description MetaData attribute.

public static string ToName(this Enum value)

{

var attribute = value.GetAttribute<DescriptionAttribute>();

return attribute == null ? value.ToString() : attribute.Description;

}

public static T GetValueFromDescription<T>(this string description)

{

var type = typeof(T);

if (!type.IsEnum) throw new InvalidOperationException();

foreach (var field in type.GetFields())

{

var attribute = Attribute.GetCustomAttribute(field,

typeof(DescriptionAttribute)) as DescriptionAttribute;

if (attribute != null)

{

if (attribute.Description == description)

return (T)field.GetValue(null);

}

else

{

if (field.Name == description)

return (T)field.GetValue(null);

}

}

throw new ArgumentException("Not found.", "description");

}

}

}

A.21 Класс-расширение функционала стандартного интерфейса IEnumerable, позволяющий осуществлять разбиение на участки заданного размера

using System.Collections.Generic;

using System.Linq;

namespace ChemicalAnalyses.Alumni

{

public static class IEnumerableExtensions

{

//Taken from https://stackoverflow.com/questions/419019/split-list-into-sublists-with-linq

public static IEnumerable<IEnumerable<T>> Chunk<T>(this IEnumerable<T> source, int chunksize)

{

int pos = 0;

while (source.Skip(pos).Any())

{

yield return source.Skip(pos).Take(chunksize);

pos += chunksize;

}

}

}

}

A.22 Класс контекста БД

using System;

using System.Text.RegularExpressions;

using System.Data.Entity;

using System.Configuration;

using System.Data.Entity.Infrastructure;

using System.Data.Entity.Infrastructure.Interception;

namespace SA\_EF

{

[DbConfigurationType(typeof(FE6CodeConfig))]

public partial class ChemicalAnalysesEntities : DbContext

{

private static string \_userName = "";

public static string UserName { set { if (value != null) \_userName = value; } }

private static string \_password = "";

public static string Password { set { if (value != null) \_password = value; } }

private static bool \_areUserNameAndPwdSet = false;

public static bool AreUserNameAndPwdSet { get {return \_areUserNameAndPwdSet; } }

private static bool \_isAdmin = false;

public static bool IsAdmin { get { return \_isAdmin; } }

public static string connectionString { private get; set; } = "name=CAEntities";

private static DbConnectionApplicationRoleInterceptor dbConnInterceptor;

public ChemicalAnalysesEntities(bool relogin = false) :base(ConnectionStringRebuilder(connectionString))

{

if (relogin || !\_areUserNameAndPwdSet)

{

if (dbConnInterceptor != null) DbInterception.Remove(dbConnInterceptor);

dbConnInterceptor = new DbConnectionApplicationRoleInterceptor(\_userName, \_password);

DbInterception.Add(dbConnInterceptor);

DbConnectionApplicationRoleInterceptor.AppRoleTreatment += OnAppRoleTreatment;

}

}

protected void OnAppRoleTreatment(object sender, AppRoleTreatmentEventArgs e)

{

if (e != null)

{

\_areUserNameAndPwdSet = e.HasAppRolePassed;

\_isAdmin = e.IsMemberOfAdmin;

}

}

protected static string ConnectionStringRebuilder(string connectionString)

{

string connstr = "";

string certName = "";

string pattern = @"(.+data source=)(\S+==)(.+user id=)(\S+[^;])(;password=)(\S+==)(.+)";

Regex regex = new Regex(pattern);

try {

connstr = ConfigurationManager.OpenExeConfiguration(ConfigurationUserLevel.None).

ConnectionStrings.ConnectionStrings["CAEntities"].ToString();

certName = Properties.Settings.Default.CertificateName;

}

catch (Exception ex) { }

MatchCollection match = regex.Matches(connstr);

if (match.Count == 1 || match[0].Groups.Count == 6)

{

X509EncDec cert = new X509EncDec(certName);

string substitution = @"$1 " + cert.DecryptRsa(match[0].Groups[2].Value)

+ "$3" + cert.DecryptRsa(match[0].Groups[4].Value) + "$5"

+ cert.DecryptRsa(match[0].Groups[6].Value) + "$7";

return regex.Replace(connstr, substitution);

}

return connectionString;

}

protected override void OnModelCreating(DbModelBuilder modelBuilder)

{ throw new UnintentionalCodeFirstException();}

public virtual DbSet<LinearCalibration> LineaCalibrations { get; set; }

public virtual DbSet<DataPoint> DataPoints { get; set; }

public virtual DbSet<SaltAnalysisData> SaltAnalysisDatas { get; set; }

public virtual DbSet<CalibrationType> CalibrationType { get; set; }

public virtual DbSet<Sample> Samples { get; set; }

}

}

A.23 Класс, осуществляющий перехват событий установления и разрыва соединений с БД

using System;

using System.Data.Entity.Infrastructure.Interception;

using System.Data.Common;

using System.Data.SqlClient;

using System.Data;

using System.Diagnostics;

namespace SA\_EF

{

public class DbConnectionApplicationRoleInterceptor : IDbConnectionInterceptor

{

private readonly string \_appRole;

private readonly string \_password;

private byte[] \_cookie;

private static bool \_isSetApproleExecuted = false;

public DbConnectionApplicationRoleInterceptor(){}

public DbConnectionApplicationRoleInterceptor(string appRole, string password)

{

\_appRole = appRole;

\_password = password;

}

public void Opened(DbConnection connection, DbConnectionInterceptionContext interceptionContext)

{

if (connection.State != ConnectionState.Open) return;

ActivateApplicationRole(connection, \_appRole, \_password);

}

public void Closing(DbConnection connection, DbConnectionInterceptionContext interceptionContext)

{

if (connection.State != ConnectionState.Open) return;

DeActivateApplicationRole(connection, \_cookie);

}

public virtual void ActivateApplicationRole(DbConnection dbConn, string appRoleName, string password)

{

if (dbConn == null)

throw new ArgumentNullException("DbConnection");

if (ConnectionState.Open != dbConn.State)

throw new InvalidOperationException("DBConnection must be opened before activating application role");

if (string.IsNullOrWhiteSpace(appRoleName))

throw new ArgumentNullException("appRoleName");

if (password == null)

throw new ArgumentNullException("password");

SetApplicationRole(dbConn, appRoleName, password);

}

private void SetApplicationRole(DbConnection dbConn, string appRoleName, string password)

{

if (!\_isSetApproleExecuted) {

using (var cmd = dbConn.CreateCommand())

{

cmd.CommandType = CommandType.StoredProcedure;

cmd.CommandText = "sp\_setapprole";

cmd.Parameters.Add(new SqlParameter("@rolename", appRoleName));

cmd.Parameters.Add(new SqlParameter("@password", password));

cmd.Parameters.Add(new SqlParameter("@fCreateCookie", SqlDbType.Bit) { Value = true });

var cookie = new SqlParameter("@cookie", SqlDbType.Binary, 50)

{Direction = ParameterDirection.InputOutput};

cmd.Parameters.Add(cookie);

try { cmd.ExecuteNonQuery(); }

catch (Exception ex) { }

int res = 0;

if (cookie.Value != null

//Check if cookie value is not 0xFFFFFFFF == ERROR

&& BitConverter.ToInt32((byte[])cookie.Value, 0) != -1)

{

\_cookie = (byte[])cookie.Value;

\_isSetApproleExecuted = true;

cmd.CommandType = CommandType.Text;

cmd.CommandText = "SELECT IS\_MEMBER(N'db\_accessadmin')";

try { res = (int)cmd.ExecuteScalar(); }

catch { }

}

OnAppRoleTreatment(new AppRoleTreatmentEventArgs(\_isSetApproleExecuted, res == 1 &&

\_isSetApproleExecuted));

}

}

}

public virtual void DeActivateApplicationRole(DbConnection dbConn, byte[] cookie)

{

if (\_isSetApproleExecuted)

{

using (var cmd = dbConn.CreateCommand())

{

cmd.CommandText = "sp\_unsetapprole";

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.Add(new SqlParameter("@cookie", SqlDbType.VarBinary, 50) { Value = cookie });

Debug.WriteLine("ExecutingNonQuery to Unset Application Role");

try

{ cmd.ExecuteNonQuery();}

catch (Exception ex) {}

}

\_isSetApproleExecuted = false;

}

}

#region Other DbConnection Interception

public void BeganTransaction(DbConnection connection,

BeginTransactionInterceptionContext interceptionContext){ }

public void BeginningTransaction(DbConnection connection,

BeginTransactionInterceptionContext interceptionContext){ }

public void Closed(DbConnection connection, DbConnectionInterceptionContext interceptionContext) { }

public void ConnectionStringGetting(DbConnection connection,

DbConnectionInterceptionContext<string> interceptionContext) {}

public void ConnectionStringGot(DbConnection connection,

DbConnectionInterceptionContext<string> interceptionContext) {}

public void ConnectionStringSet(DbConnection connection,

DbConnectionPropertyInterceptionContext<string> interceptionContext){ }

public void ConnectionStringSetting(DbConnection connection,

DbConnectionPropertyInterceptionContext<string> interceptionContext){ }

public void ConnectionTimeoutGetting(DbConnection connection,

DbConnectionInterceptionContext<int> interceptionContext){ }

public void ConnectionTimeoutGot(DbConnection connection,

DbConnectionInterceptionContext<int> interceptionContext){ }

public void DataSourceGetting(DbConnection connection,

DbConnectionInterceptionContext<string> interceptionContext){ }

public void DataSourceGot(DbConnection connection,

DbConnectionInterceptionContext<string> interceptionContext){ }

public void DatabaseGetting(DbConnection connection,

DbConnectionInterceptionContext<string> interceptionContext){ }

public void DatabaseGot(DbConnection connection,

DbConnectionInterceptionContext<string> interceptionContext){ }

public void Disposed(DbConnection connection, DbConnectionInterceptionContext interceptionContext) { }

public void Disposing(DbConnection connection, DbConnectionInterceptionContext interceptionContext) { }

public void EnlistedTransaction(DbConnection connection,

EnlistTransactionInterceptionContext interceptionContext){ }

public void EnlistingTransaction(DbConnection connection,

EnlistTransactionInterceptionContext interceptionContext){ }

public void Opening(DbConnection connection, DbConnectionInterceptionContext interceptionContext){}

public void ServerVersionGetting(DbConnection connection,

DbConnectionInterceptionContext<string> interceptionContext){ }

public void ServerVersionGot(DbConnection connection,

DbConnectionInterceptionContext<string> interceptionContext){ }

public void StateGetting(DbConnection connection,

DbConnectionInterceptionContext<ConnectionState> interceptionContext){ }

public void StateGot(DbConnection connection,

DbConnectionInterceptionContext<ConnectionState> interceptionContext){ }

#endregion

public static event AppRoleTreatmentEventHandler AppRoleTreatment;

protected virtual void OnAppRoleTreatment (AppRoleTreatmentEventArgs e)

{

AppRoleTreatment?.Invoke(this, e);

}

}

public class AppRoleTreatmentEventArgs : EventArgs

{

private readonly bool \_hasAppRolePassed = false;

private readonly bool \_isMemberOfAdmin = false;

public AppRoleTreatmentEventArgs(bool hasApprolePassed, bool isMemberOfAdmin)

{

\_isMemberOfAdmin = isMemberOfAdmin;

\_hasAppRolePassed = hasApprolePassed;

}

public bool HasAppRolePassed

{ get { return \_hasAppRolePassed; } }

public bool IsMemberOfAdmin

{ get { return \_isMemberOfAdmin; } }

}

public delegate void AppRoleTreatmentEventHandler(object sender, AppRoleTreatmentEventArgs e);

}

A.24 Сущность «Образец»

using System;

using System.Runtime.CompilerServices;

using System.ComponentModel;

namespace SA\_EF

{

public partial class Sample : INotifyPropertyChanged

{

private string \_labnumber;

public string LabNumber

{

get { return \_labnumber; }

set

{

if (value == null || value.Equals(String.Empty))

throw new ArgumentNullException("LabNumber", "Номер не может быть пустой строкой," +

"\nтак как используется для идентификации");

if (value?.Length > 15 || value?.Length < 2)

throw new ArgumentOutOfRangeException("LabNumber", "Неверный формат номера!");

if (value.Contains(";"))

throw new ArgumentOutOfRangeException("LabNumber", "Номер не может содержать символ ';'!");

\_labnumber = value;

OnPropertyChanged("LabNumber");

}

}

private DateTime \_samplingdate = DateTime.Today;

public DateTime SamplingDate

{

get { return \_samplingdate; }

set

{

if (value > DateTime.Now)

throw new ArgumentOutOfRangeException("SamplingDate", "Дата отбора не может лежать в будущем!");

\_samplingdate = value;

OnPropertyChanged("SamplingDate");

}

}

private string \_desc;

public string Description

{

get { return \_desc; }

set

{

if (value == null || value.Equals(string.Empty))

throw new ArgumentNullException("Description", "Описание не может быть пустым");

if (value?.Length > 200) throw new ArgumentOutOfRangeException("Description", "Слишком длинная строка!");

\_desc = value;

OnPropertyChanged("Description");

}

}

public override string ToString()

{

return String.Format("Лабораторный номер: {0}, Дата отбора: {1:dd-MM-yyy}\n Описание: {2}",

LabNumber, SamplingDate, Description);

}

public event PropertyChangedEventHandler PropertyChanged;

public void OnPropertyChanged([CallerMemberName]string prop = "")

{

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(prop));

}

}

}

A.25 Сущность «Анализ» (часть свойств, хранимых в БД)

using System;

using System.Runtime.CompilerServices;

using System.ComponentModel;

using System.ComponentModel.DataAnnotations.Schema;

namespace SA\_EF

{

public partial class SaltAnalysisData : INotifyPropertyChanged

{

private decimal \_wetWeight = 4;

public decimal WetWeight

{

get { return \_wetWeight; }

set

{

if (value <= 0) throw new ArgumentOutOfRangeException("WetWeight",

"Значение сырой навески не может быть отрицательным числом!");

\_wetWeight = value;

OnPropertyChanged("WetWeight");

}

}

private decimal \_magnesiumTitre = 1;

public decimal MagnesiumTitre

{

get { return \_magnesiumTitre; }

set

{

if (value <= 0) throw new ArgumentOutOfRangeException("MagnesiumTitre",

"Значение титра не может быть отрицательным числом");

\_magnesiumTitre = value;

OnPropertyChanged("MagnesiumTitre");

}

}

private decimal \_magnesiumAliquote = 50;

public decimal MagnesiumAliquote

{

get { return \_magnesiumAliquote; }

set

{

if (value <= 0) throw new ArgumentOutOfRangeException("MagnesiumAliquote",

"Значение аликвоты не может быть отрицательным числом");

\_magnesiumAliquote = value;

OnPropertyChanged("MagnesiumAliquote");

}

}

private decimal \_magnesiumTrilonTitre = 1;

public decimal MagnesiumTrilonTitre

{

get { return \_magnesiumTrilonTitre; }

set

{

if (value <= 0) throw new ArgumentOutOfRangeException("MagnesiumTrilonTitre",

"Значение титра не может быть отрицательным числом");

\_magnesiumTrilonTitre = value;

OnPropertyChanged("MagnesiumTrilonTitre");

}

}

private int \_kaliumCalibration = 1;

public int KaliumCalibration

{

get { return \_kaliumCalibration; }

set

{

\_kaliumCalibration = value;

OnPropertyChanged("KaliumCalibration");

}

}

private DateTime \_analysisdate = DateTime.Today;

public DateTime AnalysisDate

{

get { return \_analysisdate; }

set

{

if (value > DateTime.Now)

throw new ArgumentOutOfRangeException("AnalysisDate",

"Дата анализа не может лежать в будущем!");

\_analysisdate = value;

OnPropertyChanged("AnalysisDate");

}

}

private int \_kaliumDiapason = 1;

public int KaliumDiapason

{

get { return \_kaliumDiapason; }

set

{

if (!(value == 1 || value == 2))

throw new ArgumentOutOfRangeException("KaliumDiapason",

"Значение диапазона 1 или 2");

\_kaliumDiapason = value;

OnPropertyChanged("KaliumDiapason");

}

}

private Nullable<decimal> \_kaliumConcentration;

public Nullable<decimal> KaliumConcentration

{

get { return \_kaliumConcentration; }

set

{

\_kaliumConcentration = value;

OnPropertyChanged("KaliumConcentration");

}

}

private decimal \_sulfatesCrucibleEmptyWeight = 10;

public decimal SulfatesCrucibleEmptyWeight

{

get { return \_sulfatesCrucibleEmptyWeight; }

set

{

if (value <= 0) throw new ArgumentOutOfRangeException("SulfatesCrucibleEmptyWeight", "Значение веса пустого тигля не может быть отрицательным числом!");

\_sulfatesCrucibleEmptyWeight = value;

OnPropertyChanged("SulfatesCrucibleEmptyWeight");

}

}

private decimal \_sulfatesCrucibleFullWeight = 12;

public decimal SulfatesCrucibleFullWeight

{

get { return \_sulfatesCrucibleFullWeight; }

set

{

if (value <= 0) throw new ArgumentOutOfRangeException("SulfatesCrucibleFullWeight", "Значение веса тигля с осадком не может быть отрицательным числом!");

if (value <= SulfatesCrucibleEmptyWeight) throw new ArgumentOutOfRangeException("SulfatesCrucibleFullWeight", "Значение веса тигля с осадком не может быть меньшим или равным весу пустого тигля!");

\_sulfatesCrucibleFullWeight = value;

OnPropertyChanged("SulfatesCrucibleFullWeight");

}

}

private decimal \_residuumCrucibleFullWeight = 15;

public decimal ResiduumCrucibleFullWeight

{

get { return \_residuumCrucibleFullWeight; }

set

{

if (value <= 0) throw new ArgumentOutOfRangeException("ResiduumCrucibleFullWeight", "Значение веса тигля с осадком не может быть отрицательным числом!");

if (value <= ResiduumCrucibleEmptyWeight) throw new ArgumentOutOfRangeException("ResiduumCrucibleFullWeight", "Значение веса тигля с осадком не может быть меньше или равно весу пустого тигля!");

\_residuumCrucibleFullWeight = value;

OnPropertyChanged("ResiduumCrucibleFullWeight");

}

}

private decimal \_residuumCrucibleEmptyWeight = 10;

public decimal ResiduumCrucibleEmptyWeight

{

get { return \_residuumCrucibleEmptyWeight; }

set

{

if (value <= 0) throw new ArgumentOutOfRangeException("ResiduumCrucibleEmptyWeight", "Значение веса пустого бюкса не может быть отрицательным числом!");

\_residuumCrucibleEmptyWeight = value;

OnPropertyChanged("ResiduumCrucibleEmptyWeight");

}

}

private decimal \_humidityCrucibleEmptyWeight = 10;

public decimal HumidityCrucibleEmptyWeight

{

get { return \_humidityCrucibleEmptyWeight; }

set

{ if (value <= 0) throw new ArgumentOutOfRangeException("HumidityCrucibleEmptyWeight", "Значение веса пустого тигля не может быть отрицательным числом!");

\_humidityCrucibleEmptyWeight = value;

OnPropertyChanged("HumidityCrucibleEmptyWeight");

}

}

private decimal \_humidityCrucibleWetSampleWeight = 15;

public decimal HumidityCrucibleWetSampleWeight

{

get { return \_humidityCrucibleWetSampleWeight; }

set

{

if (value <= 0) throw new ArgumentOutOfRangeException("HumidityCrucibleWetSampleWeight", "Значение веса тигля с сырой навеской не может быть отрицательным числом!");

if (value <= HumidityCrucibleEmptyWeight) throw new ArgumentOutOfRangeException("HumidityCrucibleWetSampleWeight", "Значение веса тигля с сырой навеской не может быть меньшим или равным весу пустого тигля!");

\_humidityCrucibleWetSampleWeight = value;

OnPropertyChanged("HumidityCrucibleWetSampleWeight");

}

}

private decimal \_humidityCrucibleDry110SampleWeight = 14;

public decimal HumidityCrucibleDry110SampleWeight

{

get { return \_humidityCrucibleDry110SampleWeight; }

set

{

if (value <= 0) throw new ArgumentOutOfRangeException("HumidityCrucibleDry110SampleWeight", "Значение веса тигля с сухой (110) навеской не может быть равным нулю или отрицательным числом!");

if (value <= HumidityCrucibleEmptyWeight) throw new ArgumentOutOfRangeException("HumidityCrucibleDry110SampleWeight", "Значение веса тигля с сухой (110) навеской не может быть меньшим или равным весу пустого тигля!");

if (value > HumidityCrucibleWetSampleWeight) throw new ArgumentOutOfRangeException("HumidityCrucibleDry110SampleWeight", "Значение веса тигля с сухой (110) навеской не может быть большим веса тигля с сырой навеской!");

\_humidityCrucibleDry110SampleWeight = value;

OnPropertyChanged("HumidityCrucibleDry110SampleWeight");

}

}

private decimal? \_humidityCrucibleDry180SampleWeight;

public decimal? HumidityCrucibleDry180SampleWeight

{

get { return this.\_humidityCrucibleDry180SampleWeight; }

set

{

if (value != null)

{

if (value < 0) throw new ArgumentOutOfRangeException("HumidityCrucibleDry180SampleWeight", "Значение веса тигля с сухой (180) навеской не может быть равным нулю или отрицательным числом!");

if (value <= HumidityCrucibleEmptyWeight) throw new ArgumentOutOfRangeException("HumidityCrucibleDry180SampleWeight", "Значение веса тигля с сухой (180) навеской не может быть меньшим или равным весу пустого тигля!");

if (value > HumidityCrucibleWetSampleWeight) throw new ArgumentOutOfRangeException("HumidityCrucibleDry180SampleWeight", "Значение веса тигля с сухой (180) навеской не может быть большим веса тигля с сырой навеской!");

if (value > HumidityCrucibleDry110SampleWeight) throw new ArgumentOutOfRangeException("HumidityCrucibleDry180SampleWeight", "Значение веса тигля с сухой (180) навеской не может быть большим веса тигля с навеской при 110!");

\_humidityCrucibleDry180SampleWeight = value;

OnPropertyChanged("HumidityCrucibleDry180SampleWeight");

}

else \_humidityCrucibleDry180SampleWeight = null;

OnPropertyChanged("HumidityCrucibleDry180SampleWeight");

}

}

private decimal \_calciumTitre = 5;

public decimal CalciumTitre

{

get { return \_calciumTitre; }

set

{

if (value <= 0) throw new ArgumentOutOfRangeException("CalciumTitre", "Значение титра не может быть отрицательным числом");

\_calciumTitre = value;

OnPropertyChanged("CalciumTitre");

}

}

private decimal \_chlorumTitre = 3;

public decimal ChlorumTitre

{

get { return \_chlorumTitre; }

set

{

if (value <= 0) throw new ArgumentOutOfRangeException("ChlorumTitre",

"Значение титра не может быть отрицательным числом");

\_chlorumTitre = value;

OnPropertyChanged("ChlorumTitre");

}

}

private decimal \_bromumTitre = 7;

public decimal BromumTitre

{

get { return \_bromumTitre; }

set

{

if (value <= 0) throw new ArgumentOutOfRangeException("BromumTitre", "Значение титра не может быть отрицательным числом");

\_bromumTitre = value;

OnPropertyChanged("BromumTitre");

}

}

private decimal \_kaliumValue = 10;

public decimal KaliumValue

{

get { return \_kaliumValue; }

set

{

if (value <= 0) throw new ArgumentOutOfRangeException("KaliumValue",

"Значение показаний не может быть отрицательным числом");

\_kaliumValue = value;

OnPropertyChanged("KaliumValue");

}

}

private decimal \_carbonatesTitre = 0;

public decimal CarbonatesTitre

{

get { return \_carbonatesTitre; }

set

{

if (value < 0) throw new ArgumentOutOfRangeException("CarbonatesTitre",

"Значение титра не может быть отрицательным числом");

\_carbonatesTitre = value;

OnPropertyChanged("CarbonatesTitre");

}

}

private decimal \_hydrocarbonatesTitre = 0;

public decimal HydrocarbonatesTitre

{

get { return \_hydrocarbonatesTitre; }

set

{

if (value < 0) throw new ArgumentOutOfRangeException("HydrocarbonatesTitre",

"Значение титра не может быть отрицательным числом");

\_hydrocarbonatesTitre = value;

OnPropertyChanged("HydrocarbonatesTitre");

}

}

private decimal \_calciumTrilonTitre = 1;

public decimal CalciumTrilonTitre

{

get { return \_calciumTrilonTitre; }

set

{

if (value <= 0) throw new ArgumentOutOfRangeException("CalciumTrilonTitre",

"Значение титра не может быть отрицательным числом");

\_calciumTrilonTitre = value;

OnPropertyChanged("CalciumTrilonTitre");

}

}

private decimal \_calciumAliquote = 50;

public decimal CalciumAliquote

{

get { return \_calciumAliquote; }

set

{

if (value <= 0) throw new ArgumentOutOfRangeException("CalciumAliquote",

"Значение аликвоты не может быть отрицательным числом");

\_calciumAliquote = value;

OnPropertyChanged("CalciumAliquote");

}

}

private decimal \_chlorumAliquote = 5;

public decimal ChlorumAliquote

{

get { return \_chlorumAliquote; }

set

{

if (value <= 0) throw new ArgumentOutOfRangeException("ChlorumAliquote",

"Значение аликвоты не может быть отрицательным числом");

\_chlorumAliquote = value;

OnPropertyChanged("ChlorumAliquote");

}

}

private decimal \_bromumAliquote = 50;

public decimal BromumAliquote

{

get { return \_bromumAliquote; }

set

{

if (value <= 0) throw new ArgumentOutOfRangeException("BromumAliquote",

"Значение аликвоты не может быть отрицательным числом");

\_bromumAliquote = value;

OnPropertyChanged("BromumAliquote");

}

}

private decimal \_sulfatesAliquote = 100;

public decimal SulfatesAliquote

{

get { return \_sulfatesAliquote; }

set

{

if (value <= 0) throw new ArgumentOutOfRangeException("SulfatesAliquote",

"Значение аликвоты не может быть отрицательным числом");

\_sulfatesAliquote = value;

OnPropertyChanged("SulfatesAliquote");

}

}

private decimal \_kaliumVolume = 1;

public decimal KaliumVolume

{

get { return \_kaliumVolume; }

set

{

if (value <= 0) throw new ArgumentOutOfRangeException("KaliumVolume",

"Значение аликвоты не может быть отрицательным числом");

\_kaliumVolume = value;

OnPropertyChanged("KaliumVolume");

}

}

private decimal \_hgCoefficient;

public decimal HgCoefficient

{

get { return \_hgCoefficient; }

set

{

if (value <= 0) throw new ArgumentOutOfRangeException("HgCoefficient",

"Значение параметра не может быть отрицательным либо равным 0!");

\_hgCoefficient = value;

OnPropertyChanged("HgCoefficient");

}

}

private decimal \_bromumBlank = 1;

public decimal BromumBlank

{

get { return \_bromumBlank; }

set

{

if (value < 0) throw new ArgumentOutOfRangeException("BromumBlank",

"Значение не может быть отрицательным!");

\_bromumBlank = value;

OnPropertyChanged("BromumBlank");

}

}

private decimal \_bromumStandardTitre = (decimal)0.1332;

public decimal BromumStandardTitre

{

get { return \_bromumStandardTitre; }

set

{

if (value <= 0) throw new ArgumentOutOfRangeException("BromumStandardTitre",

"Значение не может быть отрицательным!");

\_bromumStandardTitre = value;

OnPropertyChanged("BromumStandardTitre");

}

}

private decimal \_sulfatesBlank = (decimal)0.001;

public decimal SulfatesBlank

{

get { return \_sulfatesBlank; }

set

{

if (value < 0) throw new ArgumentOutOfRangeException("SulfatesBlank",

"Значение не может быть отрицательным!");

\_sulfatesBlank = value;

OnPropertyChanged("SulfatesBlank");

}

}

private SaltCalculationSchemes \_defaultClaculationScheme =

SaltCalculationSchemes.Chloride;

[NotMapped]

public SaltCalculationSchemes DefaultCalculationScheme

{

get { return \_defaultClaculationScheme; }

set

{

if (!(value == SaltCalculationSchemes.Chloride

|| value == SaltCalculationSchemes.SulfateSodiumI

|| value == SaltCalculationSchemes.SulfateMagnesiumI))

throw new NotImplementedException(На данный момент доступны только хлоридная, сульфатно-натриевая(I) и сульфатно-магниевая(I) схемы");

\_defaultClaculationScheme = value;

OnPropertyChanged("DefaultCalculationScheme");

}

}

private SaltCalculationSchemes \_recommendedCalculationScheme = SaltCalculationSchemes.Chloride;

[NotMapped]

public SaltCalculationSchemes RecommendedCalculationScheme

{

get { return \_recommendedCalculationScheme; }

set

{

\_recommendedCalculationScheme = value;

OnPropertyChanged("RecommendedCalculationScheme");

}

}

[NotMapped]

public string LabNumber { get; set; } //just to show in the datagrid

public event PropertyChangedEventHandler PropertyChanged;

public void OnPropertyChanged([CallerMemberName]string prop = "")

{

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(prop));

}

}

}

A.26 Сущность «Анализ» (часть расчетных свойств и методов расчета), а также перечисление, описывающее возможные расчетные схемы

using SA\_EF.Interfaces;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.ComponentModel.DataAnnotations.Schema;

using System.Configuration;

using System.Globalization;

using System.IO;

using System.Reflection;

using System.Windows.Media;

namespace SA\_EF

{

public partial class SaltAnalysisData : ISaltAnalysisCalculation, ISaltAnalysisDryData, ISaltAnalysisCalcResults

{

#region DrywWeights

[NotMapped]

public decimal MgWet { get; set; }

[NotMapped]

public decimal MgDry { get; set; }

[NotMapped]

public decimal CaDry { get; set; }

[NotMapped]

public decimal ClDry { get; set; }

[NotMapped]

public decimal BrDry { get; set; }

[NotMapped]

public decimal ResiduumDry { get; set; }

[NotMapped]

public decimal SulfatesDry { get; set; }

[NotMapped]

public decimal SampleCorrectedDryWeight { get; set; }

[NotMapped]

public decimal CarbonatesDry { get; set; }

[NotMapped]

public decimal HydrocarbonatesDry { get; set; }

[NotMapped]

public decimal HumidityContent { get; set; }

[NotMapped]

public decimal KDry { get; set; }

#endregion DryWeights

static IDictionary<int, ILinearCalibration> lcDict;

//Application Level settings holding atomic weights of chemical elements used in calculation

static ClientSettingsSection elementsWeights;

static ClientSettingsSection userSettings;

#region SchemeResults

[NotMapped]

public decimal CaSO4 { get; set; }

[NotMapped]

public decimal Na { get; set; }

[NotMapped]

public decimal? CrystWater { get; set; }

[NotMapped]

public decimal Carnallite { get; set; }

[NotMapped]

public decimal NaCl { get; set; }

[NotMapped]

public decimal KCl { get; set; }

[NotMapped]

public decimal CaCl2 { get; set; }

[NotMapped]

public decimal? MgCl2 { get; set; }

[NotMapped]

public decimal MgCl2AnyCase { get; set; }

[NotMapped]

public decimal KBr { get; set; }

[NotMapped]

public decimal? HygroWater { get; set; }

[NotMapped]

public decimal HygroWaterAnyCase { get; set; }

[NotMapped]

public decimal CaHCO3\_2 { get; set; }

[NotMapped]

public decimal MgSO4 { get; set; }

[NotMapped]

public decimal Na2SO4 { get; set; }

[NotMapped]

public decimal NaBr { get; set; }

private bool \_isCalculated = false;

[NotMapped]

public bool IsCalculated

{

get { return \_isCalculated; }

set

{

\_isCalculated =value;

OnPropertyChanged("IsCalculated");

}

}

private decimal \_ionSum = 0;

[NotMapped]

public decimal IonSum

{

get { return \_ionSum; }

}

private decimal \_saltSum = 0;

[NotMapped]

public decimal SaltSum

{

get { return \_saltSum; }

}

private SolidColorBrush \_ionSumColor;

[NotMapped]

public SolidColorBrush IonSumColor

{

get { return \_ionSumColor; }

set

{

\_ionSumColor = value;

OnPropertyChanged("IonSumColor");

}

}

private SolidColorBrush \_saltSumColor;

[NotMapped]

public SolidColorBrush SaltSumColor

{

get {return \_saltSumColor; }

set

{

\_saltSumColor = value;

OnPropertyChanged("SaltSumColor");

}

}

#endregion SchemeResults

#region AtomicWeights\_Of\_ChemicalElememts

static decimal awMg;

static decimal awCa;

static decimal awH;

static decimal awO;

static decimal awCl;

static decimal awNa;

static decimal awK;

static decimal awC;

static decimal awS;

static decimal awBr;

static decimal awB;

#endregion AtomicWeights\_Of\_ChemicalElememts

#region Application level calculated consts

private static decimal \_water2MagnesiumRatioInCarnallite;

private static decimal \_SO4\_2\_CaS04;

private static decimal \_CaSO4\_2\_SO4;

private static decimal \_CaCl2\_2\_Ca;

private static decimal \_MgCl2\_2\_Mg;

private static decimal \_KCl\_2\_K;

private static decimal \_NaCl\_2\_Cl;

private static decimal \_KBr\_2\_Br;

private static decimal \_Carnallite\_2\_Magnesium;

private static decimal \_Eq\_CO3;

private static decimal \_Eq\_HCO3;

private static decimal \_Eq\_Mg;

private static decimal \_Eq\_Ca;

private static decimal \_Eq\_SO4;

private static decimal \_Ca\_2\_HCO3;

private static decimal \_SO4\_2\_Ca;

private static decimal \_SO4\_2\_Mg;

private static decimal \_Na\_2\_SO4;

//May be changed at the user level so need to hold for each instance separately

private decimal carnalliteThreshold = 0.0008M;

#endregion

[NotMapped]

public string AnalysisDescription { get; set; }

private static decimal sumTolerance;

private static SolidColorBrush redBrush, greenBrush, blueBrush;

public SaltAnalysisData()

{

if (lcDict is null) lcDict = new Dictionary<int, ILinearCalibration>();

if (elementsWeights is null)

{

Uri UriAssemblyFolder = new Uri(Path.GetDirectoryName(Assembly.GetExecutingAssembly().GetName().CodeBase));

string appPath = UriAssemblyFolder.LocalPath;

//Open the configuration file and retrieve the applicationSettings section.

try

{

Configuration config = ConfigurationManager.OpenExeConfiguration(appPath + @"\ChemicalAnalyses.exe");

elementsWeights = (ClientSettingsSection)config.SectionGroups["applicationSettings"].Sections[0];

userSettings = (ClientSettingsSection)config.SectionGroups["userSettings"].Sections[0];

}

catch { }

NumberFormatInfo nfi = new NumberFormatInfo { NumberDecimalSeparator = "." };

//read application level constants

if (elementsWeights == null || !decimal.TryParse(elementsWeights.Settings.Get("Mg").Value.ValueXml.InnerText, NumberStyles.Number, nfi, out awMg))

awMg = 24.305M;

if (elementsWeights == null || !decimal.TryParse(elementsWeights.Settings.Get("H").Value.ValueXml.InnerText, NumberStyles.Number, nfi, out awH)) awH = 1.008M;

if (elementsWeights == null || !decimal.TryParse(elementsWeights.Settings.Get("O").Value.ValueXml.InnerText, NumberStyles.Number, nfi, out awO)) awO = 15.999M;

if (elementsWeights == null || !decimal.TryParse(elementsWeights.Settings.Get("Ca").Value.ValueXml.InnerText,

NumberStyles.Number, nfi, out awCa)) awCa = 40.078M;

if (elementsWeights == null || !decimal.TryParse(elementsWeights.Settings.Get("Cl").Value.ValueXml.InnerText,

NumberStyles.Number, nfi, out awCl)) awCl = 35.45M;

\_water2MagnesiumRatioInCarnallite = 6 \* (2 \* awH + awO) / awMg;

if (elementsWeights == null || !decimal.TryParse(elementsWeights.Settings.Get("Na").Value.ValueXml.InnerText,

NumberStyles.Number, nfi, out awNa)) awNa = 23.99M;

if (elementsWeights == null || !decimal.TryParse(elementsWeights.Settings.Get("K").Value.ValueXml.InnerText,

NumberStyles.Number, nfi, out awK)) awK = 39.099M;

if (elementsWeights == null || !decimal.TryParse(elementsWeights.Settings.Get("C").Value.ValueXml.InnerText,

NumberStyles.Number, nfi, out awC)) awC = 12.011M;

if (elementsWeights == null || !decimal.TryParse(elementsWeights.Settings.Get("S").Value.ValueXml.InnerText,

NumberStyles.Number, nfi, out awS)) awS = 32.07M;

if (elementsWeights == null || !decimal.TryParse(elementsWeights.Settings.Get("Br").Value.ValueXml.InnerText,

NumberStyles.Number, nfi, out awBr)) awBr = 79.9M;

if (elementsWeights == null || !decimal.TryParse(elementsWeights.Settings.Get("B").Value.ValueXml.InnerText,

NumberStyles.Number, nfi, out awB)) awB = 10.81M;

if (userSettings == null || !decimal.TryParse(userSettings.Settings.Get("SumTolerance").Value.ValueXml.InnerText,

NumberStyles.Number, nfi, out sumTolerance)) sumTolerance = 0.005M;

//ions ratio in minerals constants

\_SO4\_2\_CaS04 = (awS + 4 \* awO) / (awS + 4 \* awO + awCa);

\_CaSO4\_2\_SO4 = (awS + 4 \* awO + awCa) / (awS + 4 \* awO);

\_CaCl2\_2\_Ca = (awCa + awCl \* 2) / awCa;

\_MgCl2\_2\_Mg = (awMg + awCl \* 2) / awMg;

\_KCl\_2\_K = (awK + awCl) / awK;

\_NaCl\_2\_Cl = (awNa + awCl) / awCl;

\_KBr\_2\_Br = (awK + awBr) / awBr;

\_Carnallite\_2\_Magnesium = (12 \* awH + 6 \* awO + awMg + 3 \* awCl + awK) / awMg;

\_Ca\_2\_HCO3 = awCa / (2 \* (awH + awC + 3 \* awO));

\_SO4\_2\_Ca = (awS + 4 \* awO) / awCa;

\_SO4\_2\_Mg = (awS + 4 \* awO) / awMg;

\_Na\_2\_SO4 = (2 \* awNa) / (awS + 4 \* awO);

//mass percentage to normality concentration constants

\_Eq\_CO3 = 1000 / (awC + 3 \* awO);

\_Eq\_HCO3 = 1000 / (awC + awH + 3 \* awO);

\_Eq\_Mg = 1000 / (awMg);

\_Eq\_Ca = 1000 / (awCa);

\_Eq\_SO4 = 1000 / (awS + 4 \* awO);

//brushes needed to signal quality of calcs

if (redBrush == null || greenBrush == null || blueBrush == null)

{

var converter = new BrushConverter();

greenBrush=(SolidColorBrush)converter.ConvertFromString(Colors.Green.ToString());

greenBrush.Freeze();

blueBrush = (SolidColorBrush)converter.ConvertFromString(Colors.Blue.ToString());

blueBrush.Freeze();

redBrush = (SolidColorBrush)converter.ConvertFromString(Colors.Red.ToString());

redBrush.Freeze();

}

}

}

public SaltAnalysisData(IDictionary<int, ILinearCalibration> lc) : this()

{ if (lc != null) lcDict = lc; }

public SaltAnalysisData(decimal CarnalliteThreshold) : this()

{ carnalliteThreshold = CarnalliteThreshold; }

public SaltAnalysisData(IDictionary<int, ILinearCalibration> lc,

decimal CarnalliteThreshold) : this(CarnalliteThreshold)

{ if (lc != null) lcDict = lc; }

public void CalcDryValues(SaltCalculationSchemes defScheme)

{

MgWet = (MagnesiumTitre \* MagnesiumTrilonTitre / MagnesiumAliquote

- CalciumTitre \* CalciumTrilonTitre / CalciumAliquote)

//0.0125 = 0.5 \* 0.05 \* 500 /1000

\* 0.0125M \* awMg / WetWeight;

switch (defScheme)

{

case SaltCalculationSchemes.Chloride:

HumidityContent = (MgWet >= carnalliteThreshold)? (HumidityCrucibleWetSampleWeight - ((HumidityCrucibleDry180SampleWeight.HasValue) ? HumidityCrucibleDry180SampleWeight.Value : HumidityCrucibleDry110SampleWeight)) / (HumidityCrucibleWetSampleWeight - HumidityCrucibleEmptyWeight): (HumidityCrucibleWetSampleWeight - HumidityCrucibleDry110SampleWeight) / (HumidityCrucibleWetSampleWeight - HumidityCrucibleEmptyWeight);

break;

default:

HumidityContent = (HumidityCrucibleWetSampleWeight - HumidityCrucibleDry110SampleWeight) / (HumidityCrucibleWetSampleWeight - HumidityCrucibleEmptyWeight);

break;

}

//Set corrected dry weight of the sample depending on the scheme

SampleCorrectedDryWeight = CalcCorrectedDryWeight(this, DefaultCalculationScheme);

MgDry = (MagnesiumTitre \* MagnesiumTrilonTitre / MagnesiumAliquote

- CalciumTitre \* CalciumTrilonTitre / CalciumAliquote)

\* 0.0125M \* awMg / SampleCorrectedDryWeight;

CaDry = CalciumTitre \* CalciumTrilonTitre \* 0.0125M

\* awCa / (SampleCorrectedDryWeight \* CalciumAliquote);

ClDry = ChlorumTitre \* awCl \* HgCoefficient / (SampleCorrectedDryWeight \* ChlorumAliquote \* 20);

BrDry = (BromumTitre - BromumBlank) \* BromumStandardTitre \* 0.5M

/ (SampleCorrectedDryWeight \* BromumAliquote);

ResiduumDry = (ResiduumCrucibleFullWeight - ResiduumCrucibleEmptyWeight) / SampleCorrectedDryWeight;

SulfatesDry = (SulfatesCrucibleFullWeight - SulfatesCrucibleEmptyWeight - SulfatesBlank) \*

205.75M / (SampleCorrectedDryWeight \* SulfatesAliquote);

CarbonatesDry = CarbonatesTitre / (1000 \* SampleCorrectedDryWeight);

HydrocarbonatesDry = HydrocarbonatesTitre / (1000 \* SampleCorrectedDryWeight);

}

public decimal CalcCorrectedDryWeight(ISaltAnalysisDryData dryData, SaltCalculationSchemes defScheme)

{

switch (defScheme)

{

case SaltCalculationSchemes.Chloride:

return (dryData.MgWet >= carnalliteThreshold)

? NewDryWeight(WetWeight, MgWet, dryData.HumidityContent, \_water2MagnesiumRatioInCarnallite)

: WetWeight \* (1 - dryData.HumidityContent);

default://All other schemes

return WetWeight \* (1 - dryData.HumidityContent);

}

}

public decimal CalcKaliumValue()

{

ILinearCalibration lc;

if (!lcDict.ContainsKey(KaliumCalibration))

{

using (var context = new ChemicalAnalysesEntities())

{

lc = context.LineaCalibrations.Find(KaliumCalibration);

(lc as LinearCalibration).GetLinearCoefficients();

lcDict.Add(KaliumCalibration, lc);

}

}

else { lc = lcDict[KaliumCalibration]; }

try

{

return lc.ValueToConcentration(KaliumValue, KaliumDiapason - 1) \* KaliumVolume

/ (2 \* SampleCorrectedDryWeight);

}

catch (Exception ex)

{

throw new DivideByZeroException("Скорректированный сухой вес не может быть равен нулю", ex);

}

}

public SaltCalculationSchemes CalcRecommendedScheme(ISaltAnalysisDryData dryData)

{

decimal Coeff1 = 0, Coeff2 = 0, Coeff3 = 0, Coeff4 = 0;

try

{

Coeff1 = (dryData.CarbonatesDry \* \_Eq\_CO3 + dryData.HydrocarbonatesDry \* \_Eq\_HCO3) /

(dryData.CaDry \* \_Eq\_Ca + MgDry \* \_Eq\_Mg);

}

catch { }

if (Coeff1 >= 1) return SaltCalculationSchemes.Carbonate;

try

{

Coeff2 = (dryData.CarbonatesDry \* \_Eq\_CO3 + dryData.HydrocarbonatesDry \* \_Eq\_HCO3

+ dryData.SulfatesDry \* \_Eq\_SO4) / (dryData.CaDry \* \_Eq\_Ca + dryData.MgDry \* \_Eq\_Mg);

}

catch { }

try { Coeff3 = dryData.SulfatesDry \* \_Eq\_SO4 / (dryData.CaDry \* \_Eq\_Ca); }

catch { }

try { Coeff4 = (dryData.CarbonatesDry \* \_Eq\_CO3 + dryData.HydrocarbonatesDry \* \_Eq\_HCO3)

/ (dryData.CaDry \* \_Eq\_Ca); }

catch { }

if (Coeff2 >= 1)

{

if (Coeff4 < 1) return SaltCalculationSchemes.SulfateSodiumI;

return SaltCalculationSchemes.SulfateSodiumII;

}

else

{

if (Coeff3 >= 1)

{

if (Coeff4 < 1) return SaltCalculationSchemes.SulfateMagnesiumI;

return SaltCalculationSchemes.SulfateMagnesiumII;

}

return SaltCalculationSchemes.Chloride;

}

}

public ISaltAnalysisCalcResults CalcSchemeResults(ISaltAnalysisDryData dryData,

SaltCalculationSchemes defSchema)

{

decimal \_IonSum = dryData.KDry + dryData.CaDry + dryData.MgDry

+ dryData.SulfatesDry + dryData.ClDry + dryData.BrDry + dryData.ResiduumDry;

switch (defSchema)

{

case SaltCalculationSchemes.Chloride:

{

CaSO4 = dryData.SulfatesDry \* \_CaSO4\_2\_SO4;

decimal \_Ca\_bound2\_SO4 = CaSO4 - dryData.SulfatesDry;

decimal \_Ca\_bound2\_Cl = dryData.CaDry - \_Ca\_bound2\_SO4;

CaCl2 = \_Ca\_bound2\_Cl \* \_CaCl2\_2\_Ca;

decimal \_Cl\_bound2\_Ca = CaCl2 - \_Ca\_bound2\_Cl;

MgCl2AnyCase = dryData.MgDry \* \_MgCl2\_2\_Mg;

MgCl2 = (dryData.MgWet >= carnalliteThreshold) ? dryData.MgDry \*

\_MgCl2\_2\_Mg : (decimal?)null;

decimal \_Cl\_bound\_2\_Mg = (MgCl2.HasValue) ? MgCl2.Value : 0 - dryData.MgDry;

KBr = dryData.BrDry \* \_KBr\_2\_Br;

decimal \_K\_bound2\_Br = KBr - dryData.BrDry;

decimal \_K\_bound\_2\_Cl = dryData.KDry - \_K\_bound2\_Br;

KCl = \_K\_bound\_2\_Cl \* \_KCl\_2\_K;

decimal \_Cl\_bound2\_K = KCl - \_K\_bound\_2\_Cl;

decimal \_summaryCl = \_Cl\_bound2\_Ca + \_Cl\_bound\_2\_Mg + \_Cl\_bound2\_K;

decimal \_CL\_bound2\_Na = dryData.ClDry - \_summaryCl;

Na = \_CL\_bound2\_Na \* awNa / awCl;

NaCl = Na + \_CL\_bound2\_Na;

CrystWater = (dryData.MgWet >= carnalliteThreshold)? dryData.MgDry \* \_water2MagnesiumRatioInCarnallite : (decimal?)null;

HygroWater = HumidityContent - CrystWater;

Carnallite = dryData.MgDry \* \_Carnallite\_2\_Magnesium;

HygroWaterAnyCase = (dryData.MgWet >= carnalliteThreshold) ? dryData.HumidityContent - dryData.MgDry \* \_water2MagnesiumRatioInCarnallite: dryData.HumidityContent;

IsCalculated = true;

\_saltSum = NaCl + KCl + CaSO4 + CaCl2 + (MgCl2 ?? 0) + KBr + ResiduumDry + Carnallite;

}

break;

case SaltCalculationSchemes.SulfateSodiumI:

{

decimal \_Ca\_bound2\_HCO3 = dryData.HydrocarbonatesDry \* \_Ca\_2\_HCO3; //1

CaHCO3\_2 = dryData.HydrocarbonatesDry + \_Ca\_bound2\_HCO3; //2

decimal \_Ca\_bound2\_SO4 = dryData.CaDry - \_Ca\_bound2\_HCO3; //3

decimal \_SO4\_bound2\_Ca = \_Ca\_bound2\_SO4 \* \_SO4\_2\_Ca; //4

CaSO4 = \_Ca\_bound2\_SO4 + \_SO4\_bound2\_Ca; //5

decimal \_SO4\_bound2\_Mg = dryData.MgDry \* \_SO4\_2\_Mg; //6

MgSO4 = \_SO4\_bound2\_Mg + dryData.MgDry; //7

decimal \_SO4\_bound2\_Na = dryData.SulfatesDry - \_SO4\_bound2\_Ca - \_SO4\_bound2\_Mg; //8

decimal \_Na\_bound2\_SO4 = \_SO4\_bound2\_Na \* \_Na\_2\_SO4; //9

Na2SO4 = \_SO4\_bound2\_Na + \_Na\_bound2\_SO4; //10

decimal \_Cl\_bound2\_K = dryData.KDry \* awCl / awK; //11

KCl = dryData.KDry + \_Cl\_bound2\_K; //12

decimal \_Cl\_bound2\_Na = dryData.ClDry - \_Cl\_bound2\_K; //13

decimal \_Na\_bound2\_Cl = \_Cl\_bound2\_Na \* awNa / awCl; //14

NaCl = \_Cl\_bound2\_Na + \_Na\_bound2\_Cl; //15

decimal \_Na\_bound2\_Br = dryData.BrDry \* awNa / awBr; //16

NaBr = dryData.BrDry + \_Na\_bound2\_Br; //17

Na = \_Na\_bound2\_SO4 + \_Na\_bound2\_Cl + \_Na\_bound2\_Br; //20

CrystWater = (HumidityCrucibleDry110SampleWeight - HumidityCrucibleDry180SampleWeight ?? 0 )/ (HumidityCrucibleWetSampleWeight - HumidityCrucibleEmptyWeight);

HygroWater = HygroWaterAnyCase = (HumidityCrucibleWetSampleWeight - HumidityCrucibleDry110SampleWeight) / (HumidityCrucibleWetSampleWeight - HumidityCrucibleEmptyWeight);

IsCalculated = true;

\_saltSum = NaCl + KCl + CaSO4 + NaBr + MgSO4 + Na2SO4 + ResiduumDry + CrystWater ?? 0;

}

break;

case SaltCalculationSchemes.SulfateMagnesiumI:

{

decimal \_Ca\_bound2\_HCO3 = dryData.HydrocarbonatesDry \* \_Ca\_2\_HCO3; //1

CaHCO3\_2 = dryData.HydrocarbonatesDry + \_Ca\_bound2\_HCO3; //2

decimal \_Ca\_bound2\_SO4 = dryData.CaDry - \_Ca\_bound2\_HCO3; //3

decimal \_SO4\_bound2\_Ca = \_Ca\_bound2\_SO4 \* \_SO4\_2\_Ca; //4

CaSO4 = \_Ca\_bound2\_SO4 + \_SO4\_bound2\_Ca; //5

decimal \_SO4\_bound2\_Mg = dryData.SulfatesDry - \_SO4\_bound2\_Ca; //6

decimal \_Mg\_bound2\_SO4 = \_SO4\_bound2\_Mg / \_SO4\_2\_Mg; //7

MgSO4 = \_SO4\_bound2\_Mg + \_Mg\_bound2\_SO4; //8

decimal \_Mg\_bound2\_Cl = dryData.MgDry - \_Mg\_bound2\_SO4; //9

decimal \_Cl\_bound2\_Mg = \_Mg\_bound2\_Cl \* 2 \* awCl / awMg; //10

MgCl2 = \_Mg\_bound2\_Cl + \_Cl\_bound2\_Mg; //11

decimal \_Cl\_bound2\_K = dryData.KDry \* awCl / awK; //12

KCl = dryData.KDry + \_Cl\_bound2\_K; //13

decimal \_Cl\_bound2\_Na = dryData.ClDry - \_Cl\_bound2\_K - \_Cl\_bound2\_Mg; //14

decimal \_Na\_bound2\_Cl = \_Cl\_bound2\_Na \* awNa / awCl; //15

NaCl = \_Cl\_bound2\_Na + \_Na\_bound2\_Cl; //16

decimal \_Na\_bound2\_Br = dryData.BrDry \* awNa / awBr; //17

NaBr = dryData.BrDry + \_Na\_bound2\_Br; //18

Na = \_Na\_bound2\_Cl + \_Na\_bound2\_Br; //21

CrystWater = (HumidityCrucibleDry110SampleWeight - HumidityCrucibleDry180SampleWeight ?? 0) / (HumidityCrucibleWetSampleWeight - HumidityCrucibleEmptyWeight);

HygroWater = HygroWaterAnyCase = (HumidityCrucibleWetSampleWeight - HumidityCrucibleDry110SampleWeight) / (HumidityCrucibleWetSampleWeight - HumidityCrucibleEmptyWeight);

IsCalculated = true;

\_saltSum = NaCl + KCl + CaSO4 + MgSO4 + (MgCl2 ?? 0) + NaBr + ResiduumDry + CrystWater ?? 0;

}

break;

default: //All others - not yet implemented

break;

}

\_ionSum = \_IonSum + Na + (CrystWater ?? 0);

return this;

}

public ISaltAnalysisCalcResults CalcSchemeTolerance (ISaltAnalysisCalcResults res)

{

ISaltAnalysisCalcResults \_res;

\_res = res;

if (Math.Abs(1 - res.IonSum) < sumTolerance) \_res.IonSumColor = greenBrush;

else

{

if ((1 - res.IonSum) > 0) \_res.IonSumColor = redBrush;

else \_res.IonSumColor = blueBrush;

}

if (Math.Abs(1 - res.SaltSum) < sumTolerance) \_res.SaltSumColor = greenBrush;

else

{

if ((1 - res.SaltSum) > 0) \_res.SaltSumColor = redBrush;

else \_res.SaltSumColor = blueBrush;

}

return \_res;

}

private decimal NewDryWeight(decimal SampleWet, decimal MgWet,

decimal Humidity180, decimal WaterInCarnallite)

{

decimal WCryst2, WHygr2, SampleDry2, MgDry1, MgDry2;

MgDry1 = MgWet;

WCryst2 = Math.Round(MgDry1 \* WaterInCarnallite, 5);

WHygr2 = Humidity180 - WCryst2;

SampleDry2 = SampleWet \* (1 - WHygr2);

MgDry2 = MgWet \* SampleWet / SampleDry2;

if (Math.Abs(MgDry2 - MgDry1) > 0.0001M)

return NewDryWeight(SampleDry2, MgDry2, Humidity180, WaterInCarnallite);

else return SampleDry2;

}

}

[Flags()]

public enum SaltCalculationSchemes

{

[Description("Карбонатная")]

Carbonate = 1,

[Description("Сульфатно-натриевая (I тип)")]

[SchemeRealized]

SulfateSodiumI = 2,

[Description("Сульфатно-натриевая (II тип)")]

SulfateSodiumII = 4,

[Description("Сульфатно-магниевая (I тип)")]

[SchemeRealized]

SulfateMagnesiumI = 8,

[Description("Сульфатно-магниевая (II тип)")]

SulfateMagnesiumII = 16,

[Description("Хлоридная")]

[SchemeRealized]

Chloride = 32

};

}

A.27 Сущность «Калибровка», а также классы позволяющие вызывать хранимые процедуры, использующие табличные параметры

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Linq;

using System.Runtime.CompilerServices;

using System.ComponentModel.DataAnnotations.Schema;

using System.ComponentModel;

using SA\_EF.Interfaces;

using System.Data;

using EntityFrameworkExtras.EF6;

namespace SA\_EF

{

public partial class LinearCalibration : INotifyPropertyChanged, ILinearCalibration

{

private DateTime \_date = DateTime.Today;

public DateTime CalibrationDate

{

get { return \_date; }

set

{

if (value <= DateTime.Today)

{

\_date = value;

OnPropertyChanged("CalibrationDate");

}

else throw new ArgumentOutOfRangeException("CalibrationDate",

"Дата калибровки не может лежать в будущем!");

}

}

private string \_calibrationType = "Kalium";

public string CalibrationType

{

get { return \_calibrationType; }

set

{

\_calibrationType = value;

OnPropertyChanged("CalibrationType");

}

}

private string \_description = "Введите описание";

public string Description

{

get { return \_description; }

set

{

if (value == null || value.Trim() == "")

throw new ArgumentNullException("Description", "Введите описание!");

\_description = value;

OnPropertyChanged("Description");

}

}

private decimal[] \_intercept;

[NotMapped]

public decimal[] Intercept

{

get { return \_intercept; }

set

{

\_intercept = value;

OnPropertyChanged("Intercept");

}

}

private decimal[] \_slope;

[NotMapped]

public decimal[] Slope

{

get { return \_slope; }

set

{

\_slope = value;

OnPropertyChanged("Slope");

}

}

private decimal[] \_rSquared;

[NotMapped]

public decimal[] RSquared

{

get { return \_rSquared; }

private set //is calculated and set internally

{

\_rSquared = value;

OnPropertyChanged("RSquared");

}

}

private ObservableCollection<DataPoint>[] \_linearCalibrationData;

[NotMapped]

public ObservableCollection<DataPoint>[] LinearCalibrationData

{

get { return \_linearCalibrationData; }

set

{

\_linearCalibrationData = value;

OnPropertyChanged("LinearCalibrationData");

}

}

public event PropertyChangedEventHandler PropertyChanged;

public void OnPropertyChanged([CallerMemberName]string prop = "")

{ PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(prop)); }

public override string ToString()

{

return CalibrationDate.ToShortDateString().ToString() + " " + Description;

}

public void GetLinearCoefficients()

{

if (LinearCalibrationData == null || LinearCalibrationData.Any(p => p == null)) return;

else {

for (int i =0; i <= LinearCalibrationData.Rank; i++)

{

LinearCalibrationData[i] =

new ObservableCollection<DataPoint>(CalibrationData.Where(p => p.Diapason == i + 1));

}

}

for (int i = 0; i <= LinearCalibrationData.Rank; i++)

{

int Count = LinearCalibrationData[i].Count;

decimal \_sumConcentration = 0;

decimal \_sumValues = 0;

decimal \_sumSquares = 0;

decimal \_sumProducts = 0;

decimal d = 0;

decimal valueMean = LinearCalibrationData[i].Average(x => x.Value);

foreach (DataPoint t in LinearCalibrationData[i])

{

\_sumConcentration += t.Concentration;

\_sumValues += t.Value;

\_sumSquares += t.Concentration \* t.Concentration;

\_sumProducts += t.Concentration \* t.Value;

d += (decimal)Math.Pow((double)(t.Value - valueMean), 2.0);

}

decimal delta = \_sumSquares \* Count - \_sumConcentration \* \_sumConcentration;

if (delta == 0)

{

Slope[i] = decimal.MaxValue;

Intercept[i] = decimal.MaxValue;

RSquared[i] = 0;

}

else

{

Slope[i] = (\_sumProducts \* Count - \_sumConcentration \* \_sumValues) / delta;

Intercept[i] = (\_sumValues \* \_sumSquares - \_sumProducts \* \_sumConcentration) / delta;

if (d == 0) RSquared[i] = 0;

else RSquared[i] = 1 - LinearCalibrationData[i].Sum(p => (decimal)Math.Pow((double)(p.Value -

(Slope[i] \* p.Concentration + Intercept[i])), 2.0)) / d;

}

}

}

public decimal ValueToConcentration(decimal val, int diap)

{

if (diap < 0 || diap > 1) throw new ArgumentOutOfRangeException("Diapason", "Недопустимый номер диапазона");

if (val <= 0) throw new ArgumentOutOfRangeException("Value", "Неверное значение показателя");

if (val < LinearCalibrationData[diap].Min(p => p.Value) || val > LinearCalibrationData[diap].Max(p => p.Value))

{//calculate by coefficients

try { return (val - Intercept[diap]) / Slope[diap]; }

catch (Exception ex)

{

throw new ArgumentOutOfRangeException("Нулевое значение углового коэффициента", ex);

}

}

else

{//calculate by the way of interpolation between two dots

try

{

int i = LinearCalibrationData[diap].IndexOf(LinearCalibrationData[diap]

.Where(p => p.Value > val).First());

return (val - LinearCalibrationData[diap][i - 1].Value)

\* (LinearCalibrationData[diap][i].Concentration

- LinearCalibrationData[diap][i - 1].Concentration)

/ (LinearCalibrationData[diap][i].Value - LinearCalibrationData[diap][i - 1].Value)

+ LinearCalibrationData[diap][i - 1].Concentration;

}

catch (Exception ex)

{

throw new ArgumentOutOfRangeException(@"Две последующие точки в калибровочных данных имеют одинаковые значения показателя", ex);

}

}

}

public bool ContainsEqualDataPoints(int diap)

{

if (diap < 0 || diap > 1) return true; // contains

if (LinearCalibrationData != null && LinearCalibrationData[diap] != null)

{

try

{

(LinearCalibrationData[diap].ToList()).ToDictionary(p => p.Concentration, p => p);

(LinearCalibrationData[diap].ToList()).ToDictionary(p => p.Value, p => p);

return false;

}

catch

{ return true; //exception thrown by dictionary - diapason contains equal

}

}

else return true;

}

}

public enum ChemicalElemetCalibration

{

[Description("Калий")]

Kalium,

[Description("Натрий")]

Natrium

};

[StoredProcedure("UpdateCalibrationData")]

public class UpdateLCWithSP

{

[StoredProcedureParameter(SqlDbType.Udt)]

public List<LCData> tmp { get; set; }

}

[UserDefinedTableType("TempCalibrationData")]

public class LCData

{

[UserDefinedTableTypeColumn(1, "IDCalibration")]

public int IDCalibration { get; set; }

[UserDefinedTableTypeColumn(2, "IDCalibrationData")]

public int IDCalibrationData { get; set; }

[UserDefinedTableTypeColumn(3, "Diapason")]

public int Diapason { get; set; }

[UserDefinedTableTypeColumn(4, "Concentration")]

public decimal Concentration { get; set; }

[UserDefinedTableTypeColumn(5, "Value")]

public decimal Value { get; set; }

}

[StoredProcedure("DeleteCalibrationByID")]

public class DeleteCalibrationByID

{

[StoredProcedureParameter(SqlDbType.Int)]

public int Calibration\_ID { get; set; }

}

}

A.28 Класс, хранящий значения толеранса параметров, используемых для сравнения схем, а также вспомогательный класс, хранящий пару параметр-значение

using System;

using System.Text;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Runtime.CompilerServices;

using System.ComponentModel;

namespace SA\_EF

{

[Serializable]

public class SchemeResultsTolerance : INotifyPropertyChanged

{

private bool \_isUniversalTolerance = false;

public bool IsUniversalTolerance

{

get { return \_isUniversalTolerance; }

set

{

\_isUniversalTolerance = value;

OnPropertyChanged(nameof(IsUniversalTolerance));

}

}

private decimal? \_universalTolerance = 0.005M;

public decimal? UniversalTolerance

{

get { return \_universalTolerance; }

set

{

if (value != null && value.HasValue)

{

if (value > 0.1M || value <= 0) throw new ArgumentOutOfRangeException(nameof(UniversalTolerance), "Значение толеранса должно быть больше 0 и меньше 10%");

\_universalTolerance = value;

OnPropertyChanged(nameof(UniversalTolerance));

}

}

}

private ObservableCollection<ParameterValuePair> \_schemeTolerances;

public ObservableCollection<ParameterValuePair> SchemeTolerances

{

get { return \_schemeTolerances; }

set

{

if (value != null)

{

\_schemeTolerances = value;

OnPropertyChanged(nameof(SchemeTolerances));

}

}

}

public SchemeResultsTolerance(IEnumerable<ParameterValuePair> collection)

{

if (collection != null) SchemeTolerances =

new ObservableCollection<ParameterValuePair>(collection);

else SchemeTolerances = new ObservableCollection<ParameterValuePair>();

}

public SchemeResultsTolerance() { }

public SchemeResultsTolerance(string serializedString)

{

if (serializedString != null && !serializedString.Equals(string.Empty))

{

string[] strs = serializedString.Split(new[] { "\r\n", "\r", "\n" }, StringSplitOptions.None);

if (strs.Length == 3)

{

bool isU;

if (bool.TryParse(strs[0], out isU)) IsUniversalTolerance = isU;

decimal unT;

if (decimal.TryParse(strs[1], out unT)) UniversalTolerance = unT;

string[] vals = strs[2].Split(';');

if (vals.Length % 2 == 0)

{

decimal d;

SchemeTolerances = new ObservableCollection<ParameterValuePair>();

for (int i = 0; i < vals.Length;)

{

if (decimal.TryParse(vals[i + 1], out d)) SchemeTolerances.Add(new ParameterValuePair() { Item1=vals[i], Item2=d });

i += 2;

}

}

}

}

}

public event PropertyChangedEventHandler PropertyChanged;

public void OnPropertyChanged([CallerMemberName]string prop = "")

{ PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(prop)); }

public override string ToString()

{

StringBuilder stringBuilder = new StringBuilder();

string delimiter = string.Empty;

stringBuilder.Append(IsUniversalTolerance + Environment.NewLine + UniversalTolerance + Environment.NewLine);

foreach (ParameterValuePair kvp in SchemeTolerances)

{

stringBuilder.Append(delimiter + kvp.Item1 + ";" + kvp.Item2.ToString());

delimiter = ";";

}

return stringBuilder.ToString();

}

}

public class ParameterValuePair: INotifyPropertyChanged

{

private string \_item1;

public string Item1

{

get { return \_item1;}

set

{

\_item1 = value;

OnPropertyChanged(nameof(Item1));

}

}

private decimal? \_item2 = 0;

public decimal? Item2

{

get { return \_item2; }

set

{

if (value.HasValue && (value < 0.001M || value > 0.1M))

throw new ArgumentOutOfRangeException(nameof(Item2),"Значение должно находиться в пределах 0.1%—10%");

\_item2 = value;

OnPropertyChanged(nameof(Item2));

}

}

public event PropertyChangedEventHandler PropertyChanged;

public void OnPropertyChanged([CallerMemberName]string prop = "")

{ PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(prop)); }

}

}

A.29 Интерфейсы, используемые при работе с анализами и калибровками

using System.Windows.Media;

namespace SA\_EF.Interfaces

{

public interface ISaltAnalysisCalculation

{

decimal CalcKaliumValue();

SaltCalculationSchemes CalcRecommendedScheme(ISaltAnalysisDryData dryData);

decimal CalcCorrectedDryWeight(ISaltAnalysisDryData dryData, SaltCalculationSchemes defScheme);

ISaltAnalysisCalcResults CalcSchemeResults(ISaltAnalysisDryData dryData, SaltCalculationSchemes defSchema);

}

public interface ISaltAnalysisDryData

{

decimal MgWet { get; set; }

[SchemesToCheckAttibute(SaltCalculationSchemes.Chloride | SaltCalculationSchemes.SulfateMagnesiumI

| SaltCalculationSchemes.SulfateSodiumI)]

[CustomDescription("Магний")]

decimal MgDry { get; set; }

[SchemesToCheckAttibute(SaltCalculationSchemes.Chloride | SaltCalculationSchemes.SulfateMagnesiumI

| SaltCalculationSchemes.SulfateSodiumI)]

[CustomDescription("Кальций")]

decimal CaDry { get; set; }

[SchemesToCheckAttibute(SaltCalculationSchemes.Chloride | SaltCalculationSchemes.SulfateMagnesiumI

| SaltCalculationSchemes.SulfateSodiumI)]

[CustomDescription("Хлор")]

decimal ClDry { get; set; }

[SchemesToCheckAttibute(SaltCalculationSchemes.Chloride | SaltCalculationSchemes.SulfateMagnesiumI

| SaltCalculationSchemes.SulfateSodiumI)]

[CustomDescription("Бром")]

decimal BrDry { get; set; }

[SchemesToCheckAttibute(SaltCalculationSchemes.Chloride | SaltCalculationSchemes.SulfateMagnesiumI

| SaltCalculationSchemes.SulfateSodiumI)]

[CustomDescription("Нерастворимый осадок")]

decimal ResiduumDry { get; set; }

[SchemesToCheckAttibute(SaltCalculationSchemes.Chloride | SaltCalculationSchemes.SulfateMagnesiumI

| SaltCalculationSchemes.SulfateSodiumI)]

[CustomDescription("Сульфаты")]

decimal SulfatesDry { get; set; }

decimal SampleCorrectedDryWeight { get; set; }

decimal CarbonatesDry { get; set; }

decimal HydrocarbonatesDry { get; set; }

decimal HumidityContent { get; set; }

[SchemesToCheckAttibute(SaltCalculationSchemes.Chloride | SaltCalculationSchemes.SulfateMagnesiumI

| SaltCalculationSchemes.SulfateSodiumI)]

[CustomDescription("Калий")]

decimal KDry { get; set; }

}

public interface ISaltAnalysisCalcResults

{

[SchemesToCheckAttibute(SaltCalculationSchemes.Chloride | SaltCalculationSchemes.SulfateMagnesiumI

| SaltCalculationSchemes.SulfateSodiumI)]

[CustomDescription("Сульфат кальция")]

decimal CaSO4 { get; set; }

[SchemesToCheckAttibute(SaltCalculationSchemes.Chloride | SaltCalculationSchemes.SulfateMagnesiumI

| SaltCalculationSchemes.SulfateSodiumI)]

[CustomDescription("Натрий")]

decimal Na { get; set; }

[SchemesToCheckAttibute(SaltCalculationSchemes.Chloride | SaltCalculationSchemes.SulfateMagnesiumI

| SaltCalculationSchemes.SulfateSodiumI)]

[CustomDescription("Кристаллогидратная вода")]

decimal? CrystWater { get; set; }

decimal Carnallite { get; set; }

[SchemesToCheckAttibute(SaltCalculationSchemes.Chloride | SaltCalculationSchemes.SulfateMagnesiumI

| SaltCalculationSchemes.SulfateSodiumI)]

[CustomDescription("Хлорид натрия")]

decimal NaCl { get; set; }

[SchemesToCheckAttibute(SaltCalculationSchemes.Chloride | SaltCalculationSchemes.SulfateMagnesiumI

| SaltCalculationSchemes.SulfateSodiumI)]

[CustomDescription("Хлорид калия")]

decimal KCl { get; set; }

[SchemesToCheckAttibute]

[CustomDescription("Хлорид кальция")]

decimal CaCl2 { get; set; }

[SchemesToCheckAttibute(SaltCalculationSchemes.Chloride | SaltCalculationSchemes.SulfateMagnesiumI)]

[CustomDescription("Хлорид магния")]

decimal? MgCl2 { get; set; }

decimal MgCl2AnyCase { get; set; }

[SchemesToCheckAttibute]

[CustomDescription("Бромид калия")]

decimal KBr { get; set; }

[SchemesToCheckAttibute(SaltCalculationSchemes.Chloride | SaltCalculationSchemes.SulfateMagnesiumI

| SaltCalculationSchemes.SulfateSodiumI)]

[CustomDescription("Гигроскопическая влага")]

decimal? HygroWater { get; set; }

decimal HygroWaterAnyCase { get; set; }

decimal CaHCO3\_2 { get; set; }

[SchemesToCheckAttibute(SaltCalculationSchemes.SulfateMagnesiumI | SaltCalculationSchemes.SulfateSodiumI)]

[CustomDescription("Сульфат магния")]

decimal MgSO4 { get; set; }

[SchemesToCheckAttibute(SaltCalculationSchemes.SulfateSodiumI)]

[CustomDescription("Сульфат натрия")]

decimal Na2SO4 { get; set; }

[SchemesToCheckAttibute(SaltCalculationSchemes.SulfateMagnesiumI | SaltCalculationSchemes.SulfateSodiumI)]

decimal NaBr { get; set; }

string LabNumber { get; set; }

string AnalysisDescription { get; set; }

SaltCalculationSchemes RecommendedCalculationScheme { get; set; }

SaltCalculationSchemes DefaultCalculationScheme { get; set; }

bool IsCalculated { get; set; }

[SchemesToCheckAttibute(SaltCalculationSchemes.Chloride | SaltCalculationSchemes.SulfateMagnesiumI

| SaltCalculationSchemes.SulfateSodiumI)]

[CustomDescription("Сумма ионных форм")]

decimal IonSum { get; }

[SchemesToCheckAttibute(SaltCalculationSchemes.Chloride | SaltCalculationSchemes.SulfateMagnesiumI

| SaltCalculationSchemes.SulfateSodiumI)]

[CustomDescription("Сумма солевых форм")]

decimal SaltSum { get; }

SolidColorBrush IonSumColor { get; set; }

SolidColorBrush SaltSumColor { get; set; }

}

}

namespace SA\_EF.Interfaces

{

public interface ILinearCalibration

{

int CalibrationID { get; set; }

decimal[] Slope { get; set; }

decimal[] Intercept { get; set; }

decimal ValueToConcentration(decimal val, int diap);

}

}

A.30 Атрибуты, используемые указания свойств расчетных схем

using System;

namespace SA\_EF

{

[AttributeUsage(AttributeTargets.Property, AllowMultiple = false, Inherited = true)]

public sealed class SchemesToCheckAttibute: Attribute

{

public SaltCalculationSchemes Scheme;

public SchemesToCheckAttibute(SaltCalculationSchemes sch) { Scheme = sch; }

public SchemesToCheckAttibute():this(SaltCalculationSchemes.Chloride){}

}

[AttributeUsage(AttributeTargets.Field, AllowMultiple = false, Inherited = false)]

public sealed class SchemeRealizedAttribute : Attribute

{

public SchemeRealizedAttribute() { }

}

[AttributeUsage(AttributeTargets.Property, AllowMultiple = false, Inherited = true)]

public sealed class CustomDescriptionAttribute : Attribute

{

public string Description;

public CustomDescriptionAttribute(string description) { Description = description; }

public override string ToString() { return Description; }

}

}

A.31 Сущность «Точка калибровки»

using System;

using System.Runtime.CompilerServices;

using System.ComponentModel;

namespace SA\_EF

{

public partial class DataPoint : INotifyPropertyChanged, IEquatable<DataPoint>

{

private decimal conc = 0.001M;

public decimal Concentration

{

get { return conc; }

set

{

if (value <= 0) throw new ArgumentOutOfRangeException("Concentration",

"Концентрация должна быть положительным числом!");

else { conc = value; OnPropertyChanged("Concentration"); }

}

}

private decimal val = 1;

public decimal Value

{

get { return val; }

set

{

if (value <= 0) throw new ArgumentOutOfRangeException("Value",

"Показание прибора должно быть положительным числом!");

else { val = value; OnPropertyChanged("value"); }

}

}

public event PropertyChangedEventHandler PropertyChanged;

public void OnPropertyChanged([CallerMemberName]string prop = "")

{ PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(prop)); }

public override bool Equals(object obj)

{ return this.Equals(obj as DataPoint); }

public bool Equals(DataPoint dp)

{

if (dp == null) return false;

return Concentration.Equals(dp.Concentration) && Value.Equals(dp.Value);

}

public override int GetHashCode()

{ return (int)(Value \* 26440451) + (int)(Concentration \* 334216273); }

public static bool operator ==(DataPoint dp1, DataPoint dp2)

{

if (ReferenceEquals(dp1, dp2)) return true;

if (ReferenceEquals(dp1, null) || ReferenceEquals(dp2, null)) return false;

return dp1.Equals(dp2);

}

public static bool operator !=(DataPoint dp1, DataPoint dp2)

{ return !(dp1 == dp2); }

}

}

A.32 Класс, используемый для декодирования строки поключения using System;

using System.Text;

using System.Security.Cryptography;

using System.Security.Cryptography.X509Certificates;

namespace SA\_EF

{

public class X509EncDec

{

private static X509Certificate2 certificate;

public X509EncDec(string certificateName)

{ if (certificate == null) certificate = getCertificate(certificateName); }

private X509Certificate2 getCertificate(string certificateName)

{

X509Store my = new X509Store(StoreName.My, StoreLocation.LocalMachine);

my.Open(OpenFlags.ReadOnly);

X509Certificate2Collection collection = my.Certificates.Find(X509FindType.FindBySubjectName, certificateName, false);

if (collection.Count == 1)

{ return collection[0]; }

else if (collection.Count > 1)

{

throw new Exception(string.Format("More than one certificate with name '{0}' found in store LocalMachine/My.", certificateName));

}

else

{

throw new Exception(string.Format("Certificate '{0}' not found in store LocalMachine/My.", certificateName));

}

}

public string EncryptRsa(string input)

{

string output = string.Empty;

RSACryptoServiceProvider csp = (RSACryptoServiceProvider)certificate.PublicKey.Key;

{

byte[] bytesData = Encoding.UTF8.GetBytes(input);

byte[] bytesEncrypted = csp.Encrypt(bytesData, false);

output = Convert.ToBase64String(bytesEncrypted);

}

return output;

}

public string DecryptRsa(string encrypted)

{

string text = string.Empty;

RSACryptoServiceProvider csp = (RSACryptoServiceProvider)certificate.PrivateKey;

{

byte[] bytesEncrypted = Convert.FromBase64String(encrypted);

byte[] bytesDecrypted = csp.Decrypt(bytesEncrypted, false);

text = Encoding.UTF8.GetString(bytesDecrypted);

}

return text;

}

}}

A.33 Пользовательский элемент управления, используемый для вывода на печать результатов расчета

using Microsoft.Office.Interop.Excel;

using SA\_EF;

using SA\_EF.Interfaces;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Runtime.CompilerServices;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

namespace PrintHelper

{

public partial class SchemesPrintingGrid : UserControl, INotifyPropertyChanged

{

public IEnumerable<ISaltAnalysisCalcResults> lSA { get; set; }

public SchemesPrintingGrid(IEnumerable<ISaltAnalysisCalcResults> lsa)

{

InitializeComponent();

lSA = lsa;

DataContext = this;

}

//Autonumbering

private void dgrdMain\_LoadingRow(object sender, DataGridRowEventArgs e)

{ e.Row.Header = (e.Row.GetIndex() + 1).ToString(); }

private bool \_showHygroscopicWaterForAll = true;

public bool ShowHygroscopicWaterForAll

{

get { return \_showHygroscopicWaterForAll; }

set

{

\_showHygroscopicWaterForAll = value;

OnPropertyChanged(nameof(ShowHygroscopicWaterForAll));

}

}

private bool \_useBKRespresentationVariant = true;

public bool UseBKRespresentationVariant

{

get { return \_useBKRespresentationVariant; }

set

{

\_useBKRespresentationVariant = value;

OnPropertyChanged(nameof(UseBKRespresentationVariant));

}

}

public event PropertyChangedEventHandler PropertyChanged;

public void OnPropertyChanged([CallerMemberName]string prop = "")

{ PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(prop)); }

public SaltCalculationSchemes ResultsType

{

get { return (SaltCalculationSchemes)GetValue(ResultsTypeProperty); }

set { SetValue(ResultsTypeProperty, value); }

}

public static readonly DependencyProperty ResultsTypeProperty =

DependencyProperty.Register("ResultsType",

typeof(SaltCalculationSchemes), typeof(SchemesPrintingGrid),

new FrameworkPropertyMetadata(SaltCalculationSchemes.Chloride));

public void ExportToExcel (ref Workbook workbook)

{

if (workbook != null)

{

try

{

Worksheet ws = workbook.Worksheets.Add();

ws.Name = Name;

int i = 1;

foreach (DataGridColumn clmn in dgrdMain.Columns)

{

if (clmn.Visibility == Visibility.Visible)

{

if (clmn.Header.GetType().Equals(typeof(TextBlock)))

ws.Cells[1, i] = ((TextBlock)clmn.Header).Text;

else if (clmn.Header.GetType().Equals(typeof(StackPanel)))

{

string s="";

foreach (TextBlock tb in ((StackPanel)clmn.Header).Children)

s += tb.Text+Environment.NewLine;

ws.Cells[1, i] = s;

}

Binding bnd = null;

if (clmn.GetType().Equals(typeof(DataGridTextColumn)))

bnd = (Binding)(((DataGridTextColumn)clmn).Binding);

int j = 2;

foreach (ISaltAnalysisCalcResults dr in dgrdMain.ItemsSource)

{

if (bnd != null)

ws.Cells[j++, i] = dr.GetType().GetProperty(bnd?.Path.Path.ToString()).GetValue(dr)?? "—";

}

i++;

}

}

}

catch { }

}

}

}

}

<UserControl x:Class="PrintHelper.SchemesPrintingGrid"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:l="clr-namespace:PrintHelper"

xmlns:saef="clr-namespace:SA\_EF;assembly=SA\_EF"

mc:Ignorable="d" d:DesignHeight="300" d:DesignWidth="1100"

Foreground="Transparent"

DataContext="{Binding}">

<UserControl.Resources>

<l:doubleToPercentageConverter x:Key="dbl2prcConv"/>

<l:DecimalsToSumConverter x:Key="dcml2SumConv"/>

<l:BooleanToVisibilityConverter x:Key="bln2VisibilityConv"/>

<l:BooleanToVisibilityNegativeConverter x:Key="bln2VisibilityNegConv"/>

<l:SchemeToVisibilityConverter x:Key="sch2VisibilityConv"/>

<l:SchemeToVisibilityMultipleConverter x:Key="sch2VisibilityMultConv"/>

</UserControl.Resources>

<DataGrid AutoGenerateColumns="False" BorderThickness="0"

x:Name="dgrdMain" CanUserAddRows="False" ItemsSource="{Binding Path=lSA}"

HeadersVisibility="All" GridLinesVisibility="All" LoadingRow="dgrdMain\_LoadingRow">

<DataGrid.Resources>

<Style BasedOn="{StaticResource {x:Type TextBlock}}" TargetType="{x:Type TextBlock}">

<Setter Property="FontSize" Value="18"/>

<Setter Property="FontWeight" Value="Bold"/>

<Setter Property="Margin" Value="2,0,2,0"/>

<Setter Property="Foreground" Value="Black"/>

</Style>

<Style TargetType="{x:Type DataGridColumnHeader}">

<Setter Property="HorizontalContentAlignment" Value="Center"/>

<Setter Property="VerticalContentAlignment" Value="Center"/>

<Setter Property="Background" Value="White"/>

<Setter Property="BorderThickness">

<Setter.Value>

<Thickness Bottom="1" Top="1" Left="1" Right="0.55"/>

</Setter.Value>

</Setter>

<Setter Property="BorderBrush" Value="Black"/>

</Style>

<Style x:Key="Centering" TargetType="{x:Type TextBlock}">

<Setter Property="HorizontalAlignment" Value="Center" />

</Style>

<Style TargetType="{x:Type DataGridCell}">

<Setter Property="TextBlock.TextAlignment" Value="Center"></Setter>

</Style>

<l:BindingProxy x:Key="FirstColumnCheckedProxy" Data="{Binding ShowHygroscopicWaterForAll}"/>

<l:BindingProxy x:Key="ResultsTypeProxy" Data="{Binding ResultsType}"/>

</DataGrid.Resources>

<DataGrid.Template>

<ControlTemplate TargetType="{x:Type DataGrid}">

<Border BorderBrush="{TemplateBinding BorderBrush}"

BorderThickness="{TemplateBinding BorderThickness}"

Background="{TemplateBinding Background}"

Padding="{TemplateBinding Padding}"

SnapsToDevicePixels="True">

<ScrollViewer x:Name="DG\_ScrollViewer"

Focusable="false">

<ScrollViewer.Template>

<ControlTemplate TargetType="{x:Type ScrollViewer}">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="Auto" />

<ColumnDefinition Width="\*" />

<ColumnDefinition Width="Auto" />

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="Auto" />

<RowDefinition Height="\*" />

<RowDefinition Height="Auto" />

</Grid.RowDefinitions>

<Button Command="{x:Static DataGrid.SelectAllCommand}" Focusable="false" Style="{DynamicResource {ComponentResourceKey ResourceId=DataGridSelectAllButtonStyle, TypeInTargetAssembly={x:Type DataGrid}}}" Visibility="{Binding HeadersVisibility, ConverterParameter={x:Static DataGridHeadersVisibility.All}, Converter={x:Static DataGrid.HeadersVisibilityConverter}, RelativeSource={RelativeSource AncestorType={x:Type DataGrid}}}" Width="{Binding RowHeaderActualWidth, RelativeSource={RelativeSource Mode=FindAncestor, AncestorType={x:Type DataGrid}}}" IsEnabled="True"/>

<DataGridColumnHeadersPresenter x:Name="PART\_ColumnHeadersPresenter" Grid.Column="1" Visibility="{Binding HeadersVisibility, ConverterParameter={x:Static DataGridHeadersVisibility.Column}, Converter={x:Static DataGrid.HeadersVisibilityConverter}, RelativeSource={RelativeSource AncestorType={x:Type DataGrid}}}" />

<ScrollContentPresenter x:Name="PART\_ScrollContentPresenter" CanContentScroll="{TemplateBinding CanContentScroll}" Grid.ColumnSpan="2" Grid.Row="1" />

<ScrollBar x:Name="PART\_VerticalScrollBar" Grid.Column="2" Maximum="{TemplateBinding ScrollableHeight}" Orientation="Vertical" Grid.Row="1" Visibility="{TemplateBinding ComputedVerticalScrollBarVisibility}" Value="{Binding VerticalOffset, Mode=OneWay, RelativeSource={RelativeSource TemplatedParent}}" ViewportSize="{TemplateBinding ViewportHeight}" />

<Grid Grid.Column="1"

Grid.Row="2">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="{Binding NonFrozenColumnsViewportHorizontalOffset, RelativeSource={RelativeSource AncestorType={x:Type DataGrid}}}" />

<ColumnDefinition Width="\*" />

</Grid.ColumnDefinitions>

<ScrollBar x:Name="PART\_HorizontalScrollBar" Grid.Column="1" Maximum="{TemplateBinding ScrollableWidth}" Orientation="Horizontal" Visibility="{TemplateBinding ComputedHorizontalScrollBarVisibility}" Value="{Binding HorizontalOffset, Mode=OneWay, RelativeSource={RelativeSource TemplatedParent}}" ViewportSize="{TemplateBinding ViewportWidth}" />

</Grid>

</Grid>

</ControlTemplate>

</ScrollViewer.Template>

<ItemsPresenter SnapsToDevicePixels="{TemplateBinding SnapsToDevicePixels}" />

</ScrollViewer>

</Border>

</ControlTemplate>

</DataGrid.Template>

<DataGrid.Columns>

<DataGridTextColumn Binding="{Binding Path=LabNumber, UpdateSourceTrigger=PropertyChanged}" IsReadOnly="True">

<DataGridTextColumn.Header>

<TextBlock Text="Лаб.№"/>

</DataGridTextColumn.Header>

</DataGridTextColumn>

<DataGridTextColumn IsReadOnly="True" MinWidth="150" MaxWidth="170" Binding="{Binding Path=AnalysisDescription, UpdateSourceTrigger=PropertyChanged, Mode=TwoWay, NotifyOnSourceUpdated=True, NotifyOnTargetUpdated=True, NotifyOnValidationError=True, ValidatesOnExceptions=True}">

<DataGridTextColumn.Header>

<TextBlock Text="Описание"/>

</DataGridTextColumn.Header>

</DataGridTextColumn>

<DataGridTextColumn Binding="{Binding Path=KDry, UpdateSourceTrigger=PropertyChanged, Converter={StaticResource dbl2prcConv}, StringFormat={}{0:0.00}}" IsReadOnly="True">

<DataGridTextColumn.Header>

<TextBlock Text="K⁺"/>

</DataGridTextColumn.Header>

</DataGridTextColumn>

<DataGridTextColumn Binding="{Binding Path=Na, UpdateSourceTrigger=PropertyChanged, Converter={StaticResource dbl2prcConv}, StringFormat={}{0:0.00}}" IsReadOnly="True">

<DataGridTextColumn.Header>

<TextBlock Text="Na⁺"/>

</DataGridTextColumn.Header>

</DataGridTextColumn>

<DataGridTextColumn Binding="{Binding Path=CaDry, UpdateSourceTrigger=PropertyChanged, Converter={StaticResource dbl2prcConv}, StringFormat={}{0:0.00}}" IsReadOnly="True">

<DataGridTextColumn.Header>

<TextBlock Text="Ca²⁺"/>

</DataGridTextColumn.Header>

</DataGridTextColumn>

<DataGridTextColumn Binding="{Binding Path=MgDry, UpdateSourceTrigger=PropertyChanged, Converter={StaticResource dbl2prcConv}, StringFormat={}{0:0.000}}" IsReadOnly="True">

<DataGridTextColumn.Header>

<TextBlock Text="Mg²⁺"/>

</DataGridTextColumn.Header>

</DataGridTextColumn>

<DataGridTextColumn Binding="{Binding Path=SulfatesDry, UpdateSourceTrigger=PropertyChanged, Converter={StaticResource dbl2prcConv}, StringFormat={}{0:0.00}}" IsReadOnly="True">

<DataGridTextColumn.Header>

<TextBlock Text="SO₄²¯"/>

</DataGridTextColumn.Header>

</DataGridTextColumn>

<DataGridTextColumn Binding="{Binding Path=ClDry, UpdateSourceTrigger=PropertyChanged, Converter={StaticResource dbl2prcConv}, StringFormat={}{0:0.00}}" IsReadOnly="True">

<DataGridTextColumn.Header>

<TextBlock Text="Cl¯"/>

</DataGridTextColumn.Header>

</DataGridTextColumn>

<DataGridTextColumn Binding="{Binding Path=BrDry, UpdateSourceTrigger=PropertyChanged, Converter={StaticResource dbl2prcConv}, StringFormat={}{0:0.000}}" IsReadOnly="True">

<DataGridTextColumn.Header>

<TextBlock Text="Br¯"/>

</DataGridTextColumn.Header>

</DataGridTextColumn>

<DataGridTextColumn Binding="{Binding Path=ResiduumDry, UpdateSourceTrigger=PropertyChanged, Converter={StaticResource dbl2prcConv}, StringFormat={}{0:0.00}}" IsReadOnly="True">

<DataGridTextColumn.Header>

<TextBlock Text="Н.О."/>

</DataGridTextColumn.Header>

</DataGridTextColumn>

<DataGridTextColumn Binding="{Binding Path=CrystWater, UpdateSourceTrigger=PropertyChanged, Converter={StaticResource dbl2prcConv}, StringFormat={}{0:0.00}, TargetNullValue=—}" IsReadOnly="True">

<DataGridTextColumn.Header>

<StackPanel Orientation="Vertical">

<TextBlock Text="H₂O" Foreground="Black"/>

<TextBlock Text="крист." FontWeight="Normal" FontSize="13" Foreground="Black"/>

</StackPanel>

</DataGridTextColumn.Header>

</DataGridTextColumn>

<DataGridTemplateColumn IsReadOnly="True">

<DataGridTemplateColumn.Header>

<TextBlock Text="∑"/>

</DataGridTemplateColumn.Header>

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<TextBlock>

<TextBlock.Text>

<MultiBinding Converter="{StaticResource dcml2SumConv}" StringFormat="{}{0:0.00}">

<Binding Path="KDry"/>

<Binding Path="Na"/>

<Binding Path="CaDry"/>

<Binding Path="MgDry"/>

<Binding Path="SulfatesDry"/>

<Binding Path="ClDry"/>

<Binding Path="BrDry"/>

<Binding Path="CrystWater"/>

<Binding Path="ResiduumDry"/>

</MultiBinding>

</TextBlock.Text>

</TextBlock>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

<DataGridTextColumn Binding="{Binding Path=NaCl, UpdateSourceTrigger=PropertyChanged, Converter={StaticResource dbl2prcConv}, StringFormat={}{0:0.00}}" IsReadOnly="True">

<DataGridTextColumn.Header>

<TextBlock Text="NaCl"/>

</DataGridTextColumn.Header>

</DataGridTextColumn>

<DataGridTextColumn Binding="{Binding Path=KCl, UpdateSourceTrigger=PropertyChanged, Converter={StaticResource dbl2prcConv}, StringFormat={}{0:0.00}}" IsReadOnly="True">

<DataGridTextColumn.Header>

<TextBlock Text="KCl"/>

</DataGridTextColumn.Header>

</DataGridTextColumn>

<DataGridTextColumn Binding="{Binding Path=CaSO4, UpdateSourceTrigger=PropertyChanged, Converter={StaticResource dbl2prcConv}, StringFormat={}{0:0.00}}" IsReadOnly="True">

<DataGridTextColumn.Header>

<TextBlock Text="CaSO₄"/>

</DataGridTextColumn.Header>

</DataGridTextColumn>

<DataGridTextColumn Binding="{Binding Path=MgSO4, UpdateSourceTrigger=PropertyChanged, Converter={StaticResource dbl2prcConv}, StringFormat={}{0:0.00}}" IsReadOnly="True">

<DataGridTextColumn.Visibility>

<MultiBinding Converter="{StaticResource sch2VisibilityMultConv}">

<Binding Source="{StaticResource ResultsTypeProxy}" Path="Data"/>

<Binding Source="{x:Static saef:SaltCalculationSchemes.SulfateSodiumI}"/>

<Binding Source="{x:Static saef:SaltCalculationSchemes.SulfateMagnesiumI}"/>

</MultiBinding>

</DataGridTextColumn.Visibility>

<DataGridTextColumn.Header>

<TextBlock Text="MgSO₄"/>

</DataGridTextColumn.Header>

</DataGridTextColumn>

<DataGridTextColumn Binding="{Binding Path=CaCl2, UpdateSourceTrigger=PropertyChanged, Converter={StaticResource dbl2prcConv}, StringFormat={}{0:0.00}}" IsReadOnly="True" Visibility="{Binding Path=Data, Converter={StaticResource sch2VisibilityConv}, Source={StaticResource ResultsTypeProxy}, ConverterParameter={x:Static saef:SaltCalculationSchemes.Chloride}}">

<DataGridTextColumn.Header>

<TextBlock Text="CaCl₂"/>

</DataGridTextColumn.Header>

</DataGridTextColumn>

<DataGridTextColumn Binding="{Binding Path=MgCl2, UpdateSourceTrigger=PropertyChanged, Converter={StaticResource dbl2prcConv}, StringFormat={}{0:0.00}, TargetNullValue=—}" IsReadOnly="True">

<DataGridTextColumn.Visibility>

<MultiBinding Converter="{StaticResource sch2VisibilityMultConv}">

<Binding Source="{StaticResource ResultsTypeProxy}" Path="Data"/>

<Binding Source="{x:Static saef:SaltCalculationSchemes.Chloride}"/>

<Binding Source="{x:Static saef:SaltCalculationSchemes.SulfateMagnesiumI}"/>

</MultiBinding>

</DataGridTextColumn.Visibility>

<DataGridTextColumn.Header>

<TextBlock Text="MgCl₂"/>

</DataGridTextColumn.Header>

</DataGridTextColumn>

<DataGridTextColumn Binding="{Binding Path=KBr, UpdateSourceTrigger=PropertyChanged, Converter={StaticResource dbl2prcConv}, StringFormat={}{0:0.000}, TargetNullValue=—}" IsReadOnly="True" Visibility="{Binding Path=Data, Converter={StaticResource sch2VisibilityConv}, Source={StaticResource ResultsTypeProxy}, ConverterParameter={x:Static saef:SaltCalculationSchemes.Chloride}}">

<DataGridTextColumn.Header>

<TextBlock Text="KBr"/>

</DataGridTextColumn.Header>

</DataGridTextColumn>

<DataGridTextColumn Binding="{Binding Path=NaBr, UpdateSourceTrigger=PropertyChanged, Converter={StaticResource dbl2prcConv}, StringFormat={}{0:0.00}}" IsReadOnly="True">

<DataGridTextColumn.Visibility>

<MultiBinding Converter="{StaticResource sch2VisibilityMultConv}">

<Binding Source="{StaticResource ResultsTypeProxy}" Path="Data"/>

<Binding Source="{x:Static saef:SaltCalculationSchemes.SulfateSodiumI}"/>

<Binding Source="{x:Static saef:SaltCalculationSchemes.SulfateMagnesiumI}"/>

</MultiBinding>

</DataGridTextColumn.Visibility>

<DataGridTextColumn.Header>

<TextBlock Text="NaBr"/>

</DataGridTextColumn.Header>

</DataGridTextColumn>

<DataGridTextColumn Binding="{Binding Path=HygroWater, UpdateSourceTrigger=PropertyChanged, Converter={StaticResource dbl2prcConv}, StringFormat={}{0:0.00}, TargetNullValue=—}" IsReadOnly="True" Visibility="{Binding Path=Data, Converter={StaticResource bln2VisibilityNegConv}, Source={StaticResource FirstColumnCheckedProxy}}">

<DataGridTextColumn.Header>

<StackPanel Orientation="Vertical">

<TextBlock Text="H₂O" Foreground="Black"/>

<TextBlock Text="гигр." FontWeight="Normal" FontSize="13" Foreground="Black"/>

</StackPanel>

</DataGridTextColumn.Header>

</DataGridTextColumn>

<DataGridTextColumn x:Name="clmWaterForAll" Binding="{Binding Path=HygroWaterAnyCase, UpdateSourceTrigger=PropertyChanged, Converter={StaticResource dbl2prcConv}, StringFormat={}{0:0.00}, TargetNullValue=—}" IsReadOnly="True" Visibility="{Binding Path=Data, Converter={StaticResource bln2VisibilityConv}, Source={StaticResource FirstColumnCheckedProxy}}">

<DataGridTextColumn.Header>

<StackPanel Orientation="Vertical">

<TextBlock Text="H₂O" Foreground="Black"/>

<TextBlock Text="гигр." FontWeight="Normal" FontSize="13" Foreground="Black"/>

</StackPanel>

</DataGridTextColumn.Header>

</DataGridTextColumn>

</DataGrid.Columns>

</DataGrid>

</UserControl>

1. Необходимо использовать пару логин-пароль, имеющую только полномочия на вход, но не имеющую полномочий на чтение/изменение данных [↑](#footnote-ref-1)
2. Здесь и далее в тестировании функций, относящихся к модулю работы с образцами, у тестируемых образцов предполагается наличие как минимум одного анализа [↑](#footnote-ref-2)