**Техническое задание**

**Описание**

Нужно сделать программу для работы с базой данных плюс пару других функций.

В программе 4 вкладки – эскизы находятся в середине документа.

Целевая система – Windows 7 SP1 на которой установлен .NET 4.5.1. Только эта версия. Либо .NET Core поновее если она работает на Windows 7 SP1.

База данных MS SQLServer 2017

Среда разработки Visual Studio 2017 (предпочтительно), ну или 2015. Язык C#, GUI программы должен быть написан на WPF, обязательно MVVM на основе какой-нибудь библиотеки.

Nuget пакеты которые нужно использовать обязательно – Dapper (ORM для SQLServer). Остальные по согласованию.

Ваша задача – заложить архитектуру приложения и реализовать ключевые на текущий момент потребности.

Часть задач уже реализована в виде отдельных небольших задач, в дальнейшем будет перенесена в это новое приложение.

Также Вы должны быть готовы к дальнейшим доработкам за дополнительную плату.

Хотя приложение и небольшое, вся архитектура приложения должна быть построена в лучших традициях объектно – ориентированного программирования. Приветствуется использование паттернов там где оно реально нужно. Грамотный и хороший код за который нестыдно. Который бы с гордостью опубликовали на Github. Если Вы говнокодите – проходим мимо.

В первую очередь приложение необходимо для решения своих задач. Во вторую – познакомиться с хорошим сторонним кодом, дополнительными библиотеками.

Некоторые пожелания/особенности еще дополнительно описаны в самом конце документа.

Приложение будет запускаться на сервере, к которому подключаются 5-10 пользователей по RDP.

Каждый пользователь запускает свой экземпляр программы.

И также приложение будет запускаться на виртуальных машинах, которые в свою очередь будут запускаться на этом же сервере. Виртуальных машин много, тысячи.

В каждой виртуальной машине программа запускается в единственном экземпляре.

Виртуальные машины постоянно создаются / удаляются и загружаются / ставятся на паузу.

**Информация по базе данных**

Каждая виртуальная машина имеет соответствующий номер, в базах данных этот номер обозначается полем Number, номера могут идти от 1000 до 99999.

Если упоминается слова виртуалка, номер виртуалки, машина, номер машины, аккаунт, номер аккаунта, номер, то эти слова в основном являются синонимами, и так или иначе сводятся к полю Number в таблицах базы данных. В интерфейсе программы поле Number будет обзываться как Account.

Когда виртуальная машина с определенным номером загружается, сторонняя программа обновляет в таблице AccessItems строку с соответствующим Number и пишет в поле Username – имя пользователя, загрузившего виртуалку, а в поле LastDate – дату, когда началась загрузка.

**Структура базы данных**

Таблица AccessItems

|  |  |  |
| --- | --- | --- |
| Название | Тип | Комментарий |
| Id | int | Первичный ключ |
| Number | string | Номер аккаунта |
| Username | string | Пользователь, который загрузил виртуалку с этим номером в текущий момент. Виртуалка запущена пользователем – строка содержит имя пользователя. Виртуалка остановлена пользователем – строка содержит пустое значение. |
| Access | int | Уровень доступа пользователя к данным в программе. Могут быть значения 1, 2, 3, не исключено появление других. |
| LastDate | Datetime | Дата обновления записи |

CREATE TABLE [AccessItems]

(

[Id] integer not null identity primary key,

[Number] nvarchar(40) default '',

[Username] nvarchar(40) default '',

[Access] integer default 0,

[LastDate] datetime default current\_timestamp

);

Пример таблицы AccessItems с данными

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Id | Number | Username | Access | LastDate |
| 3209 | 4200 |  | 1 |  |
| 3210 | 4201 | Admin | 2 | Какая то дата |
| 3211 | 4202 | Admin | 1 |  |
| 3212 | 4203 |  | 3 |  |
| 3113 | 4204 | User4 | 3 |  |

Таблица UsernameItems

CREATE TABLE [UsernameItems]

(

[Id] integer not null identity primary key,

[Username] nvarchar(40) default '',

[Access] integer default 0

);

insert into UsernameItems (Username, Access) values ('Admin', 1);

insert into UsernameItems (Username, Access) values ('User1', 3);

insert into UsernameItems (Username, Access) values ('User2', 2);

insert into UsernameItems (Username, Access) values ('User3', 2);

insert into UsernameItems (Username, Access) values ('User4', 3);

Пример таблицы UsernameItems с данными

|  |  |  |
| --- | --- | --- |
| Id | Username | Access |
| 1 | Admin | 1 |
| 2 | User1 | 3 |
| 3 | User2 | 2 |
| 4 | User3 | 2 |
| 5 | User4 | 3 |

Таблица AccountItems

CREATE TABLE [AccountItems]

(

[Id] integer not null identity primary key,

[Number] nvarchar(40) default '',

[Version] integer default 0,

[Page] nvarchar(4000) default '',

[Username] nvarchar(40) default '',

[LastDate] datetime default current\_timestamp

);

Пример таблицы AccountItems с данными

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Id | Number | Version | Page | Username | LastDate |
| 10000 | 4200 | 1 | Text1 | Admin |  |
| 10001 | 4201 | 1 | Text2 | Admin |  |
| 10002 | 4201 | 2 | Text22 | User2 |  |
| 10003 | 4201 | 3 | Text234 | Admin |  |
| 10004 | 3805 | 15 | Info |  |  |

Таблица StatisticItems (она еще не используется, только придумана). С ее реализацией есть два возможных подхода. Либо хранить поля с FirstDateValue по Name2CommentStatus в одном поле в формате json, либо все отдельно по полям.

|  |  |  |
| --- | --- | --- |
| Название | Тип | Комментарий |
| Id | int | Первичный ключ |
| Number | string |  |
| Version | int |  |
| Status | int |  |
| Parent | string |  |
| Username | string |  |
| LastDate | datetime |  |
| FirstDateValue | string |  |
| FirstDateStatus | int |  |
| CommentValue | string |  |
| CommentStatus | int |  |
| Name1BalanceValue | string |  |
| Name1BalanceStatus | int |  |
| Name1CommentValue | string |  |
| Name1CommentStatus | int |  |
| Name2BalanceValue | string |  |
| Name2BalanceStatus | int |  |
| Name2CommentValue | string |  |
| Name2CommentStatus | int |  |

В программе есть вкладка Account – она отображает данные из таблицы AccountItems.

Есть вкладка Statistic – она отображает данные из StatisticItems.

Еще есть две вспомогательные таблицы UsernameItems и AccountItems, они устанавливают ограничение для просмотра пользователями данных из таблиц AccountItems и StatisticItems.

В UsernameItems устанавливаются уровни доступа для каждого пользователя.

В AccountItems для каждого аккаунта установлен уровень доступа, т.е. ограничение по пользователям которые могут смотреть данные.

Поясню на примерах.

В таблице UsernameItems для Admin поле Access = 1, Admin может видеть все номера из AccessItems, где Access равно 1 или 2 или 3.

В таблице UsernameItems для User4 поле Access = 3, значит он имеет доступ ко всем номерам для которых в таблице AccessItems поле Access = 3. А где значение поля Access 1 или 2 он уже не может просматривать данные.

Если для User3 Access = 2 из UsernameItems, значит он можеть видеть все номера из AccessItems где Access = 2 или 3, ну и соответственно загрузить из таблиц AccountItems или StatisticItems только то что он может видеть согласно AccountItems.

В таблицах AccountItems и StatisticItems записи не обновляются. При любом изменении данных у аккаунта происходит добавление новой записи с этим же номером и следующим номером версии. Т.е. изменения происходит так:

|  |  |
| --- | --- |
| Number | Version |
| 4500 | 1 |
| 4500 | 2 |
| 4500 | … |
| 4500 | 30 |
| 4500 | … |

Актуальная версия – всегда с последним Version для этого Number.

Таблица AccessItems содержит одну строку на каждый номер.

Информация по интерфейсу программы.

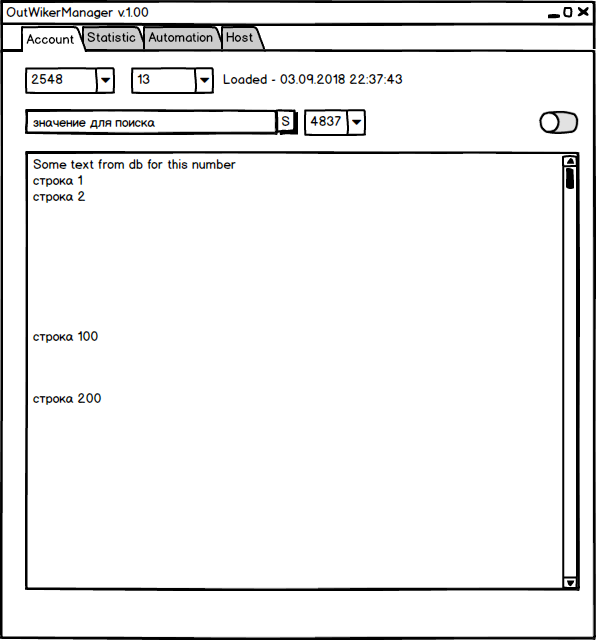
На вкладках логика контролов в программе на сервере и в виртуальных машинах немного различается. Для сервера будет обозначение на главной или серверной машине, для виртуальных машин – на виртуальной машине.

Запущенное на серверной машине приложение может работать с любым из номеров, а запущенной с виртуалки, приложение работает только с номером этой виртуалки.

Для определения запущено приложение под виртуалкой или под сервером, сделайте так – в папке с приложением файл текстовый с произвольным именем – если в нем пусто, то тогда сервер, а если в нем есть строка и в ней 4-5 значный номер – виртуалка. Определяется при старте программы, и не изменяется в дальнейшем. На этом пусть основывается вся логика.

В виртуальной машине у нас устанавливается одно значение Number на все вкладки при запуске и изменению не подлежит, все контролы связанные с ним становятся readonly.

**Вкладка Account**



Первый элемент – Combobox с номером аккаунта.

Второй элемент – Combobox с номером версии этого аккаунта.

Третий элемент – Label со статусом. Статусы пропишу отдельно ниже.

Четвертый элемент – Комбинация контролов для поиска. Можно сделать их в expander или еще как то, так как использоваться будут редко, но они нужны.

Пятый элемент – кнопка включения / отключения режима редактирования для шестого элемента.

Шестой элемент – большой Textbox с переносом строк, собственно сама нужная информация, в базе это поле Page.

Теперь по логике.

Первый элемент – Combobox с номером аккаунта в серверной машине содержит список всех машин. Тупо список, без учета уровня доступа. Либо выбор из списка, либо вводом с клавиатуры или вставкой из буфера обмена номера. Получить список можно наверное по AccessItems. Или distinct от всех номеров в AccountsItems. (Надо обдумать). В виртуальной машине поле становится readonly и содержит один номер аккаунта, без возможности выбора.

Второй элемент – Combobox с номером версии изменения данных по этому аккаунту. В серверной версии содержит список всех номеров по порядку, по умолчанию всегда загружается последняя актуальная версия. Она всегда readonly, но если нажать на элемент пять, то включится режим редактирования для большого Textbox. Ввод запретить, только выбор из списка. По сути эта возможность нужна для readonly просмотра истории изменений. Изменили значение version загрузили соответствующее изменение. Редактирование возможно только последней актуальной версии, и тут уже зависит от пятого элемента. Для виртуальной машины всегда отображается только последняя версия, readonly и с отключенным выбором. Как обычный readonly textbox по поведению.

Третий элемент – Label со статусом, состоит из статуса ниже и времени когда произошло изменение статуса (Обычно DateTime.Now). Время в виде 05.09.2018 00:59:34. Например.

“Loaded - 05.09.2018 00:59:34”

Возможные статусы

None, Loaded, Saved, Denied, Error

None – пока данные не загружены. Первый раз пока не загружены, и когда сеть отвалится, тоже устанавливается.

Loaded – данные загружены из базы актуальная версия.

Saved – данные сохранены в базу.

Denied – если пользователь выбрал номер аккаунта, который ему запрещен к просмотру.

Error – произошла ошибка или исключение с базой, например к базе не подключается (бывает часто, особенно пару секунд после восстановления виртуалки, пока сеть недоступна)

Четвертый элемент – возможность поиска конкретного слова/текста в поле Page среди всех номеров и получение списка этих номеров. Поиск среди истории тоже, отбирать актуальные не нужно. Пока ничего лучше не придумал как сделать из текстового поля, куда вводится значение для поиска, кнопка Search которая проводит асинхронный поиск, и combobox со списком номеров где есть эти значения. Только выбор без текстового ввода, при выборе любого найденного номера из combobox этот номер устанаваливается в первом элементе combobox и загружается данные аккаунта.

Пятый элемент – кнопка для включения редактирования большого textbox (шестого элемента).

Актуально только при последней загруженной версии аккаунта. При выборе не последнего Version для этого Number его можно сделать disabled (это получается только в серверной версии возможно, в виртуалке всегда актуальная версия)

Шестой элемент – textbox где собственно отображается весь текст из поля Page. Одинаковое поведение на серверной и виртуальных машинах.

Теперь по актуализации данных.

Данные в базу могут сохраняться и изменяться извне.

*Т.е. например если в виртуалке и сервере открыт один аккаунт, то при изменении в одном месте, спустя пару секунд он должен отобразиться во втором.*

По факту это выглядит как если у нас выбран какой то номер, мы сразу и затем каждые 2 секунды ходим в базу получить актуальную версию, статус устанавливается Loaded + время. При сохранении мы загрузку из базы отключаем на время, делаем сохранение, показываем на 2 секунды статус Saved + время, и затем снова включаем загрузку из базы каждые 2 секунды.

Это ключевой момент по данным и по сложности. Все должно происходить в асинхронном режиме. Без подтормаживаний и подвисаний, даже если запрос идет в базу когда сеть отвалилась.

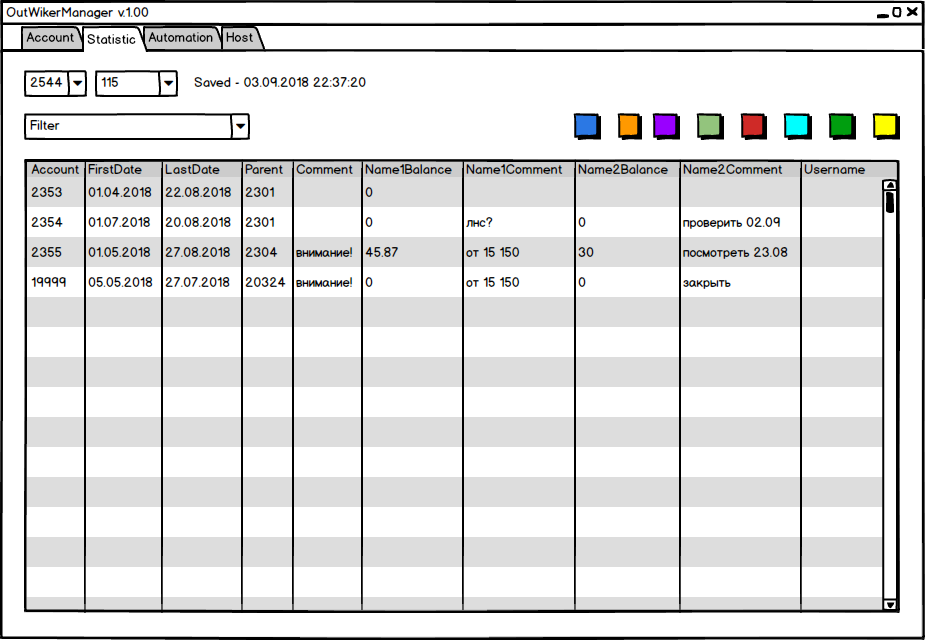
Когда мы переходим между вкладками Account, Statistic, при активации соответствующей вкладки у нас должна произойти сразу и включиться раз в 2 секунды загрузка (актуализация) из базы. Когда мы ушли со вкладки (деактивировали ее), загрузку (актуализацию) из базы нужно прекратить, чтобы базу не перегружать.

Второй момент – мы загрузили любой аккаунт, возможность редактирования у нас по умолчанию отключена, т.е. стоит режим readonly. Если пользователь нажал кнопку и включил редактирование, через 30 минут должна возможность редактирования отключиться, и включиться снова readonly. 30 минут должны идти не от таймера, а от времени установления редактирования. Как разница между текущим временем и временем установления редактирования станет больше 30 минут, отключаем возможность редактирования. Так как виртуалки все время запускаются / приостанавливаются, текущее время сильно скачет, это учитываем.

В целом на программу сделать чтобы когда происходило событие NetworkChange.NetworkAvailabilityChanged и в нем if (!e.IsAvailable), программа осуществляла два действия:  
- окно программы сворачивалось.

- большой textbox на вкладке Account обнулялся со статусом None без времени, а затем через 2 секунды снова начиналась загрузка из базы.

**Вкладка Statistic**



Первый элемент – Combobox со списком номеров машин.

Второй элемент – Combobox со списком версий выбранного номера.

Третий элемент – Label со статусом текущего действия.

Первые три элемента аналогичны как на вкладке Account, с поправкой что они работают для вкладки Statistic.

Четвертый элемент – Combobox для фильтрации аккаунтов по определенным критериям.

Пятый элемент – набор кнопок по цветам.

Шестой элемент – DataGrid со всеми данными.

Шестой элемент – DataGrid который отображает всю информацию. Содержит поля из таблицы StatisticItems в следующем порядке Account (в базе Number), Parent, FirstDate, Comment, Name1Balance, Name1Comment, Name2Balance, Name2Comment, (NameNBalance, NameNComment), Username, LastDate.

DataGrid обладает следующими фичами:

- вертикальное выделение любой группы ячеек. Т.е. как в Excel к примеру, можно выделить любой диапазон ячеек, так и тут, любые ячейки выделить можно.

- после выделения можно нажать на любую кнопку с цветом у установить у всех выделенных ячеек статус (если поле этого позволяет)

- небольшой header у строки, по клику на который выделяется сразу вся строка.

- горизонтальный и вертикальный скролбары.

- если установить цвет в ячейке account любой строки – то сразу устанавливается такой же цвет у всех других ячеек этой строки.

- текст в ячейках должен переноситься.

- текст в ячейках в режиме редактирования также должен переноситься, а не в одной строке.

- ячейки Name1Balance должны поддерживать либо отображение строки, либо если начинается строка на = операции плюс, минус, умножение, деление и скобки.

Если начинается на равно, то в режиме просмотра, должна произойти попытка посчитать числа после равно согласно арифметическим законам (если посчитать невозможно, то просто отобразить как текст). В режиме редактирования отображается как есть строка.

Если начинается не на равно, то отображаем и в режиме редактирования, и в режиме просмотра ячейки как есть.

Кнопки с цветом предназначены для установки этого цвета выделенным ячейкам в DataGrid.

Фильтры в StatisticItems

|  |  |
| --- | --- |
| All | Все Number из базы последней версии |
| All Opened | Все Number из базы последней версии где Status = 1 |
| All Closed | Все Number из базы последней версии где Status = 2 |
| All 1 | Все Number из базы последней версии где Status = 1 и Name2Balance = 0 и Name2Comment = оранжевый |
| All 2 | Все Number из базы последней версии где Status = 1 и Name2Balance > 0 и Name2Comment = оранжевый |
| All 3 | Все Number из базы последней версии где Status = 1 и Name2Balance = 0 и Name2Balance = фиолетовый |
| All 4 | Все Number из базы последней версии где Status = 1 и Name2Balance > 0 и Name2Balance = фиолетовый |
| All 5 | Все Number из базы последней версии где Status = 1 и Name2Balance = 0 и Name2Balance = зеленый и Name2Comment = зеленый и Name1Balance = синий и Name1Comment = синий |

Также как и на вкладке Account должна работать актуализация данных из базы.

Все номера которые отображаются в таблице DataGrid должны обновлять значения из базы сразу при активации вкладки, и затем каждые 2 секунды ходить в базу, загружать актуальную версию, помечать статус как Loaded + время.

Когда мы закончили редактирование в любой ячейке, должно произойти обновление данных также в базе в автоматическом режиме без всяких запросов и уведомлений. При этом у нас отключается загрузка данных из базы, обновляются данные в базе, устанавливается на 2 секунды статус Saved + время, и затем включается снова загрузка обновлений из базы.

Статусы такие же как в вкладке Account.

При выборе в таблице любой строки поле Account(Number) этой строки должно скопироваться в первый элемент Combobox вкладки Account, чтобы если мы перейдем на вкладку Account у нас автоматически подгрузились данные.

В вкладке Statistic в виртуальной машине в элементе с номером загружен номер машины, в элементе с версией последняя версия, элемент с фильтром по умолчанию пустой. При этом все эти контролы readonly или disabled чтобы возможность выбора вообще отключить. А в таблице всегда одна строка номера этой машины.

В вкладке Statistic в серверной машине поведение расширяется.

Если в элемент с номером введен или выбран какой либо номер, то поле с версией активно.

При этом поле с фильтром отключено.

Т.е. если выбран какой то номер, то загружается только этот номер, и фильтры отключаются.

Если номера нет, доступно поле с фильтром, в котором можно выбрать уже нужную задачу.

В поле с фильтром отбор номеров из базы осуществляется асинхронно после выбора нужного фильтра.

Про историю версий.

Если у нас выбран какой то номер и по дефолту стоит последняя версия, то можно выбрать любую другую прошлую версию. После выбора произойдет загрузка одной строки с этой версией в таблице. Возможность редактирования должна быть отключена!

Редактирование доступно только когда выбрана последняя версия.

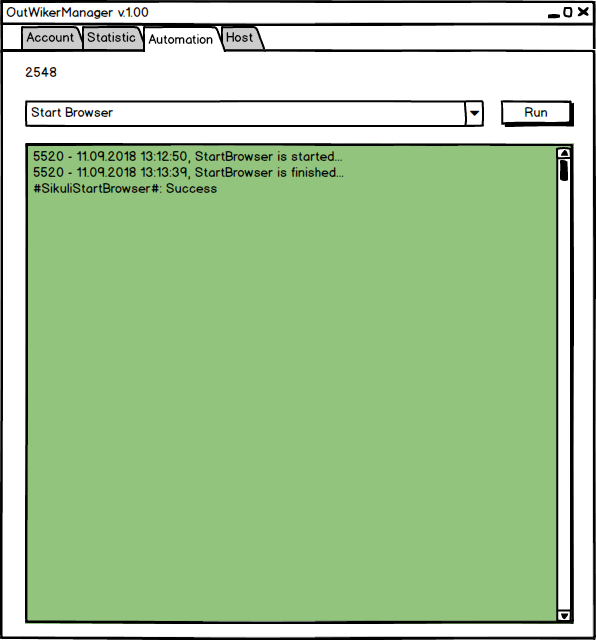
Структура таблицы StatisticItems еще существует только на бумаге, ее нужно обдумать и скорей всего выбрать из двух вариантов.

Особенность в том что поля NameNBalance, NameNComment могут добавляться в базу данных, и в идеале это делать без перекомпиляции программы, но при этом чтобы программ их подхватывала.

Но нужно где то задавать нужно ли их отображать в интерфейсе программы. Возможно придется создать отдельную таблицу в бд для учета этого. Нуждается в обсуждении в процессе разработки!

Также пока точно нет определенности с кнопками с цветами. Возможно их придется рисовать динамически, а подгружать список кнопок из базы. В процессе разработки будет решено.

**Вкладка Automation**



Первый элемент – Label с выбранным Number в вкладке Account. Обновляется при активации вкладки. Если номер не выбран тогда отображать None.

Второй элемент – Combobox со списком задач. Список и сами задачи предопределены в коде при старте программы.

Задачи которые будут тут находиться - производные от базового класса AutoTask

public class AutoTask

{

private string \_name = "";

private string \_displayName = "";

public string Name { get => \_name; set => \_name = value; }

public string DisplayName { get => \_displayName; set => \_displayName = value; }

public virtual string Execute()

{

return RunProject(); // реализацию убрал для краткости

}

public virtual void Cancel()

{

ProcessModel.CloseProcess(Config.Instance.JavaName);

}

}

Поле DisplayName отображается в Combobox.

В Combobox засунуть одну тестовую задачу. Задачи буду добавлять самостоятельно по мере необходимости.

Третий элемент Button с текстом Run или Cancel. Выбранная в Combobox задача будет запускаться нажатием на Run в асинхронном режиме. Для информации в Execute будет запускаться виртуальная машина Java с определенной уже сделанной задачей, можно для простоты просто имитировать запуск какого нибудь консольки с паузой на 1-2 минуты и записью в stdout. При старте задачи обнуляем textbox лог, пишем в него начало задачи со скриншота, при завершении добавляет окончание задачи плюс текст что вернет Execute. После нажатия на кнопку Run и пока выполняется задача текст на ней должен быть Cancel. Как задача завершиться – снова вернуть Run. Когда задача выполняется, т.е. отображается Cancel нужно чтобы также работала отмена выполнения – метод Cancel (который пока просто тупо прибивает процесс java, но от задачи к задаче может измениться).

Кнопка Run должна быть доступна только когда запущено в виртуальной машине и только когда есть номер, т.е. не None.

Четвертый элемент – Textbox с логом, в котором пишется начало действия задачи, окончание и то что вернется из Execute.

Если текст в логе удовлетворяет условию

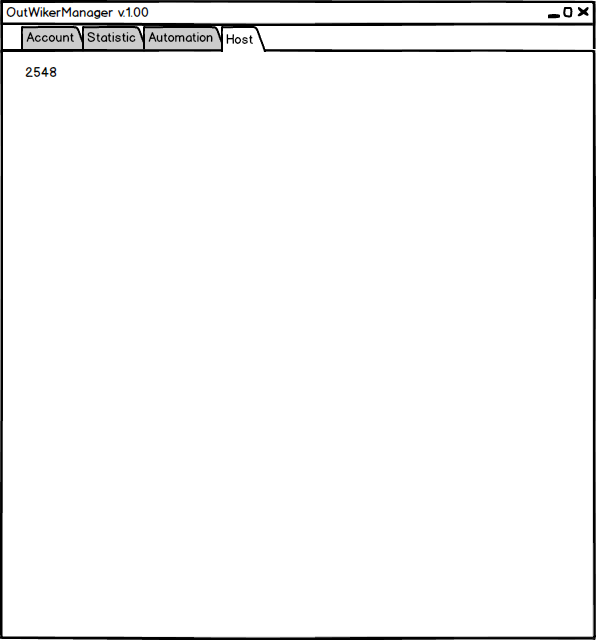
Regex.IsMatch(LogText, ": Success(\r\n)\*$")

То тогда подсвечиваем фон в textbox светло зеленым, если нет – тогда ненасыщенным красным.

Textbox должен всегда прокручиваться вниз автоматом при добавлении в него текста.

При запуске задачи окно программы должно сворачиваться, при завершении – разворачиваться.

**Вкладка Host**

****

Первый элемент – Label с выбранным Number в вкладке Account. Обновляется при активации вкладки. Если номер не выбран тогда отображать None.

**На что еще обратить внимание**

Если Вы желаете писать с тестами приложение, пожалуйста. Но не обязательно.

Весьма неплохо будет если будет какой нибудь логгер для сбора ошибок, или в целом глобальный логгер. Только формат сбора ошибок – один большой файл на сетевом диске, общий для все виртуалок.

Обязательно использовать MVVM при разработке на WPF. Можно использовать Caliburn.Micro, или ReactiveUI или что нибудь интересное.

Можно сделать приложение с использованием реактивного программирования. Давно хочу поразбираться, а тут будет повод) Но не обязательно.

Неплохо было бы еще как нибудь украсить приложение чтоб было приятней пользоваться вместо стандартный унылых контролов.

Что нибудь из разряда:

<https://github.com/MahApps/MahApps.Metro>

<https://github.com/MaterialDesignInXAML/MaterialDesignInXamlToolkit>

Если в начале работы появятся идеи по улучшению интерфейса пересмотру контролов на более оптимальные из например указанных библиотек, обсудим, скорее всего велком.

Иконки можно брать отсюда.

<https://github.com/MahApps/MahApps.Metro.IconPacks>

Желательно сделать один exe файлик, вот например этим - <https://github.com/Fody/Costura>

Другие библиотеки – по согласованию.