МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение   
высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА ТЕХНОЛОГИЙ ЗАЩИТЫ ИНФОРМАЦИИ

КУРСОВАЯ РАБОТА (ПРОЕКТ)  
ЗАЩИЩЕНА С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ассистент |  |  |  | К.А.Жиданов |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОЙ РАБОТЕ |
| **«КНОПКА ЛЕНИВОГО СОТРУДНИКА»** |
| по дисциплине: Технология программирования |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. | 5541 |  |  |  | А.Е.Михайличенко |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург

2017

Содержание

[«КНОПКА ЛЕНИВОГО СОТРУДНИКА» 1](#_Toc484472204)

[Введение. 3](#_Toc484472205)

[1. Приложение «lazy\_button». 4](#_Toc484472206)

[1.1 Теоретические сведения. 4](#_Toc484472207)

[1.2 Работа с GitHub 10](#_Toc484472208)

[1.3 Таск-трекер «teamer» 14](#_Toc484472209)

[1.4 Приложение «lazy\_button» 17](#_Toc484472210)

[2. Заключение 22](#_Toc484472211)

[Список использованных источников 23](#_Toc484472212)

Приложение «1»…………………………………………………………………24

# Введение.

В данной курсовой работе целью является исследование современных технологий разработки информационных систем.

Для данного исследования будет использована система контроля версия, система отслеживания ошибок и программа для создания автоматической документации. Наш выбор выпал на одну из самых популярных систем контроля версий – GitHub. Систему отслеживания ошибок нам так же предоставил GitHub и называется эта система Issue Bug Tracking System. Для создания автоматической документации была выбрана одна из самых популярных программ –Doxygen. Doxygen на сегодняшний день дает самый большой функционал для создания автоматической документации. Приложение Doxywizard позволяет создать автоматическую документацию из шаблонных комментариев к проекту на многих языках. Для того, чтобы нам, было, более понятно дадим несколько простых определений.

Объект работы – приложение «lazy\_button».

Предмет – программирование на C#.

Актуальность данной работы заключается в том, что на сегодняшний день индустрия удобства пользования различными приложениями занимает важное место в развитии техники.

# 1. Приложение «lazy\_button».

## 1.1 Теоретические сведения.

C# (произносится *си шарп*)  объектно-ориентированный язык программирования. Разработан в 1998 - 2001 годах группой инженеров под руководством Андерса Хейлсберга в компании [Microsoft](https://ru.wikipedia.org/wiki/Microsoft) как язык разработки приложений для платформы Microsoft .NET Framework и впоследствии был стандартизирован как ECMA-334 и ISO/[IEC](https://ru.wikipedia.org/wiki/IEC) 23270.

С# относятся к семье языков с С-подобным синтаксисом, из них его синтаксис наиболее близок к С++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, свойства, обобщенные типы и методы, итераторы, анонимные функции с поддержкой замыкании, LINQ, исключения, комментарии в формате XML.

C# разрабатывался как язык программирования прикладного уровня для CLR и, как таковой, зависит, прежде всего, от возможностей самой CLR. Это касается, прежде всего, системы типов C#, которая отражает BCL. Присутствие или отсутствие тех или иных выразительных особенностей языка диктуется тем, может ли конкретная языковая особенность быть транслирована в соответствующие конструкции CLR. Так, с развитием CLR от версии 1.1 к 2.0 значительно обогатился и сам C#; подобного взаимодействия следует ожидать и в дальнейшем (однако, эта закономерность была нарушена с выходом C# 3.0, представляющего собой расширения языка, не опирающиеся на расширения платформы .NET). CLR предоставляет C#, как и всем другим .NET-ориентированным языкам, многие возможности, которых лишены «классические» языки программирования. Например, сборка мусора не реализована в самом C#, а производится CLR для программ, написанных на C# точно так же, как это делается для программ на VB.NET, J# и др.

Система контроля версий — это система, регистрирующая изменения в одном или нескольких файлах с тем, чтобы в дальнейшем была возможность вернуться к определённым старым версиям этих файлов. Для примеров в этой книге мы будем использовать исходные коды программ, но на самом деле под версионный контроль можно поместить файлы практически любого типа.

Если вы графический или веб-дизайнер и хотели бы хранить каждую версию изображения или макета — а этого вам наверняка хочется — то пользоваться системой контроля версий будет очень мудрым решением. СКВ даёт возможность возвращать отдельные файлы к прежнему виду, возвращать к прежнему состоянию весь проект, просматривать происходящие со временем изменения, определять, кто последним вносил изменения во внезапно переставший работать модуль, кто и когда внёс в код какую-то ошибку, и многое другое. Вообще, если, пользуясь СКВ, вы всё испортите или потеряете файлы, всё можно будет легко восстановить. Вдобавок, накладные расходы за всё, что вы получаете, будут очень маленькими.

[Локальные системы контроля версий](https://git-scm.com/book/ru/ch1-1.html#%D0%9B%D0%BE%D0%BA%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5-%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B-%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D1%8F-%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D0%B9)

Многие предпочитают контролировать версии, просто копируя файлы в другой каталог (как правило добавляя текущую дату к названию каталога). Такой подход очень распространён, потому что прост, но он и чаще даёт сбои. Очень легко забыть, что ты не в том каталоге, и случайно изменить не тот файл, либо скопировать файлы не туда, куда хотел, и затереть нужные файлы.

Чтобы решить эту проблему, программисты уже давно разработали локальные СКВ с простой базой данных, в которой хранятся все изменения нужных файлов (см. рисунок 1-1).



Рис. 1.1

Одной из наиболее популярных СКВ такого типа является rcs, которая до сих пор устанавливается на многие компьютеры. Даже в современной операционной системе Mac OS X утилита rcs устанавливается вместе с Developer Tools. Эта утилита основана на работе с наборами патчей между парами версий (патч — файл, описывающий различие между файлами), которые хранятся в специальном формате на диске. Это позволяет пересоздать любой файл на любой момент времени, последовательно накладывая патчи.

[Централизованные системы контроля версий](https://git-scm.com/book/ru/ch1-1.html#%D0%A6%D0%B5%D0%BD%D1%82%D1%80%D0%B0%D0%BB%D0%B8%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B5-%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B-%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D1%8F-%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D0%B9)

Следующей основной проблемой оказалась необходимость сотрудничать с разработчиками за другими компьютерами. Чтобы решить её, были созданы централизованные системы контроля версий (ЦСКВ). В таких системах, например CVS, Subversion и Perforce, есть центральный сервер, на котором хранятся все файлы под версионным контролем, и ряд клиентов, которые получают копии файлов из него. Много лет это было стандартом для систем контроля версий (см. рис. 1-2).



Рис. 1.2

Такой подход имеет множество преимуществ, особенно над локальными СКВ. К примеру, все знают, кто и чем занимается в проекте. У администраторов есть чёткий контроль над тем, кто и что может делать, и, конечно, администрировать ЦСКВ намного легче, чем локальные базы на каждом клиенте.

Однако при таком подходе есть и несколько серьёзных недостатков. Наиболее очевидный — централизованный сервер является уязвимым местом всей системы. Если сервер выключается на час, то в течение часа разработчики не могут взаимодействовать, и никто не может сохранить новой версии своей работы. Если же повреждается диск с центральной базой данных и нет резервной копии, вы теряете абсолютно всё — всю историю проекта, разве что за исключением нескольких рабочих версий, сохранившихся на рабочих машинах пользователей. Локальные системы контроля версий подвержены той же проблеме: если вся история проекта хранится в одном месте, вы рискуете потерять всё.

[Распределённые системы контроля версий](https://git-scm.com/book/ru/ch1-1.html#%D0%A0%D0%B0%D1%81%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D1%91%D0%BD%D0%BD%D1%8B%D0%B5-%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B-%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D1%8F-%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D0%B9)

И в этой ситуации в игру вступают распределённые системы контроля версий (РСКВ). В таких системах как Git, Mercurial, Bazaar или Darcs клиенты не просто выгружают последние версии файлов, а полностью копируют весь репозиторий. Поэтому в случае, когда "умирает" сервер, через который шла работа, любой клиентский репозиторий может быть скопирован обратно на сервер, чтобы восстановить базу данных. Каждый раз, когда клиент забирает свежую версию файлов, он создаёт себе полную копию всех данных (см. рисунок 1-3).



Рис. 1.3

Кроме того, в большей части этих систем можно работать с несколькими удалёнными репозиториями, таким образом, можно одновременно работать по-разному с разными группами людей в рамках одного проекта. Так, в одном проекте можно одновременно вести несколько типов рабочих процессов, что невозможно в централизованных системах.

Система управления версиями Git.

Для работы с системами контроля версий был выбран GitHub — крупнейший веб-сервис для хостинга IT-проектов и их совместной разработки, который основан на системе [Git](https://ru.wikipedia.org/wiki/Git).

Git – это гибкая, распределенная (без единого сервера) система контроля версий, дающая массу возможностей не только разработчикам программных продуктов, но и писателям для изменения, дополнения и отслеживания изменения «рукописей» и сюжетных линий, и учителям для корректировки и развития курса лекций, и администраторам для ведения документации, и для многих других направлений, требующих управления историей изменений. У каждого разработчика, использующего Git, есть свой локальный репозиторий, позволяющий локально управлять версиями. Затем, сохраненными в локальный репозиторий данными, можно обмениваться с другими пользователями. Часто при работе с Git создают центральный репозиторий, с которым остальные разработчики синхронизируются. Пример организации системы с центральным репозиторием – это проект разработки ядра Linux’a. В этом случае все участники проекта ведут свои локальны разработки и беспрепятственно скачивают обновления из центрального репозитория. Когда необходимые работы отдельными участниками проекта выполнены и отлажены, они, после удостоверения владельцем центрального репозитория в корректности и актуальности проделанной работы, загружают свои изменения в центральный репозиторий.  
Наличие локальных репозиторием также значительно повышает надежность хранения данных, так как, если один из репозиториев выйдет из строя, данные могут быть легко восстановлены из других репозиториев.  
Работа над версиями проекта в Git может вестись в нескольких ветках, которые затем могут с легкостью полностью или частично объединяться, уничтожаться, откатываться и разрастаться во все новые и новые ветки проекта.

Основные достоинства этой системы управления версиями:

Надежная система сравнения ревизий и проверки корректности данных, основанные на алгоритме хеширования SHA1 (Secure Hash Algorithm 1).

Гибкая система ветвления проектов и слияния веток между собой.

Наличие локального репозитория, содержащего полную информацию обо всех изменениях, позволяет вести полноценный локальный контроль версий и заливать в главный репозиторий только полностью прошедшие проверку изменения.

Высокая производительность и скорость работы

Удобный и интуитивно понятный набор команд

Множество графических оболочек, позволяющих быстро и качественно вести работы с Git’ом

Возможность делать контрольные точки, в которых данные сохраняются а полностью. Это позволяет уменьшить скорость восстановления данных, так как за основу берется ближайшая контрольная точка, и восстановление идет от нее. Если бы контрольные точки отсутствовали, то восстановление больших проектов могло бы занимать часы.

Широкая распространенность, легкая доступность и качественная документация.

Гибкость системы позволяет удобно ее настраивать и даже создавать специализированные контроля системы или пользовательские интерфейсы на базе Git.

Универсальный сетевой доступ с использованием протоколов HTTP, FTP, RSYNC, SSH и др.

1.3. Система отслеживания ошибок

Система отслеживания ошибок (англ. Bug tracking system) — прикладная программа, разработанная с целью помочь разработчикам программного обеспечения (программистам, тестировщикам и др.) учитывать и контролировать ошибки (баги), найденные в программах, а также следить за процессом устранения этих ошибок.

BTS помогает программисту следить за ошибками. Когда вы замечаете ошибку, необходимо собрать о ней максимальное количество доступной информации. Необходимо быть предельно точным в наблюдениях. Особенно это касается отчетов об ошибках, приходящих от пользователей.

Как правило, BTS позволяет хранить информацию об ошибке в следующем виде:

кто сообщил о проблеме;

дата и время, когда была обнаружена проблема;

серьёзность проблемы;

описание неправильного поведения программы;

кто занимается устранением проблемы;

состояние ошибки.

Это минимальный набор требований к БД BTS, на самом же деле многие системы багтрэкинга позволяют вести намного более подробный учет ошибок. В чем то, они напоминают системы управления проектами. А многие из них интегрированы с такими системами.

Большинство программных проектов имеют трекеры ошибок. На GitHub он носит название «Issues» и имеет свой собственный раздел в каждом репозитории.

## 1.2 Работа с GitHub

Для работы с системой контроля версия нужно пройти процесс регистрации и создать репозиторий — место, где хранятся и поддерживаются какие-либо данные. Для создания требуется ввести имя репозитория, выбрать его тип (закрытый доступен платным подписчикам) и автогенерацию ридми.

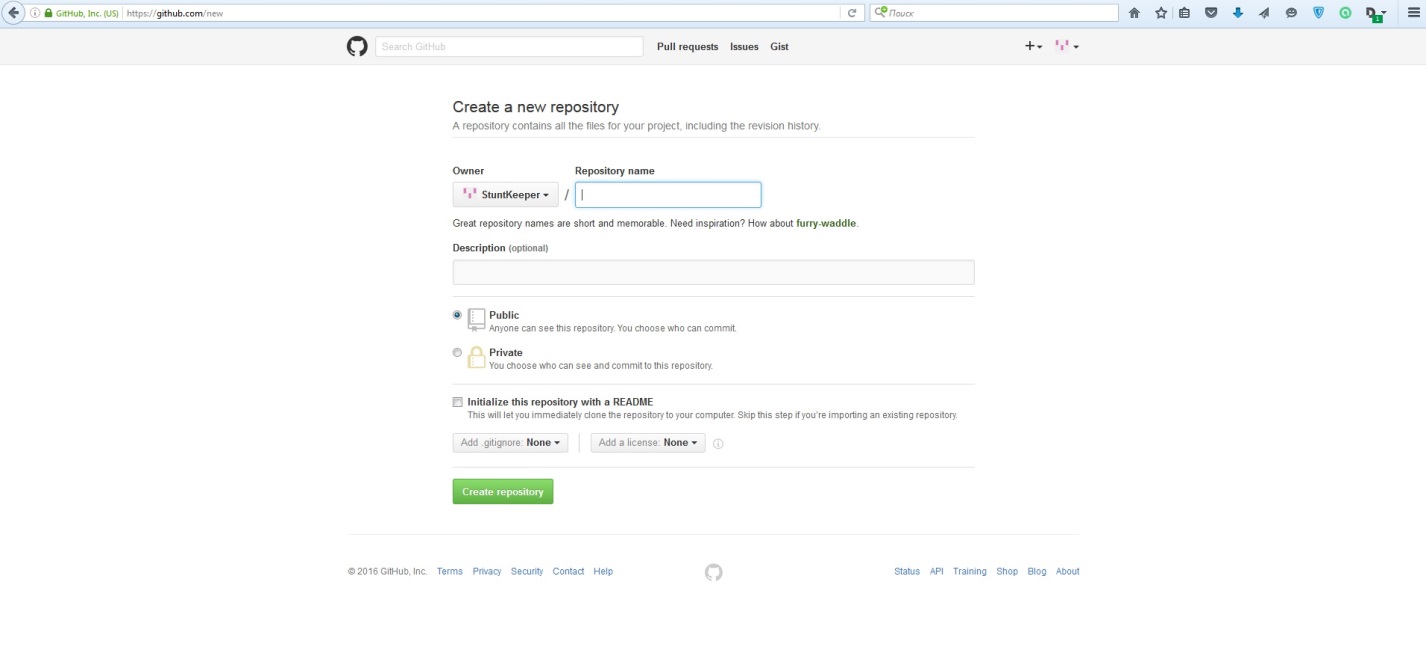


Рис. 1.4 – Создание репозитория

После создания репозитория мы сможем выкладывать наш проект и код в публичный доступ. Но для того, чтобы выложить наш проект в сеть во-первых нужно будет выполнить его сборку в Visual Studio.

После сборки в папке с нашим кодом появляется еще одна папка(Release), в которой мы можем увидеть уже готовую для работы программу.

Теперь наш проект готов к работе с GitHub.C помощью официального приложения мы выкладываем наш проект . Для того ,чтобы выложить наш проект мы копируем его в локальный репозиторий ,создаем коммит и синхронизируем с веб-сервисом.

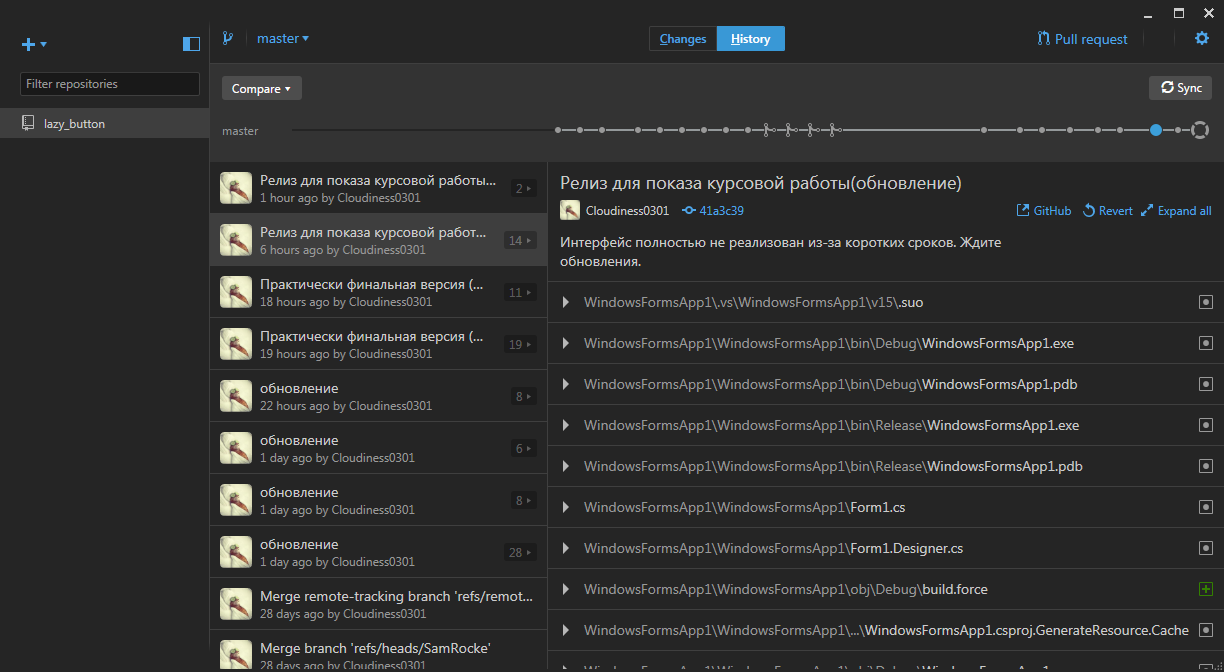


Рис. 1.5 – Синхронизация с репозиторием

После этого мы открываем онлайн версию GitHub и видим, что загрузка нашего проекта и документации прошла успешно. Далее мы создаем README.md для того,чтобы любой из пользователей мог понять ,что представленно в данном репозитории.

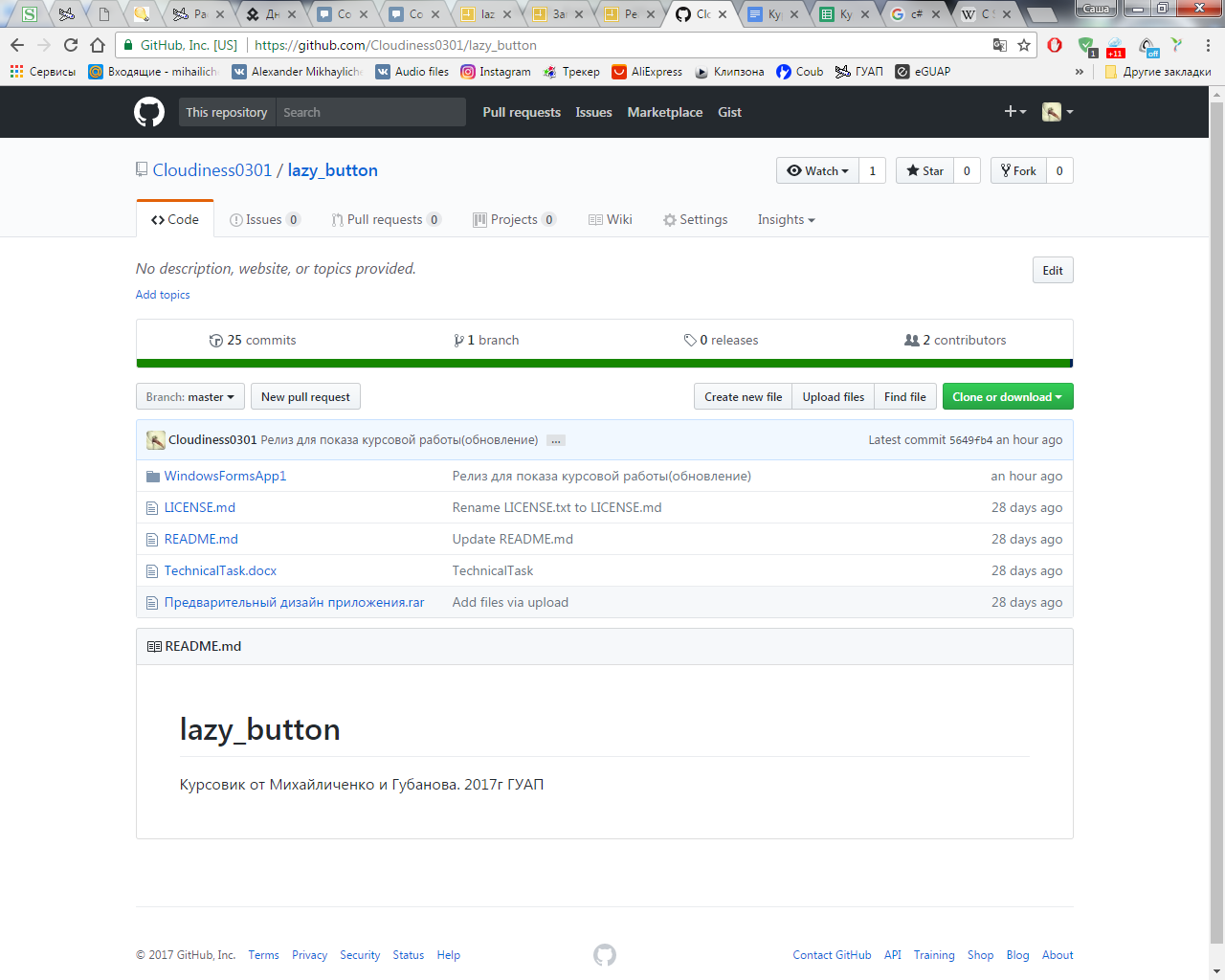


Рис. 1.6 - Репозиторий

Для того чтобы пользователи могли добавлять сообщения об ошибках, уведомлять разработчика об ошибках и помогать в их решение нужно перейти во вкладку «Issue».

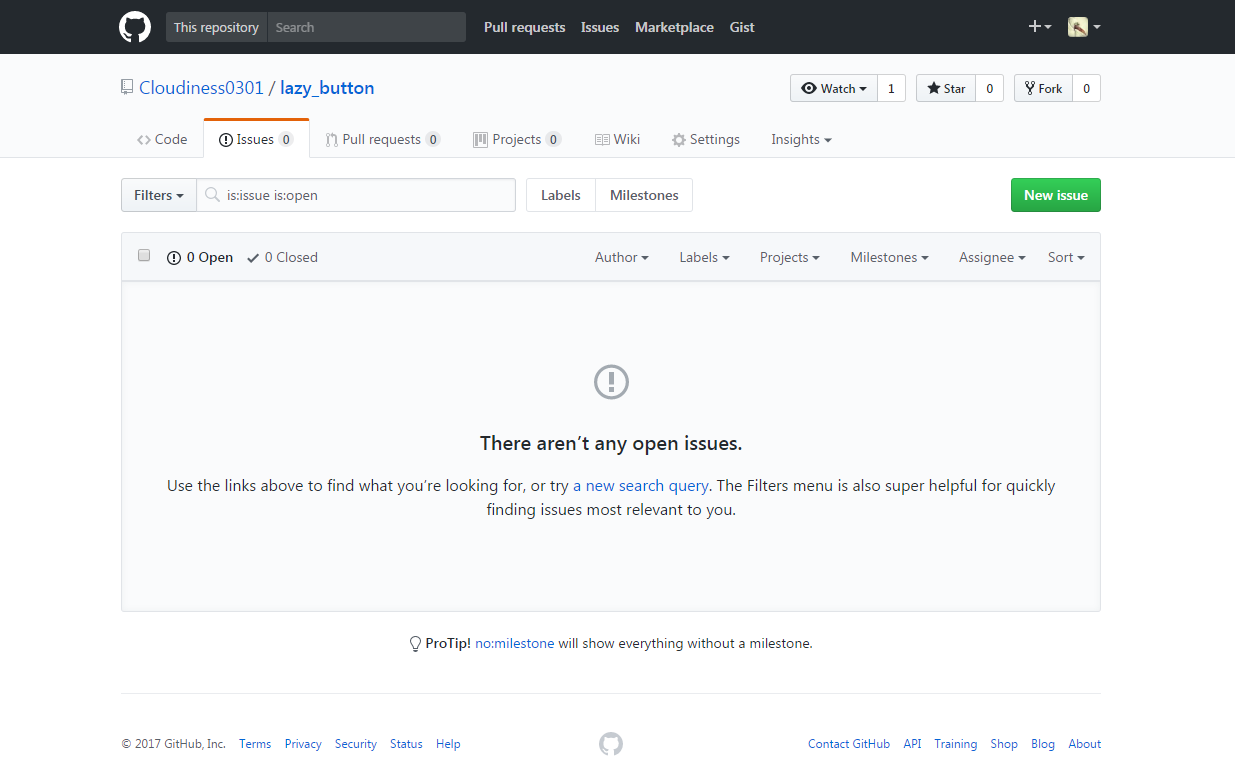


Рис. 1.7 - issue

Для того, чтобы уведомить разработчика об ошибке нужно нажать на кнопку «New issue».В этом окне вы должны четко описать действия ,при которых вы нашли ошибку, и указать на метод решения данной ошибки. Эта проблема будет висеть в данном разделе, пока разработчик не закроет её в связи с внесённым исправлением.

## 1.3 Таск-трекер «teamer»

Ateve Commercial представила новый проект — [Teamer.ru](http://www.onlineprojects.ru/out.php?url=http://www.teamer.ru/). С его помощью организация командной работы станет намного проще — как для компаний, так и для команд фрилансеров. Проект предназначен для использования небольшими группами людей, которые решают некие задачи. При его создании мы сделали упор прежде всего на работу с интернет-проектами. Люди группируются в проекты и ставят друг для друга задания (тикеты). В проектах бывают управляющие (они видят все задания, созданные в рамках проекта) и исполнители (они видят только те задания, которые касаются непосредственно их). Все эти люди могут писать сообщения (комментарии) в рамках заданий и прикреплять к ним файлы (скриншоты, текстовые документы и т.п.).  
  
Основные разделы сайта — «Что делать?», «Кто виноват?» и «Где все?».

В первом автоматически сосредотачиваются текущие и будущие задания.

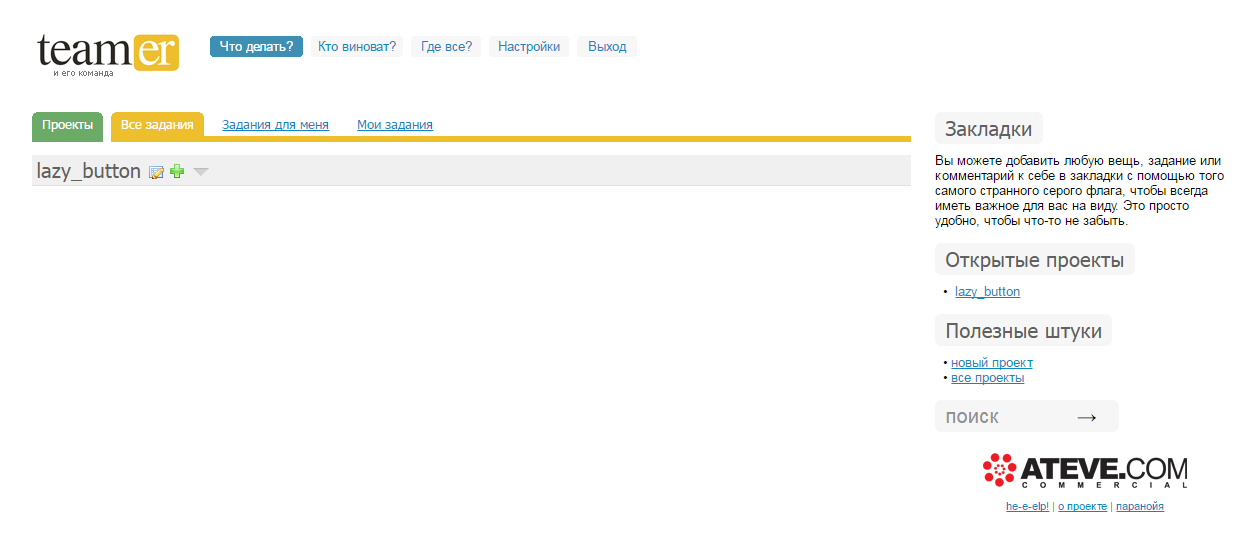


Рис. 1.8 – Что делать?

«Кто виноват?» полезен управляющим проектами: в нём в виде упрощённого календаря отображаются сроки выполнения заданий, в результате чего очень просто определить, кто из исполнителей не выполняет задания вовремя.

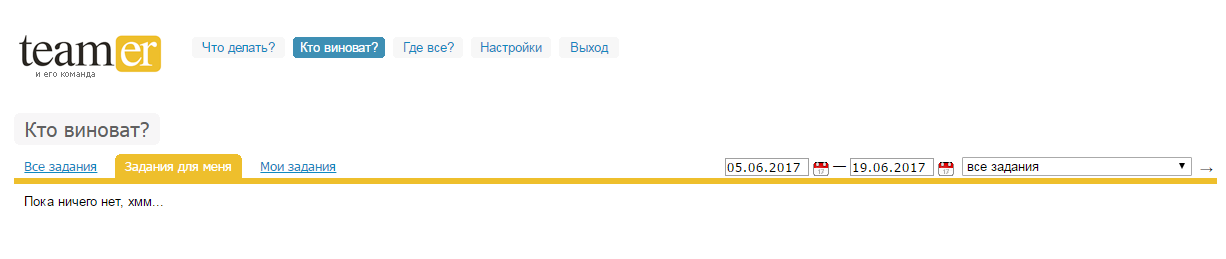


Рис. 1.9 – Кто виноват?

«Где все?» — список Ваших коллег: на этой странице выводится их аватар, имя, роль, а также проекты, в которых Вы являетесь коллегами.

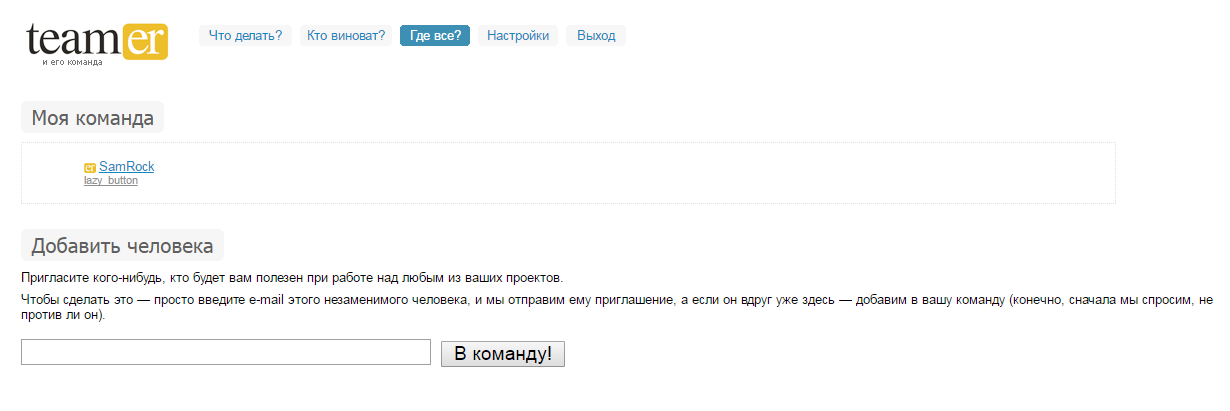


Рис. 1.10 – Где все?

Реализована на сайте и удобная система поиска, которая будет полезна при большом количестве принадлежащих Вам проектов. А для истинных трудоголиков мы сделали кнопку «Мне повезёт»: она используется в тот момент, когда работа есть, а чем конкретно заняться — вы не знаете, не можете определиться.  
  
На страже Вашей уверенности в том, что ни одно сообщение от коллеги или заказчика не будет пропущено, стоят три наших бота, которые рассылают уведомления по e-mail, ICQ и Jabber"у соответственно. Вы узнаете всё, что Вас касается ТБКВ (Так Быстро, Как Возможно ;-).  
  
Весь этот функционал разбавлен красотой, минимализмом, продуманным юзабилити и технологией AJAX. И как принято во всех приличных проектах, его использование — бесплатно.

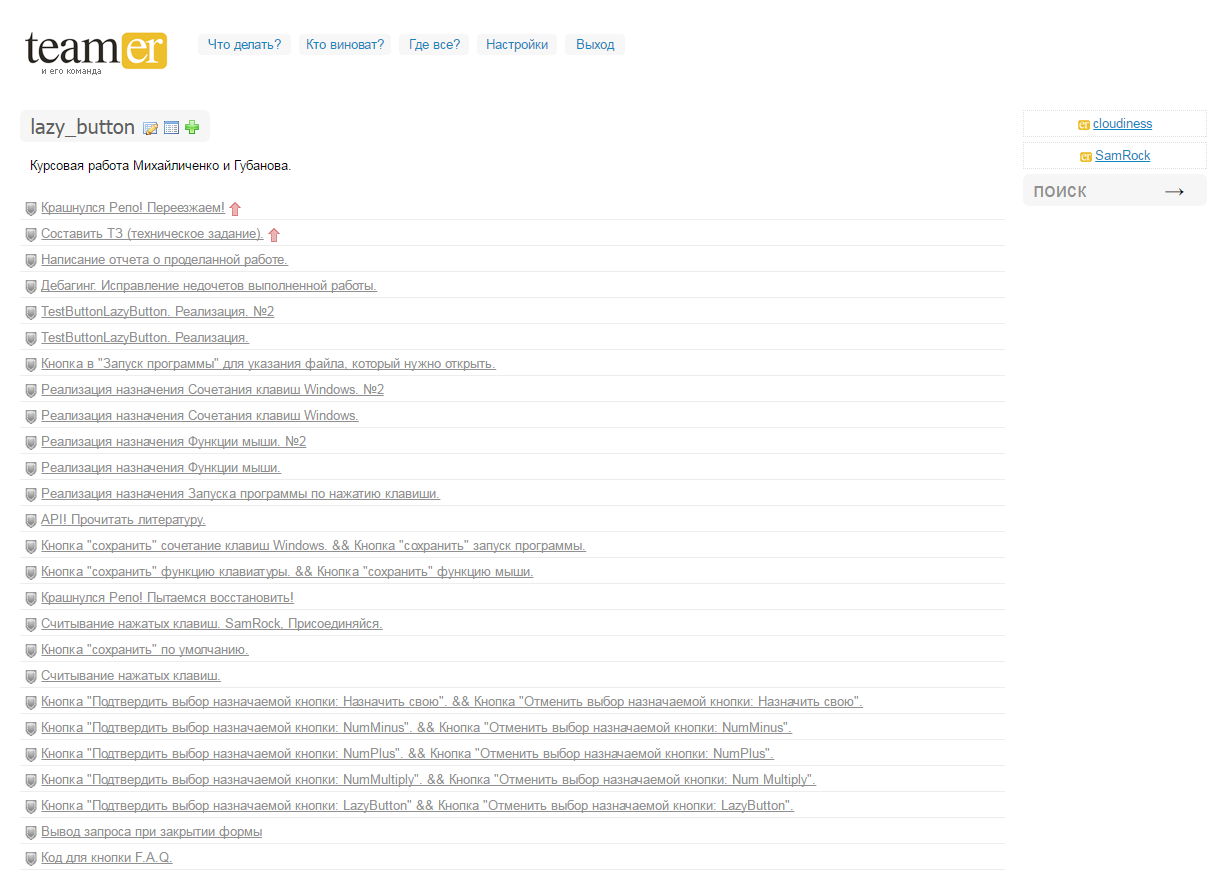


Рис. 1.11 – Список тасков

## 1.4 Приложение «lazy\_button»

Кнопка «ленивый сотрудник». Скрытно нажимаемая (например, ногой) беспроводная кнопка, которая выполняет на компьютере сочетание Alt-tab либо делает активной заданную программу (например, Excel), либо сворачивает текущую активную программу.

Задача данной кнопки состоит в создании системы, которая будет позволять пользователям производить желаемые ими действия нажатием беспроводной кнопки.

Эта прекрасная кнопка позволит вам удобным способом прожимать пальцеломательные комбинации клавиш или одним легким касанием заменит определенный порядок назойливых действий.

Всего один клик и все готово. Вдобавок вы можете ее расположить в удобном для себя месте.

Поставленная задача достигается использованием технологий предоставляемых Arduino/ESP и написанного для данной кнопки программного обеспечения, благодаря которому пользователь может назначить на кнопку желаемые действия.

Ниже приведены скриншоты работы приложения.

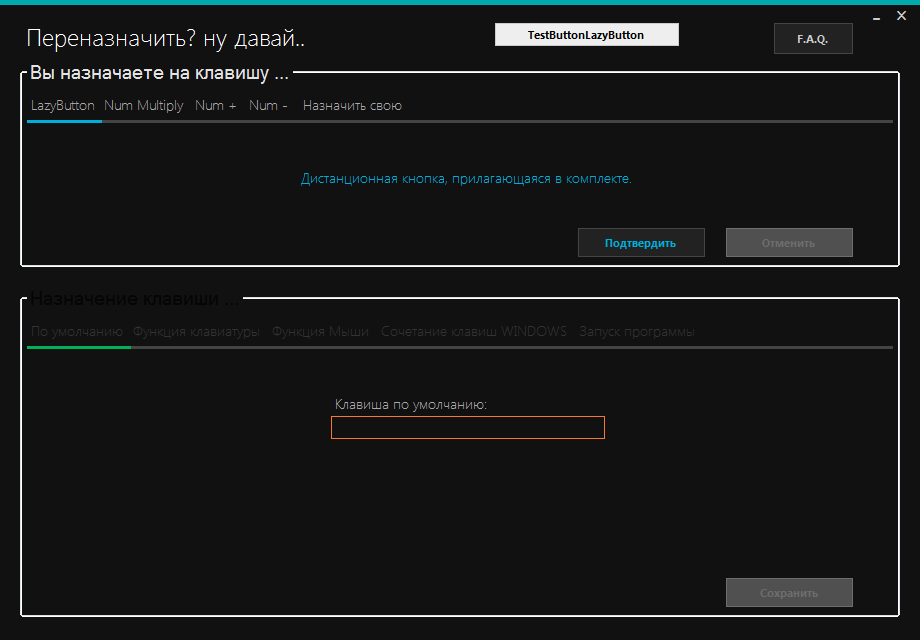


Рис. 1.12 – Стартовый вид

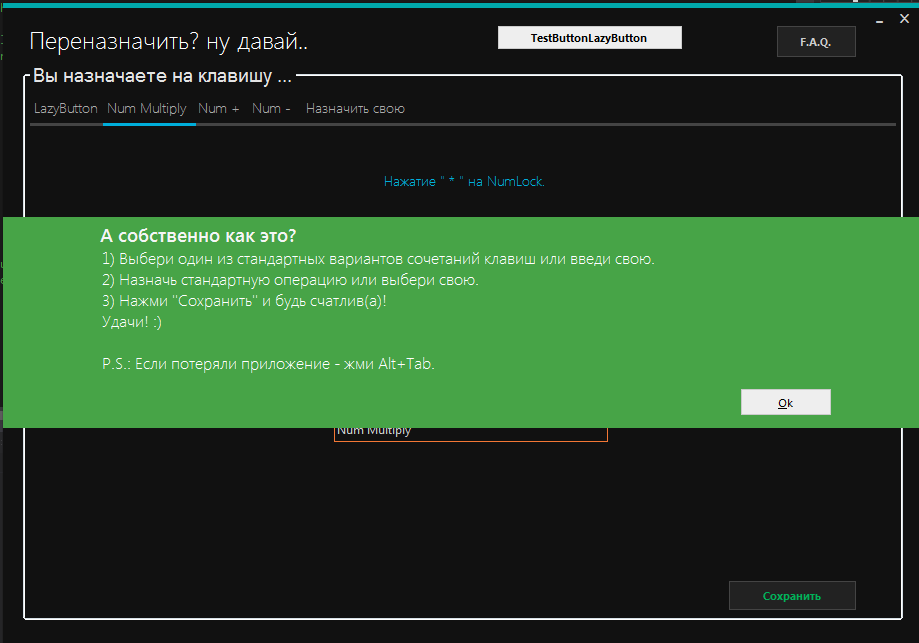


Рис. 1.13 – F.A.Q.

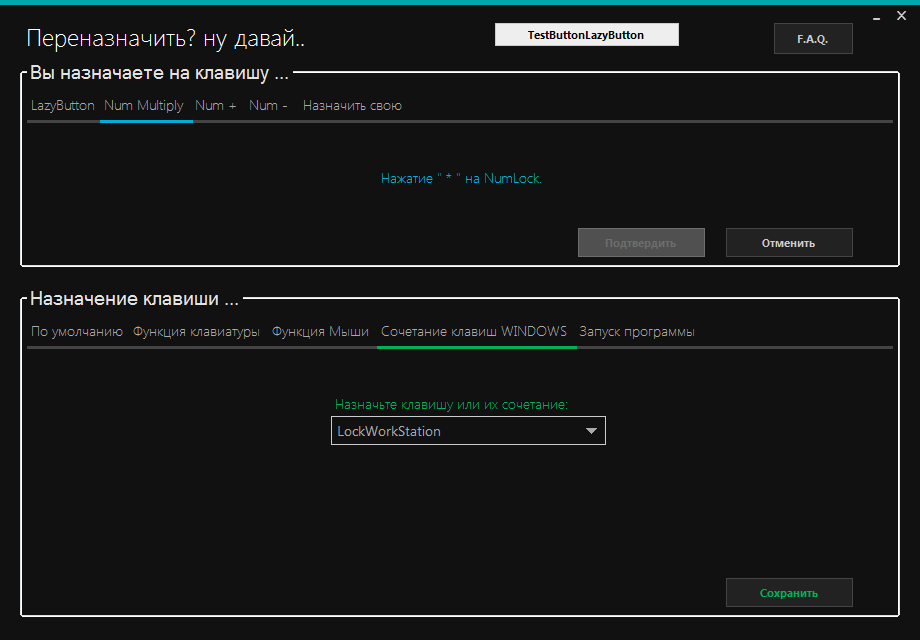


Рис. 1.14 - Назначение на NumMultiply блокировки компьютера

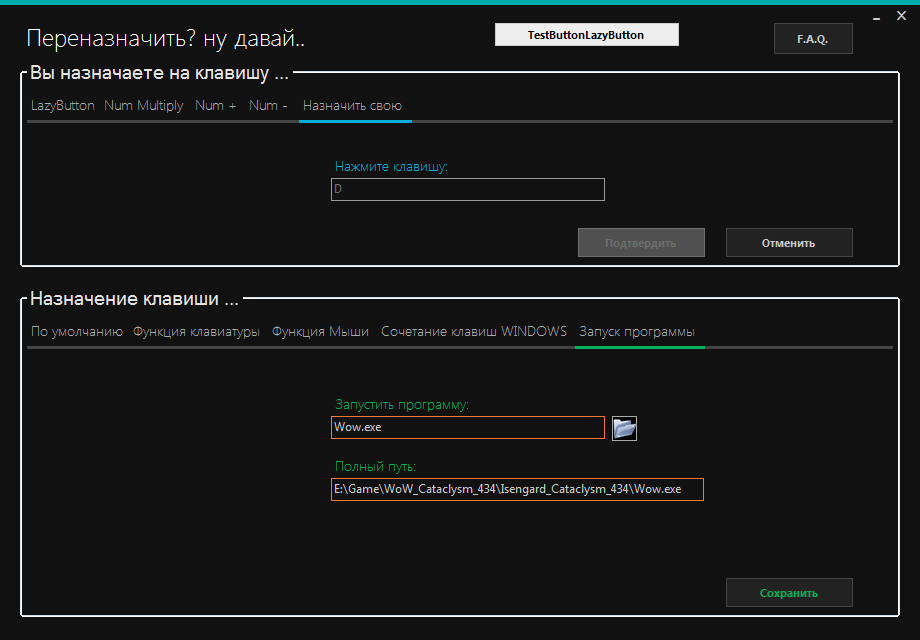


Рис. 1.15 – Назначение на клавишу D открытия программы

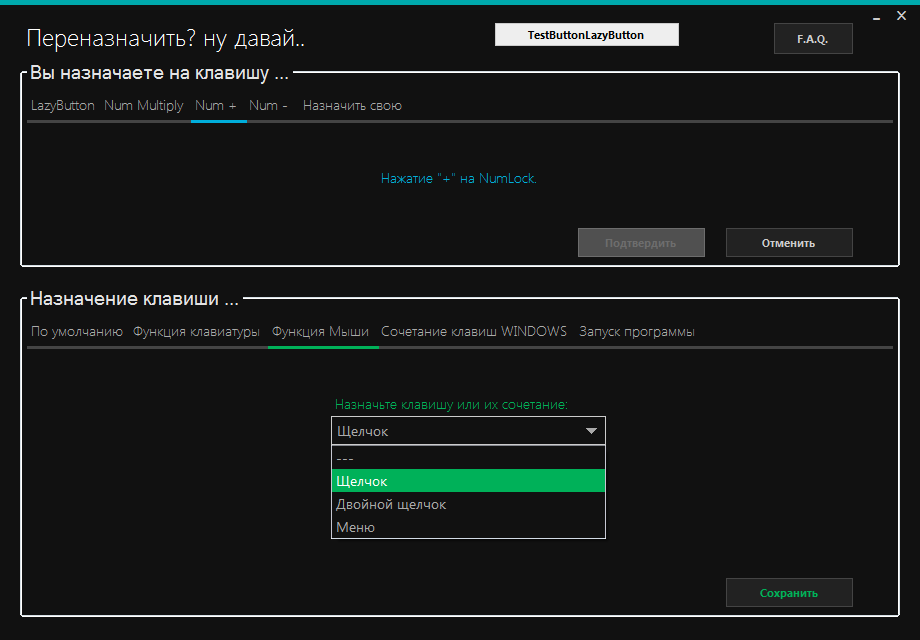


Рис. 1.16 – Меню выбора функции мыши

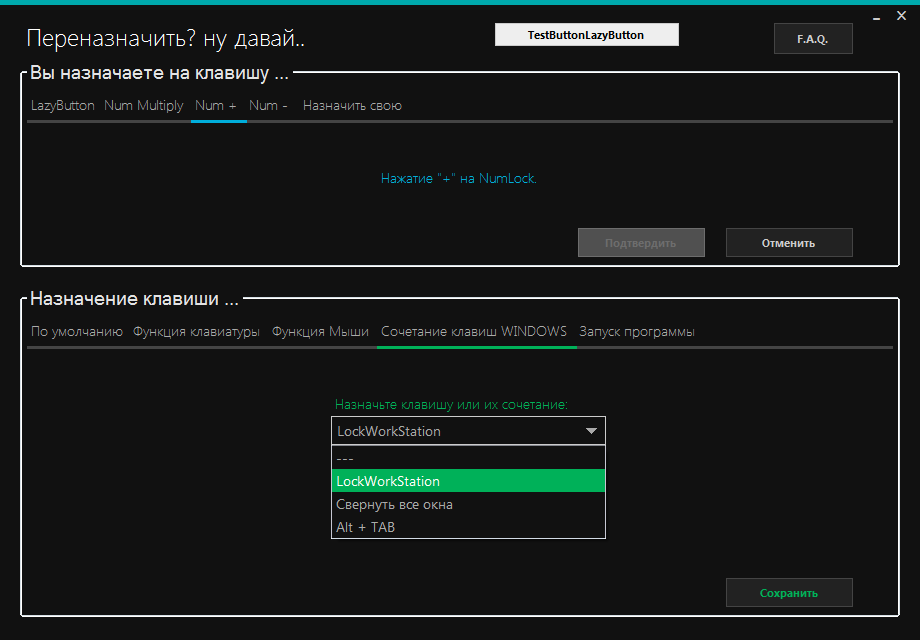


Рис. 1.17 – Меню выбора сочетания клавиш Windows

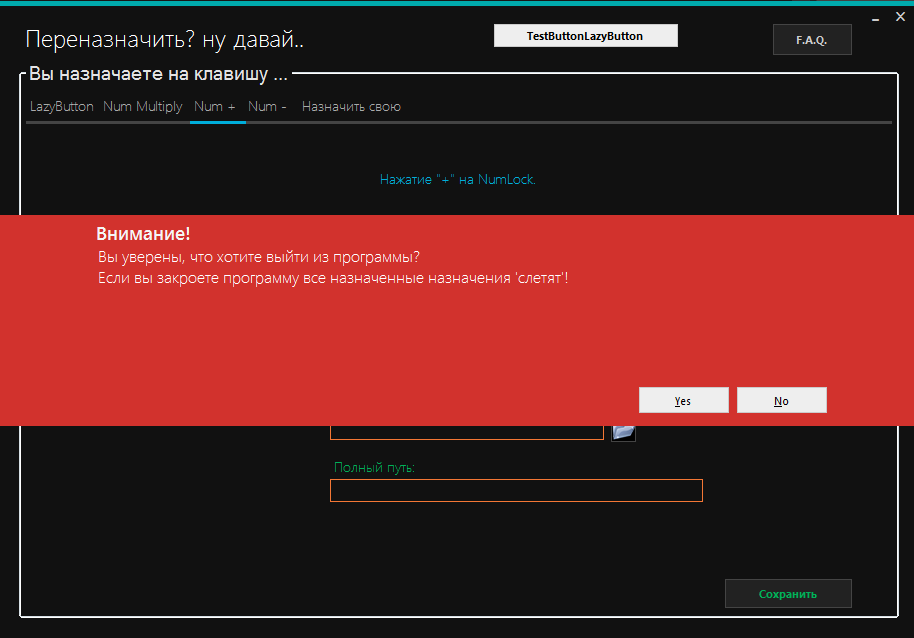


Рис. 1.18 – Предупреждение о закрытии приложения

# 2. Заключение

В ходе данной курсовой работы была написана программа. Для данной курсовой работы было разработано приложение «lazy\_button».

Данное приложение является примитивной версией, но во время разработки оно позволила мне поближе познакомится с базовыми концепциями программирования на С# и основными концепциями ООП.

Программирование приложений – это трудоемкий и кропотливый процесс, требующий особой сосредоточенности, творчества и внимания. Даже для создания самого примитивного, простого приложения у программиста должны присутствовать много времени, сил, энергии и креативность, позволяющая создать сайт для продуктивного, удобного и интересного времяпрепровождения пользователя. Однако сам процесс создания приложения достаточно увлекательный и он компенсирует все физические и моральные затраты на создание и реализацию проекта. Но даже самое примитивное программирование начинается с малого. В первую очередь весь процесс, всю логику необходимо представить на бумаге, а затем уже воплотить в виде кода.

В процессе разработки данной программы, мною была прочитала специальная литература, относящееся к тематике данной проблемы, благодаря которой были усвоены многие практические и теоретические знания. Было освоены основы С#, ООП, WindowsForm. Я с помощью дебагинга быстро и легко находил и исправлял ошибки в коде программы. Закрепил свои знания и навыки программирования на языке С#.

Данное приложение обладает дружелюбным и понятным интерфейсом для освоения информации, доступной на нем, и дальнейшем его использовании. Наше приложение не загружено массой информации, а необходимая расположена в удобном доступе. Цветовая гамма подобрана для приятного чтения.

# 

# Список использованных источников

1. Форум для программистов и сисадминов [Электронный ресурс] / http://www.cyberforum.ru/
2. Ресурсы видеохостинговой компании [Электронный ресурс] / https://www.youtube.com/
3. Wiki [Электронный ресурс] / https://wiki.org/

Приложение «1»

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Drawing.Drawing2D;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using MetroFramework.Components; //для работы метро

using MetroFramework.Forms; //для работы метро

using System.Runtime.InteropServices;

using Microsoft.Win32;

using System.Collections;

using System.Diagnostics;

using Shell32;

namespace LazyButton

{

public partial class Form1 : MetroForm //"Form" было изначально, а не "MetroForm"

{

// Для работы с клавой.

[DllImport("USER32.DLL")]

static extern IntPtr FindWindow(string lpClassName, string lpWindowName);

[DllImport("USER32.DLL")]

static extern bool SetForegroundWindow(IntPtr hWnd);

// Для вызова функции LockWorkStation - блокировка компа = смена пользователя = Win + L.

[DllImport("user32.dll")]

private static extern void LockWorkStation();

// Щелкаем мышкой.

[DllImport("user32.dll")]

static extern void mouse\_event(int dwFlags, int dx, int dy, int dwData, UIntPtr dwExtraInfo);

const int MouseEventF\_Move = 0x0001;

const int MouseEventF\_LeftDown = 0x0002;

const int MouseEventF\_LeftUp = 0x0004;

const int MouseEventF\_RightDown = 0x0008;

const int MouseEventF\_RightUp = 0x0010;

const int MouseEventF\_Absolute = 0x8000;

public Form1()

{

InitializeComponent();

// ShowInTaskbar = false; // скрывает значка формы на панели задач. Если свернул - то можно вызвать только Alt+Tab.

}

private void Form1\_Load(object sender, EventArgs e)

{

}

//

//

// Код для кнопки F.A.Q.

private void FAQ\_Click(object sender, EventArgs e)

{

MetroFramework.MetroMessageBox.Show(this,

"1) Выбери один из стандартных вариантов сочетаний клавиш или введи свою.\n2) Назначь стандартную операцию или выбери свою.\n3) Нажми ''Сохранить'' и будь счатлив(а)!\nУдачи! :)\n\nP.S.: Если потеряли приложение - жми Alt+Tab.",

"А собственно как это?", MessageBoxButtons.OK, MessageBoxIcon.Question);

}

//

//

// Вывод запроса при закрытии формы

private void Form1\_FormClosing(object sender, FormClosingEventArgs e)

{

if (MetroFramework.MetroMessageBox.Show(this, "Вы уверены, что хотите выйти из программы?\nЕсли вы закроете программу все назначенные назначения 'слетят'!",

"Внимание!", MessageBoxButtons.YesNo, MessageBoxIcon.Stop) == DialogResult.No)

{

e.Cancel = true;

}

}

//

//

//!!!\*\*\* ВЕРХЯЯ ГРУППА \*\*\*!!!\\\

//

//

//

//

// GroupButtonsIn\_Enter - Группа кнопок назначаемых клавиш.

private void GroupButtonsIn\_Enter(object sender, EventArgs e)

{

}

//

//

// Кнопка "Подтвердить выбор назначаемой кнопки: LazyButton".

private void LazyButtonButtonОК\_Click(object sender, EventArgs e)

{

LazyButtonButtonCancel.Enabled = true;

LazyButtonButtonОК.Enabled = false;

GroupButtonsOut.Enabled = true;

NumMultiply.Enabled = false;

NumPlus.Enabled = false;

NumMinus.Enabled = false;

UserButton.Enabled = false;

DefaultTextBox1.Text = "ничего не назначено";

}

//

//

// Кнопка "Отменить выбор назначаемой кнопки: LazyButton".

private void LazyButtonButtonCancel\_Click(object sender, EventArgs e)

{

LazyButtonButtonCancel.Enabled = false;

LazyButtonButtonОК.Enabled = true;

GroupButtonsOut.Enabled = false;

NumMultiply.Enabled = true;

NumPlus.Enabled = true;

NumMinus.Enabled = true;

UserButton.Enabled = true;

DefaultTextBox1.Text = "";

KeyboardFunctionTextBox.Text = "";

MouseFunctionComboBox.SelectedItem = "---";

WINDOWSKeyCombinationComboBox.SelectedIndex = 0;

LaunchingTheProgramTextBox.Text = "";

LaunchingTheProgramFileWayTextBox.Text = "";

//ListGroupButtonsIn.Enabled = true;

}

//

//

// Кнопка "Подтвердить выбор назначаемой кнопки: NumMultiply".

private void NumMultiplyButtonOK\_Click(object sender, EventArgs e)

{

NumMultiplyButtonCancel.Enabled = true;

NumMultiplyButtonOK.Enabled = false;

GroupButtonsOut.Enabled = true;

LazyButton.Enabled = false;

NumPlus.Enabled = false;

NumMinus.Enabled = false;

UserButton.Enabled = false;

DefaultTextBox1.Text = "Num Multiply";

}

//

//

// Кнопка "Отменить выбор назначаемой кнопки: Num Multiply".

private void NumMultiplyButtonCancel\_Click(object sender, EventArgs e)

{

NumMultiplyButtonCancel.Enabled = false;

NumMultiplyButtonOK.Enabled = true;

GroupButtonsOut.Enabled = false;

LazyButton.Enabled = true;

NumPlus.Enabled = true;

NumMinus.Enabled = true;

UserButton.Enabled = true;

DefaultTextBox1.Text = "";

KeyboardFunctionTextBox.Text = "";

MouseFunctionComboBox.SelectedItem = "---";

WINDOWSKeyCombinationComboBox.SelectedIndex = 0;

LaunchingTheProgramTextBox.Text = "";

LaunchingTheProgramFileWayTextBox.Text = "";

}

//

//

// Кнопка "Подтвердить выбор назначаемой кнопки: NumPlus".

private void NumPlusButtonOK\_Click(object sender, EventArgs e)

{

NumPlusButtonCancel.Enabled = true;

NumPlusButtonOK.Enabled = false;

GroupButtonsOut.Enabled = true;

LazyButton.Enabled = false;

NumMultiply.Enabled = false;

NumMinus.Enabled = false;

UserButton.Enabled = false;

DefaultTextBox1.Text = "Num +";

}

//

//

// Кнопка "Отменить выбор назначаемой кнопки: NumPlus".

private void NumPlusButtonCancel\_Click(object sender, EventArgs e)

{

NumPlusButtonCancel.Enabled = false;

NumPlusButtonOK.Enabled = true;

GroupButtonsOut.Enabled = false;

LazyButton.Enabled = true;

NumMultiply.Enabled = true;

NumMinus.Enabled = true;

UserButton.Enabled = true;

DefaultTextBox1.Text = "";

KeyboardFunctionTextBox.Text = "";

MouseFunctionComboBox.SelectedItem = "---";

WINDOWSKeyCombinationComboBox.SelectedIndex = 0;

LaunchingTheProgramTextBox.Text = "";

LaunchingTheProgramFileWayTextBox.Text = "";

}

//

//

// Кнопка "Подтвердить выбор назначаемой кнопки: NumMinus".

private void NumMinusButtonOK\_Click(object sender, EventArgs e)

{

NumMinusButtonCancel.Enabled = true;

NumMinusButtonOK.Enabled = false;

GroupButtonsOut.Enabled = true;

LazyButton.Enabled = false;

NumMultiply.Enabled = false;

NumPlus.Enabled = false;

UserButton.Enabled = false;

DefaultTextBox1.Text = "Num -";

}

//

//

// Кнопка "Отменить выбор назначаемой кнопки: NumMinus".

private void NumMinusButtonCancel\_Click(object sender, EventArgs e)

{

NumMinusButtonCancel.Enabled = false;

NumMinusButtonOK.Enabled = true;

GroupButtonsOut.Enabled = false;

LazyButton.Enabled = true;

NumMultiply.Enabled = true;

NumPlus.Enabled = true;

UserButton.Enabled = true;

DefaultTextBox1.Text = "";

KeyboardFunctionTextBox.Text = "";

MouseFunctionComboBox.SelectedItem = "---";

WINDOWSKeyCombinationComboBox.SelectedIndex = 0;

LaunchingTheProgramTextBox.Text = "";

LaunchingTheProgramFileWayTextBox.Text = "";

}

//

//

// Кнопка "Подтвердить выбор назначаемой кнопки: Назначить свою".

private void UserButtonButtonOK\_Click(object sender, EventArgs e)

{

UserButtonButtonCancel.Enabled = true;

UserButtonButtonOK.Enabled = false;

GroupButtonsOut.Enabled = true;

UserButtonTextBox.Enabled = false;

LazyButton.Enabled = false;

NumMultiply.Enabled = false;

NumPlus.Enabled = false;

NumMinus.Enabled = false;

DefaultTextBox1.Text = UserButtonTextBox.Text;

}

//

//

// Кнопка "Отменить выбор назначаемой кнопки: Назначить свою".

private void UserButtonButtonCancel\_Click(object sender, EventArgs e)

{

UserButtonButtonCancel.Enabled = false;

UserButtonButtonOK.Enabled = true;

GroupButtonsOut.Enabled = false;

UserButtonTextBox.Enabled = true;

LazyButton.Enabled = true;

NumMultiply.Enabled = true;

NumPlus.Enabled = true;

NumMinus.Enabled = true;

DefaultTextBox1.Text = "";

UserButtonTextBox.Text = "";

KeyboardFunctionTextBox.Text = "";

MouseFunctionComboBox.SelectedItem = "---";

WINDOWSKeyCombinationComboBox.SelectedIndex = 0;

LaunchingTheProgramTextBox.Text = "";

LaunchingTheProgramFileWayTextBox.Text = "";

}

//

//

// ТекстБокс "на какую кнопку или сочетание клавиш назначить чет. UserButton".

private void UserButtonTextBox\_KeyDown(object sender, System.Windows.Forms.KeyEventArgs e)

{

// очищаем поле

UserButtonTextBox.Text = "";

// проверяем нажата ли клавиша Ctrl: если да, то записаваем в поле слово Ctrl

if (e.Control)

{

UserButtonTextBox.Text += "Ctrl+";

}

// проверяем нажата ли клавиша Shift: если да, то записаваем в поле слово Shift

if (e.Shift)

{

UserButtonTextBox.Text += "Shift+";

}

// проверяем нажата ли клавиша Alt: если да, то записаваем в поле слово Alt

if (e.Alt)

{

UserButtonTextBox.Text += "Alt+";

}

// копируем KeyData нажатой клавиши

Keys key = e.KeyData;

// извлекаем из данных о нажатой клавише коды системных кнопок, таких как Ctrl, Shift, Alt

key &= ~Keys.Control;

key &= ~Keys.Shift;

key &= ~Keys.Alt;

// выводим полученное словосочетание

UserButtonTextBox.Text += key.ToString();

}

//

//

//!!!\*\*\* НИЖНАЯЯ ГРУППА \*\*\*!!!\\\

//

//

//

//

// GroupButtonsOut\_Enter - Группа кнопок назначения клавиш.

private void GroupButtonsOut\_Enter(object sender, EventArgs e)

{

}

//

// По умолчанию.

// Кнопка "сохранить" по умолчанию.

private void DefaultButtonSave\_Click(object sender, EventArgs e)

{

KeyboardFunctionTextBox.Text = "";

MouseFunctionComboBox.SelectedItem = "---";

WINDOWSKeyCombinationComboBox.SelectedIndex = 0;

LaunchingTheProgramTextBox.Text = "";

LaunchingTheProgramFileWayTextBox.Text = "";

GroupButtonsOut.Enabled = false;

}

//

// Функция клавиатуры.

// ТекстБокс "какую кнопку или сочетание клавиш назначить. KeyboardFunction".

private void KeyboardFunctionTextBox\_KeyDown(object sender, System.Windows.Forms.KeyEventArgs e)

{

// очищаем поле

KeyboardFunctionTextBox.Text = "";

// проверяем нажата ли клавиша Ctrl: если да, то записаваем в поле слово Ctrl

if (e.Control)

{

KeyboardFunctionTextBox.Text += "Ctrl+";

}

// проверяем нажата ли клавиша Shift: если да, то записаваем в поле слово Shift

if (e.Shift)

{

KeyboardFunctionTextBox.Text += "Shift+";

}

// проверяем нажата ли клавиша Alt: если да, то записаваем в поле слово Alt

if (e.Alt)

{

KeyboardFunctionTextBox.Text += "Alt+";

}

// копируем KeyData нажатой клавиши

Keys key = e.KeyData;

// извлекаем из данных о нажатой клавише коды системных кнопок, таких как Ctrl, Shift, Alt

key &= ~Keys.Control;

key &= ~Keys.Shift;

key &= ~Keys.Alt;

// выводим полученное словосочетание

KeyboardFunctionTextBox.Text += key.ToString();

}

//

// Функция клавиатуры.

// Кнопка "сохранить" функцию клавиатуры.

private void KeyboardFunctionButtonSave\_Click(object sender, EventArgs e)

{

DefaultTextBox1.Text = "";

MouseFunctionComboBox.SelectedItem = "---";

WINDOWSKeyCombinationComboBox.SelectedIndex = 0;

LaunchingTheProgramTextBox.Text = "";

LaunchingTheProgramFileWayTextBox.Text = "";

GroupButtonsOut.Enabled = false;

}

//

// Функция мыши.

// Кнопка "сохранить" функцию мыши.

private void MouseFunctionButtonSave\_Click(object sender, EventArgs e)

{

DefaultTextBox1.Text = "";

KeyboardFunctionTextBox.Text = "";

WINDOWSKeyCombinationComboBox.SelectedIndex = 0;

LaunchingTheProgramTextBox.Text = "";

LaunchingTheProgramFileWayTextBox.Text = "";

GroupButtonsOut.Enabled = false;

}

//

// Сочетание клавиш Windows.

// Кнопка "сохранить" сочетание клавиш Windows.

private void WINDOWSKeyCombinationButtonSave\_Click(object sender, EventArgs e)

{

DefaultTextBox1.Text = "";

KeyboardFunctionTextBox.Text = "";

MouseFunctionComboBox.SelectedItem = "---";

LaunchingTheProgramTextBox.Text = "";

LaunchingTheProgramFileWayTextBox.Text = "";

GroupButtonsOut.Enabled = false;

}

//

// Запуск программы.

// Кнопка "сохранить" запуск программы.

private void LaunchingTheProgramButtonSave\_Click(object sender, EventArgs e)

{

DefaultTextBox1.Text = "";

KeyboardFunctionTextBox.Text = "";

MouseFunctionComboBox.SelectedItem = "---";

WINDOWSKeyCombinationComboBox.SelectedIndex = 0;

GroupButtonsOut.Enabled = false;

}

//

//

// Серия циклов по:

// 1) Реализация назначения Запуска программы по нажатию клавиши.

// 2) Реализация назначения Функции мыши.

// 3) Реализация назначения Сочетания клавиш Windows.

private void Form1\_KeyDown(object sender, KeyEventArgs e)

{

Keys key = e.KeyData;

KeysConverter converter = new KeysConverter();

string text = converter.ConvertToString(e.KeyCode);

// 1) Реализация назначения Запуска программы по нажатию клавиши.

// Цикл запуска программы на нажатие Num Multiply.

if (e.KeyCode == Keys.Multiply && NumMultiplyButtonCancel.Enabled == true && GroupButtonsOut.Enabled == false) //e.KeyCode == SendKeys.Equals()

{

if (LaunchingTheProgramFileWayTextBox.Text == "")

{

}

else

{

Process note = Process.Start(LaunchingTheProgramFileWayTextBox.Text);

}

}

// Цикл запуска программы на нажатие Num Plus.

if (e.KeyCode == Keys.Add && NumPlusButtonCancel.Enabled == true && GroupButtonsOut.Enabled == false) //e.KeyCode == SendKeys.Equals()

{

if (LaunchingTheProgramFileWayTextBox.Text == "")

{

}

else

{

Process note = Process.Start(LaunchingTheProgramFileWayTextBox.Text);

}

}

// Цикл запуска программы на нажатие Num Minus.

if (e.KeyCode == Keys.Subtract && NumMinusButtonCancel.Enabled == true && GroupButtonsOut.Enabled == false) //e.KeyCode == SendKeys.Equals()

{

if (LaunchingTheProgramFileWayTextBox.Text == "")

{

}

else

{

Process note = Process.Start(LaunchingTheProgramFileWayTextBox.Text);

}

}

// Цикл запуска программы на назначенную пользователем клавишу.

if (((object)text).Equals((object)UserButtonTextBox.Text) && UserButtonButtonCancel.Enabled == true && GroupButtonsOut.Enabled == false) //e.KeyCode == SendKeys.Equals()

{

if (LaunchingTheProgramFileWayTextBox.Text == "")

{

}

else

{

Process note = Process.Start(LaunchingTheProgramFileWayTextBox.Text);

}

}

// 2) Реализация назначения Функции мыши.

// Цикл запуска Щелчка мыши на нажание Num Multiply.

if (MouseFunctionComboBox.SelectedItem == "Щелчок" && e.KeyCode == Keys.Multiply && NumMultiplyButtonCancel.Enabled == true && GroupButtonsOut.Enabled == false)

{

mouse\_event(MouseEventF\_LeftDown, 0, 0, 0, UIntPtr.Zero);

mouse\_event(MouseEventF\_LeftUp, 0, 0, 0, UIntPtr.Zero);

}

// Цикл запуска Щелчка мыши на нажание Num Plus.

if (MouseFunctionComboBox.SelectedItem == "Щелчок" && e.KeyCode == Keys.Add && NumPlusButtonCancel.Enabled == true && GroupButtonsOut.Enabled == false)

{

mouse\_event(MouseEventF\_LeftDown, 0, 0, 0, UIntPtr.Zero);

mouse\_event(MouseEventF\_LeftUp, 0, 0, 0, UIntPtr.Zero);

}

// Цикл запуска Щелчка мыши на нажание Num Minus.

if (MouseFunctionComboBox.SelectedItem == "Щелчок" && e.KeyCode == Keys.Subtract && NumMinusButtonCancel.Enabled == true && GroupButtonsOut.Enabled == false)

{

mouse\_event(MouseEventF\_LeftDown, 0, 0, 0, UIntPtr.Zero);

mouse\_event(MouseEventF\_LeftUp, 0, 0, 0, UIntPtr.Zero);

}

// Цикл запуска Щелчка мыши на назначенную пользователем клавишу.

if (MouseFunctionComboBox.SelectedItem == "Щелчок" && ((object)text).Equals((object)UserButtonTextBox.Text) && UserButtonButtonCancel.Enabled == true && GroupButtonsOut.Enabled == false)

{

mouse\_event(MouseEventF\_LeftDown, 0, 0, 0, UIntPtr.Zero);

mouse\_event(MouseEventF\_LeftUp, 0, 0, 0, UIntPtr.Zero);

}

// Цикл запуска Двойного щелчка мыши на нажание Num Multiply.

if (MouseFunctionComboBox.SelectedItem == "Двойной щелчок" && e.KeyCode == Keys.Multiply && NumMultiplyButtonCancel.Enabled == true && GroupButtonsOut.Enabled == false)

{

mouse\_event(MouseEventF\_LeftDown, 0, 0, 0, UIntPtr.Zero);

mouse\_event(MouseEventF\_LeftUp, 0, 0, 0, UIntPtr.Zero);

mouse\_event(MouseEventF\_LeftDown, 0, 0, 0, UIntPtr.Zero);

mouse\_event(MouseEventF\_LeftUp, 0, 0, 0, UIntPtr.Zero);

}

// Цикл запуска Двойного щелчка мыши на нажание Num Plus.

if (MouseFunctionComboBox.SelectedItem == "Двойной щелчок" && e.KeyCode == Keys.Add && NumPlusButtonCancel.Enabled == true && GroupButtonsOut.Enabled == false)

{

mouse\_event(MouseEventF\_LeftDown, 0, 0, 0, UIntPtr.Zero);

mouse\_event(MouseEventF\_LeftUp, 0, 0, 0, UIntPtr.Zero);

mouse\_event(MouseEventF\_LeftDown, 0, 0, 0, UIntPtr.Zero);

mouse\_event(MouseEventF\_LeftUp, 0, 0, 0, UIntPtr.Zero);

}

// Цикл запуска Двойного щелчка мыши на нажание Num Minus.

if (MouseFunctionComboBox.SelectedItem == "Двойной щелчок" && e.KeyCode == Keys.Subtract && NumMinusButtonCancel.Enabled == true && GroupButtonsOut.Enabled == false)

{

mouse\_event(MouseEventF\_LeftDown, 0, 0, 0, UIntPtr.Zero);

mouse\_event(MouseEventF\_LeftUp, 0, 0, 0, UIntPtr.Zero);

mouse\_event(MouseEventF\_LeftDown, 0, 0, 0, UIntPtr.Zero);

mouse\_event(MouseEventF\_LeftUp, 0, 0, 0, UIntPtr.Zero);

}

// Цикл запуска Двойного щелчка мыши на назначенную пользователем клавишу.

if (MouseFunctionComboBox.SelectedItem == "Двойной щелчок" && ((object)text).Equals((object)UserButtonTextBox.Text) && UserButtonButtonCancel.Enabled == true && GroupButtonsOut.Enabled == false)

{

mouse\_event(MouseEventF\_LeftDown, 0, 0, 0, UIntPtr.Zero);

mouse\_event(MouseEventF\_LeftUp, 0, 0, 0, UIntPtr.Zero);

mouse\_event(MouseEventF\_LeftDown, 0, 0, 0, UIntPtr.Zero);

mouse\_event(MouseEventF\_LeftUp, 0, 0, 0, UIntPtr.Zero);

}

// Цикл запуска Меню мыши на нажание Num Multiply.

if (MouseFunctionComboBox.SelectedItem == "Меню" && e.KeyCode == Keys.Multiply && NumMultiplyButtonCancel.Enabled == true && GroupButtonsOut.Enabled == false)

{

mouse\_event(MouseEventF\_RightDown, 0, 0, 0, UIntPtr.Zero);

mouse\_event(MouseEventF\_RightUp, 0, 0, 0, UIntPtr.Zero);

mouse\_event(MouseEventF\_RightDown, 0, 0, 0, UIntPtr.Zero);

mouse\_event(MouseEventF\_RightUp, 0, 0, 0, UIntPtr.Zero);

}

// Цикл запуска Меню мыши на нажание Num Plus.

if (MouseFunctionComboBox.SelectedItem == "Меню" && e.KeyCode == Keys.Add && NumPlusButtonCancel.Enabled == true && GroupButtonsOut.Enabled == false)

{

mouse\_event(MouseEventF\_RightDown, 0, 0, 0, UIntPtr.Zero);

mouse\_event(MouseEventF\_RightUp, 0, 0, 0, UIntPtr.Zero);

mouse\_event(MouseEventF\_RightDown, 0, 0, 0, UIntPtr.Zero);

mouse\_event(MouseEventF\_RightUp, 0, 0, 0, UIntPtr.Zero);

}

// Цикл запуска Меню мыши на нажание Num Minus.

if (MouseFunctionComboBox.SelectedItem == "Меню" && e.KeyCode == Keys.Subtract && NumMinusButtonCancel.Enabled == true && GroupButtonsOut.Enabled == false)

{

mouse\_event(MouseEventF\_RightDown, 0, 0, 0, UIntPtr.Zero);

mouse\_event(MouseEventF\_RightUp, 0, 0, 0, UIntPtr.Zero);

mouse\_event(MouseEventF\_RightDown, 0, 0, 0, UIntPtr.Zero);

mouse\_event(MouseEventF\_RightUp, 0, 0, 0, UIntPtr.Zero);

}

// Цикл запуска Меню мыши на назначенную пользователем клавишу.

if (MouseFunctionComboBox.SelectedItem == "Меню" && ((object)text).Equals((object)UserButtonTextBox.Text) && UserButtonButtonCancel.Enabled == true && GroupButtonsOut.Enabled == false)

{

mouse\_event(MouseEventF\_RightDown, 0, 0, 0, UIntPtr.Zero);

mouse\_event(MouseEventF\_RightUp, 0, 0, 0, UIntPtr.Zero);

mouse\_event(MouseEventF\_RightDown, 0, 0, 0, UIntPtr.Zero);

mouse\_event(MouseEventF\_RightUp, 0, 0, 0, UIntPtr.Zero);

}

// 3) Реализация назначения Сочетания клавиш Windows.

// Цикл запуска Сочетания клавиш Windows: "LockWorkStation" на нажатие Num Multiply.

if (WINDOWSKeyCombinationComboBox.SelectedItem == "LockWorkStation" && e.KeyCode == Keys.Multiply && NumMultiplyButtonCancel.Enabled == true && GroupButtonsOut.Enabled == false) //e.KeyCode == SendKeys.Equals()

{

LockWorkStation();

}

// Цикл запуска Сочетания клавиш Windows: "LockWorkStation" на нажатие Num Plus.

if (WINDOWSKeyCombinationComboBox.SelectedItem == "LockWorkStation" && e.KeyCode == Keys.Add && NumPlusButtonCancel.Enabled == true && GroupButtonsOut.Enabled == false) //e.KeyCode == SendKeys.Equals()

{

LockWorkStation();

}

// Цикл запуска Сочетания клавиш Windows: "LockWorkStation" на нажатие Num Minus.

if (WINDOWSKeyCombinationComboBox.SelectedItem == "LockWorkStation" && e.KeyCode == Keys.Subtract && NumMinusButtonCancel.Enabled == true && GroupButtonsOut.Enabled == false) //e.KeyCode == SendKeys.Equals()

{

LockWorkStation();

}

// Цикл запуска Сочетания клавиш Windows: "LockWorkStation" на назначенную пользователем клавишу.

if (WINDOWSKeyCombinationComboBox.SelectedItem == "LockWorkStation" && ((object)text).Equals((object)UserButtonTextBox.Text) && UserButtonButtonCancel.Enabled == true && GroupButtonsOut.Enabled == false) //e.KeyCode == SendKeys.Equals()

{

LockWorkStation();

}

// Цикл запуска Сочетания клавиш Windows: "Свернуть все окна" на нажатие Num Multiply.

Shell32.Shell shell = new Shell32.Shell();

if (WINDOWSKeyCombinationComboBox.SelectedItem == "Свернуть все окна" && e.KeyCode == Keys.Multiply && NumMultiplyButtonCancel.Enabled == true && GroupButtonsOut.Enabled == false) //e.KeyCode == SendKeys.Equals()

{

shell.MinimizeAll();

}

// Цикл запуска Сочетания клавиш Windows: "Свернуть все окна" на нажатие Num Plus.

if (WINDOWSKeyCombinationComboBox.SelectedItem == "Свернуть все окна" && e.KeyCode == Keys.Add && NumPlusButtonCancel.Enabled == true && GroupButtonsOut.Enabled == false) //e.KeyCode == SendKeys.Equals()

{

shell.MinimizeAll();

}

// Цикл запуска Сочетания клавиш Windows: "Свернуть все окна" на нажатие Num Minus.

if (WINDOWSKeyCombinationComboBox.SelectedItem == "Свернуть все окна" && e.KeyCode == Keys.Subtract && NumMinusButtonCancel.Enabled == true && GroupButtonsOut.Enabled == false) //e.KeyCode == SendKeys.Equals()

{

shell.MinimizeAll();

}

// Цикл запуска Сочетания клавиш Windows: "Свернуть все окна" на назначенную пользователем клавишу.

if (WINDOWSKeyCombinationComboBox.SelectedItem == "Свернуть все окна" && ((object)text).Equals((object)UserButtonTextBox.Text) && UserButtonButtonCancel.Enabled == true && GroupButtonsOut.Enabled == false) //e.KeyCode == SendKeys.Equals()

{

shell.MinimizeAll();

}

// Цикл запуска Сочетания клавиш Windows: "Alt + TAB" на нажатие Num Multiply.

if (WINDOWSKeyCombinationComboBox.SelectedItem == "Alt + TAB" && e.KeyCode == Keys.Multiply && NumMultiplyButtonCancel.Enabled == true && GroupButtonsOut.Enabled == false) //e.KeyCode == SendKeys.Equals()

{

SendKeys.Send("%{TAB}");

}

// Цикл запуска Сочетания клавиш Windows: "Alt + TAB" на нажатие Num Plus.

if (WINDOWSKeyCombinationComboBox.SelectedItem == "Alt + TAB" && e.KeyCode == Keys.Add && NumPlusButtonCancel.Enabled == true && GroupButtonsOut.Enabled == false) //e.KeyCode == SendKeys.Equals()

{

SendKeys.Send("%{TAB}");

}

// Цикл запуска Сочетания клавиш Windows: "Alt + TAB" на нажатие Num Minus.

if (WINDOWSKeyCombinationComboBox.SelectedItem == "Alt + TAB" && e.KeyCode == Keys.Subtract && NumMinusButtonCancel.Enabled == true && GroupButtonsOut.Enabled == false) //e.KeyCode == SendKeys.Equals()

{

SendKeys.Send("%{TAB}");

}

// Цикл запуска Сочетания клавиш Windows: "Alt + TAB" на назначенную пользователем клавишу.

if (WINDOWSKeyCombinationComboBox.SelectedItem == "Alt + TAB" && ((object)text).Equals((object)UserButtonTextBox.Text) && UserButtonButtonCancel.Enabled == true && GroupButtonsOut.Enabled == false) //e.KeyCode == SendKeys.Equals()

{

SendKeys.Send("%{TAB}");

}

/\*

// Доделать это. после разбора с глобальной работой приложения. ибо закрывать свое же приложение так, когда есть "крестик" - бесмыссленно.

// Из коллекции комбоБокса "Закрыть приложение ( Alt +F4)" - удален до обновлений.

// Аналогично для "Вырезать", "Копировать", "Вставить", "Альт + Принскрин".

// Цикл запуска Сочетания клавиш Windows: "Закрыть приложение ( Alt +F4)" на нажание Num Multiply.

if (WINDOWSKeyCombinationComboBox.SelectedItem == "Закрыть приложение (Alt +F4)" && e.KeyCode == Keys.Multiply && NumMultiplyButtonCancel.Enabled == true && GroupButtonsOut.Enabled == false) //e.KeyCode == SendKeys.Equals()

{

SendKeys.Send("%{F4}");

}

// Цикл запуска Сочетания клавиш Windows: "Закрыть приложение ( Alt +F4)" на нажатие Num Plus.

if (WINDOWSKeyCombinationComboBox.SelectedItem == "Закрыть приложение (Alt +F4)" && e.KeyCode == Keys.Add && NumPlusButtonCancel.Enabled == true && GroupButtonsOut.Enabled == false) //e.KeyCode == SendKeys.Equals()

{

SendKeys.Send("%{F4}");

}

// Цикл запуска Сочетания клавиш Windows: "Закрыть приложение ( Alt +F4)" на нажатие Num Minus.

if (WINDOWSKeyCombinationComboBox.SelectedItem == "Закрыть приложение (Alt +F4)" && e.KeyCode == Keys.Subtract && NumMinusButtonCancel.Enabled == true && GroupButtonsOut.Enabled == false) //e.KeyCode == SendKeys.Equals()

{

SendKeys.Send("%{F4}");

}

// Цикл запуска Сочетания клавиш Windows: "Закрыть приложение ( Alt +F4)" на назначенную пользователем клавишу.

if (WINDOWSKeyCombinationComboBox.SelectedItem == "Закрыть приложение (Alt +F4)" && ((object)text).Equals((object)UserButtonTextBox.Text) && UserButtonButtonCancel.Enabled == true && GroupButtonsOut.Enabled == false) //e.KeyCode == SendKeys.Equals()

{

SendKeys.Send("%{F4}");

}

//альт+принскрин. без % - просто принскрин

//SendKeys.Send("%{PRTSC}");

\*/

}

//

//

// Кнопка в "Запуск программы" для указания файла, который нужно открыть.

private void LaunchingTheProgramButtonOpen\_Click(object sender, EventArgs e)

{

OpenFileDialog ofd = new OpenFileDialog();

if (ofd.ShowDialog() == DialogResult.OK)

{

LaunchingTheProgramTextBox.Text = ofd.SafeFileName;

LaunchingTheProgramFileWayTextBox.Text = ofd.FileName;

}

}

//

//

// Это типа кнопка на ардуино

private void TestButtonLazyButton\_Click(object sender, EventArgs e)

{

// 1) Реализация назначения Запуска программы по нажатию Кнопки.

// Цикл запуска программы на нажатие TestButtonLazyButton.

if (LazyButtonButtonCancel.Enabled == true && GroupButtonsOut.Enabled == false) //e.KeyCode == SendKeys.Equals()

{

if (LaunchingTheProgramFileWayTextBox.Text == "")

{

}

else

{

Process note = Process.Start(LaunchingTheProgramFileWayTextBox.Text);

}

}

// 2) Реализация назначения Функции мыши.

// Цикл запуска Щелчка мыши на нажатие TestButtonLazyButton.

if (MouseFunctionComboBox.SelectedItem == "Щелчок" && LazyButtonButtonCancel.Enabled == true && GroupButtonsOut.Enabled == false)

{

mouse\_event(MouseEventF\_LeftDown, 0, 0, 0, UIntPtr.Zero);

mouse\_event(MouseEventF\_LeftUp, 0, 0, 0, UIntPtr.Zero);

}

// Цикл запуска Двойного щелчка мыши на нажатие TestButtonLazyButton.

if (MouseFunctionComboBox.SelectedItem == "Двойной щелчок" && LazyButtonButtonCancel.Enabled == true && GroupButtonsOut.Enabled == false)

{

mouse\_event(MouseEventF\_LeftDown, 0, 0, 0, UIntPtr.Zero);

mouse\_event(MouseEventF\_LeftUp, 0, 0, 0, UIntPtr.Zero);

mouse\_event(MouseEventF\_LeftDown, 0, 0, 0, UIntPtr.Zero);

mouse\_event(MouseEventF\_LeftUp, 0, 0, 0, UIntPtr.Zero);

}

// Цикл запуска Меню мыши на нажатие TestButtonLazyButton.

if (MouseFunctionComboBox.SelectedItem == "Меню" && LazyButtonButtonCancel.Enabled == true && GroupButtonsOut.Enabled == false)

{

mouse\_event(MouseEventF\_RightDown, 0, 0, 0, UIntPtr.Zero);

mouse\_event(MouseEventF\_RightUp, 0, 0, 0, UIntPtr.Zero);

mouse\_event(MouseEventF\_RightDown, 0, 0, 0, UIntPtr.Zero);

mouse\_event(MouseEventF\_RightUp, 0, 0, 0, UIntPtr.Zero);

}

// 3) Реализация назначения Сочетания клавиш Windows.

// Цикл запуска Сочетания клавиш Windows: "LockWorkStation" на нажатие TestButtonLazyButton.

if (WINDOWSKeyCombinationComboBox.SelectedItem == "LockWorkStation" && LazyButtonButtonCancel.Enabled == true && GroupButtonsOut.Enabled == false) //e.KeyCode == SendKeys.Equals()

{

LockWorkStation();

}

// Цикл запуска Сочетания клавиш Windows: "Свернуть все окна" на нажатие TestButtonLazyButton.

Shell32.Shell shell = new Shell32.Shell();

if (WINDOWSKeyCombinationComboBox.SelectedItem == "Свернуть все окна" && LazyButtonButtonCancel.Enabled == true && GroupButtonsOut.Enabled == false) //e.KeyCode == SendKeys.Equals()

{

shell.MinimizeAll();

}

// Цикл запуска Сочетания клавиш Windows: "Alt + TAB" на нажатие TestButtonLazyButton.

if (WINDOWSKeyCombinationComboBox.SelectedItem == "Alt + TAB" && LazyButtonButtonCancel.Enabled == true && GroupButtonsOut.Enabled == false) //e.KeyCode == SendKeys.Equals()

{

SendKeys.Send("%{TAB}");

}

}

}

}