**Федеральное агентство связи ордена Трудового Красного Знамени федеральное государственное бюджетное общеобразовательное учреждение высшего образования «Московский технический университет связи и информатики»**

Практическая работа

Разработка комплекса антивирусной защиты

по дисциплине:

«Защита информации от вредоносного программного обеспечения»

Выполнил

Студент группы БВТ1801

Задоркин М.А.

Проверил:

Барков В.В.

Москва 2021

Оглавление

[Введение 6](#_Toc67690464)

[Цель работы: 6](#_Toc67690465)

[Средства разработки: 6](#_Toc67690466)

[Устройство базы данных 7](#_Toc67690467)

[Database 10](#_Toc67690468)

[ public List<VirusDS> getViruses(signature, position); 10](#_Toc67690469)

[ public List<PlanDS> getAllPlans(); 10](#_Toc67690470)

[ public bool addPlan(PlanDS plan); 10](#_Toc67690471)

[ public void removePlan(PlanDS plan); 10](#_Toc67690472)

[ public void addToQuarantine(string path); 10](#_Toc67690473)

[ public void removeFromQuarantine(string path); 10](#_Toc67690474)

[ public List<string> getQuarantineFiles(); 10](#_Toc67690475)

[ public void addToFoundViruses(string path); 10](#_Toc67690476)

[ public void removeFromFoundViruses(string path); 10](#_Toc67690477)

[ public List<string> getVirusesFiles(); 10](#_Toc67690478)

[Monitoring 11](#_Toc67690479)

[ alreadyRun 11](#_Toc67690480)

[ watcher 11](#_Toc67690481)

[ dangerFiles 11](#_Toc67690482)

[ public bool start(string path); 11](#_Toc67690483)

[ public void stop(); 11](#_Toc67690484)

[ public bool getStatus(); 11](#_Toc67690485)

[ public string result(); 11](#_Toc67690486)

[ private void OnChanged(); 11](#_Toc67690487)

[Scanner 11](#_Toc67690488)

[ scanning 11](#_Toc67690489)

[ pathName 11](#_Toc67690490)

[ isFile 11](#_Toc67690491)

[ entries 11](#_Toc67690492)

[ filesForScan 11](#_Toc67690493)

[ public bool start(string path); 11](#_Toc67690494)

[ public void stop(); 11](#_Toc67690495)

[ private void folderScanner(string folderPath); 11](#_Toc67690496)

[ private void filesScanner(); 11](#_Toc67690497)

[ public string getScanResult(); 11](#_Toc67690498)

[ScheduledInspections 12](#_Toc67690499)

[ timer 12](#_Toc67690500)

[ public void startWatching(); 12](#_Toc67690501)

[ private void checkScheduledInspections(); 12](#_Toc67690502)

[ private void startInspection(PlanDS plan); 12](#_Toc67690503)

[DangersDetection 12](#_Toc67690504)

[ SupportedExtensions 12](#_Toc67690505)

[ public bool detectDanger(string filePath); 12](#_Toc67690506)

[ private bool isExecutableOrZip(string path, Stream file); 12](#_Toc67690507)

[ private FileStream getFileStream(string path); 12](#_Toc67690508)

[ private bool verifyFileStream(Stream stream); 12](#_Toc67690509)

[ private byte[] getFileCode(Stream stream, ref int offset); 12](#_Toc67690510)

[ private string getStringOfBytes(byte[] array, int index, int length); 12](#_Toc67690511)

[ private bool signatureOnFullMatch(VirusDS virus, byte[] data, int offset); 12](#_Toc67690512)

[FilesWorker 13](#_Toc67690513)

[ public bool addFileToQuarantine(string filePath); 13](#_Toc67690514)

[ public bool removeFileFromQuarantine(string filePath); 13](#_Toc67690515)

[ public bool deleteFile(string filePath); 13](#_Toc67690516)

[FileDS 13](#_Toc67690517)

[ path 13](#_Toc67690518)

[ danger 13](#_Toc67690519)

[ isChecked 13](#_Toc67690520)

[ FilesHandler 13](#_Toc67690521)

[ fileHandler 13](#_Toc67690522)

[PlanDS 13](#_Toc67690523)

[ path 13](#_Toc67690524)

[ id 13](#_Toc67690525)

[ time 13](#_Toc67690526)

[ public string getTimeStringFormat(DateTime time); 13](#_Toc67690527)

[ public DateTime getTimeFromStringFormat(string stringTime); 13](#_Toc67690528)

[VirusDS 14](#_Toc67690529)

[ Id 14](#_Toc67690530)

[ name 14](#_Toc67690531)

[ signature 14](#_Toc67690532)

[ offsetBegin 14](#_Toc67690533)

[ offsetEnd 14](#_Toc67690534)

[API для взаимодействия с клиентом 15](#_Toc67690535)

[ void startScanner(string path); 15](#_Toc67690536)

[ void stopScanner(); 15](#_Toc67690537)

[ string getScanResult(); 15](#_Toc67690538)

[ bool getScanStatus(); 15](#_Toc67690539)

[ void startMonitoring(string path); 15](#_Toc67690540)

[ void stopMonitoring(); 15](#_Toc67690541)

[ string logMonitoring(); 15](#_Toc67690542)

[ bool getMonitoringStatus(); 15](#_Toc67690543)

[ handlerFiles(List<FileDS> files); 15](#_Toc67690544)

[ bool addPlan(PlanDS plan); 15](#_Toc67690545)

[ void removePlan(PlanDS plan); 15](#_Toc67690546)

[ List<PlanDS> getAllPlans(); 15](#_Toc67690547)

[ List<string> getVirusesFiles(); 15](#_Toc67690548)

[ List<string> getQuarantineList(); 15](#_Toc67690549)

[Клиент 16](#_Toc67690550)

[MainWindow 16](#_Toc67690551)

[Scanner 17](#_Toc67690552)

[Monitoring 17](#_Toc67690553)

[Plans 18](#_Toc67690554)

[Plan 18](#_Toc67690555)

[Quarantine 18](#_Toc67690556)

[QuarantineItem 19](#_Toc67690557)

[Viruses 19](#_Toc67690558)

[FilesProcessing 20](#_Toc67690559)

[FileProcessing 20](#_Toc67690560)

[Тестирование разработанного комплекса 21](#_Toc67690561)

[Вывод: 24](#_Toc67690562)

Введение

В настоящее время неотъемлемой частью любого компьютера является комплекс антивирусной защиты, который обеспечивает безопасность работы за компьютером. В данной практической работе будет разработан антивирус, реализовывающий базовые функции по защите компьютера.

Цель работы: разработать комплекс антивирусной защиты.

Средства разработки:

Для разработки клиента и службы был выбран язык c#.

Для разработки клиента использованы технологии Windows Presentation Foundation и xaml.

Для реализации коммуникации между приложениями использована технология Windows Communication Foundation.

Материалы проекта

Код решения и информация по установке (Readme.md) находится по ссылке: <https://github.com/maximzadorkin/antivirus>

Демонстрация проекта: <https://drive.google.com/file/d/1jTBaCgxAV57MpTnsdpyqhWeArrBWnavh/view?usp=sharing>

Устройство базы данных

Для базы было создано 4 таблицы:

* viruses – содержит опасные сигнатуры;
* found\_viruses – содержит пути до файлов, которые были помечены как опасные при запланированной проверке;
* plans – содержит дату и путь для запланированной проверки;
* quarantine – содержит пути до файлов, которые находятся в карантине.

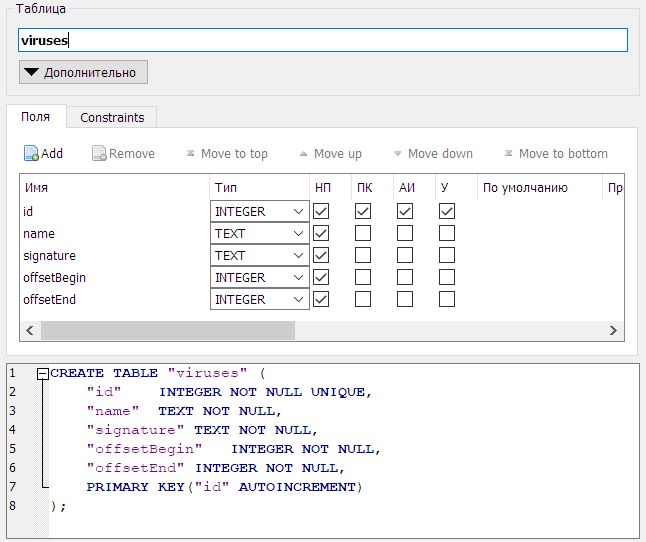


Рисунок 1 – таблица viruses

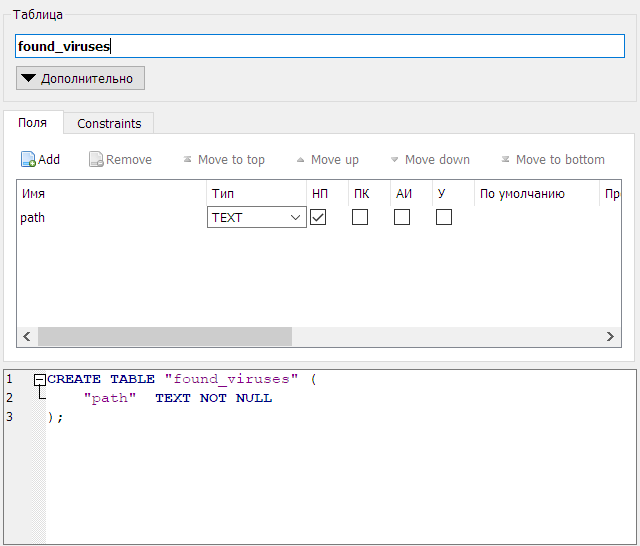


Рисунок 2 – таблица found\_viruses

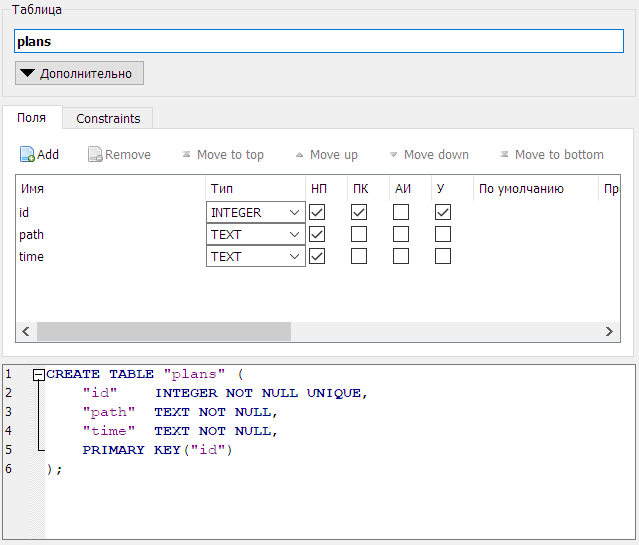


Рисунок 3 – таблица plans

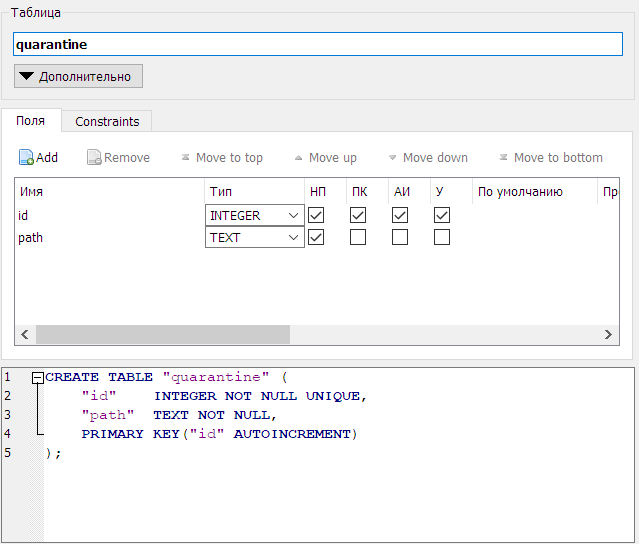


Рисунок 4 – таблица quarantine

Для разработки службы был реализован проект ServiceDll.

Основные компоненты ServiceDll:

* Database – обеспечивает взаимодействие с базой данных
* Monitoring – выполняет функции мониторинга
* Scanner – выполняет функции сканирования
* ScheduledInspections – реализует плановые проверки
* DangersDetection – обеспечивает обнаружение вирусов в файле
* FilesWorker – реализует функции работы над файлами
* FileDS – устанавливает свойства файлов
* PlanDS – устанавливает свойства планов
* VirusDS – устанавливает свойства файлов-вирусов

Ознакомимся более детально с данными классами.

Database

Данный класс обеспечивает прослойку между программой и базой данных.

Класс хранит путь до базы данных (поле source) и обеспечивает соединение (поле connection).

Имеет ряд методов:

* public List<VirusDS> getViruses(signature, position); - Осуществляет поиск в базе данных описание уязвимостей, подходящих под указанную сигнатуру, и возвращает список найденных;
* public List<PlanDS> getAllPlans(); - Осуществляет запрос всех запланированных проверок к базе данных;
* public bool addPlan(PlanDS plan); – Осуществляет добавление в базу данных новой запланированной проверки, если ее еще нет в базе;
* public void removePlan(PlanDS plan); - Осуществляет удаление из базы данных указанной запланированной проверки;
* public void addToQuarantine(string path); – Добавляет в базу данных путь до файла, который был добавлен в карантин;
* public void removeFromQuarantine(string path); - Удаляет из базы путь до файла, который ранее был в карантине;
* public List<string> getQuarantineFiles(); - Возвращает пути до всех файлов, которые сейчас находятся в карантине;
* public void addToFoundViruses(string path); - Добавляет в базу путь до файла, которые был помечен как опасный при запланированной проверке;
* public void removeFromFoundViruses(string path); - Удаляет из базы путь до файла, который ранее был помечен как опасный при запланированной проверке;
* public List<string> getVirusesFiles(); - Возвращает список всех путей до файлов, которые были помечены как опасные при запланированной проверке.

Monitoring

Данный класс обеспечивает мониторинг директории.

Содержит ряд свойств:

* alreadyRun – служит для проверки статуса работы мониторинга;
* watcher – служит для обнаружения новых файлов;
* dangerFiles – содержит список путей до файлов, которые помечены как опасные.

Имеет методы:

* public bool start(string path); – запускает мониторинг заданной директории;
* public void stop(); – останавливает мониторинг директории;
* public bool getStatus(); – возвращает статус работы мониторинга;
* public string result(); – возвращает результат работы мониторинга;
* private void OnChanged(); – обрабатывает события мониторинга и обеспечивает сканирование файлов.

Scanner

Класс обеспечивает сканирование заданного пути.

* scanning – устанавливает статус работы сканера;
* pathName – устанавливает сканируемый путь;
* isFile – определяет заданный путь является файлом или директорией;
* entries – определяет все возможные пути в заданной директории;
* filesForScan – определяет все пути, которые необходимо просканировать;
* public bool start(string path); – интерфейс для запуска сканирования;
* public void stop(); – интерфейс для остановки сканирования;
* private void folderScanner(string folderPath); – осуществляет поиск всех путей для сканирования в заданной директорией и поддиректориях;
* private void filesScanner(); – обеспечивает сканирование всех необходимых файлов;
* public string getScanResult(); – предоставляет информацию о текущем состоянии сканирования.

ScheduledInspections

Данный класс реализовывает запланированные проверки.

* timer – объект, реализующий интервальную проверку;
* public void startWatching(); – интерфейс для запуска интервальной проверки;
* private void checkScheduledInspections(); – интервальная проверка планов. Если обнаружен план, то запускается проверка;
* private void startInspection(PlanDS plan); – запускает проверку сканером плана и удаляет план из базы. После проверки записывает найденные опасные файлы в базу.

DangersDetection

Реализует сканирование файла для обнаружения уязвимостей.

* SupportedExtensions – содержит типы поддерживаемых разрешений;
* public bool detectDanger(string filePath); – предоставляет интерфейс для взаимодействия другими классами и возвращает результат работы;
* private bool isExecutableOrZip(string path, Stream file); – проверяет является файл запускаемым или архивом;
* private FileStream getFileStream(string path); – возвращает stream файла;
* private bool verifyFileStream(Stream stream); – обеспечивает проверку stream у файла;
* private byte[] getFileCode(Stream stream, ref int offset); – возвращает массив байтов stream`а;
* private string getStringOfBytes(byte[] array, int index, int length); – возвращает строковое представление массива байтов;
* private bool signatureOnFullMatch(VirusDS virus, byte[] data, int offset); – проверяет сигнатуры на совпадение.

FilesWorker

Класс обеспечивает работу над файлами.

Для этого реализованы ряд методов:

* public bool addFileToQuarantine(string filePath); – реализует добавление файла в карантин;
* public bool removeFileFromQuarantine(string filePath); – реализует удаление файла из карантина;
* public bool deleteFile(string filePath); – реализует удаление файла.

FileDS

Класс служит описанием файлов.

* path – содержит путь до файла;
* danger – устанавливает является ли файл опасным;
* isChecked – устанавливает является ли файл проверенным системой;
* FilesHandler – содержит описание возможных действий с файлом;
  + **Allow** – не совершать действий над файлом;
  + **ToQuarantine** – добавить файл в карантин;
  + **RemoveFromQuarantine** – удалить файл из карантина;
  + **Delete** – удалить файл.
* fileHandler – содержит указания, что необходимо сделать с файлом.

PlanDS

Класс описывает плановые проверки.

* path – директория для плановой проверки;
* id – уникальный идентификатор;
* time –время проверки.
* public string getTimeStringFormat(DateTime time); – возвращает строковый формат даты и времени проверки;
* public DateTime getTimeFromStringFormat(string stringTime); – возвращает формат DateTime для строкового формата даты и времени проверки.

VirusDS

Класс описывает опасные сигнатуры.

Содержит следующие описания:

* Id – уникальный идентификатор;
* name – название уязвимости;
* signature – сигнатура;
* offsetBegin – начало смещения;
* offsetEnd – конец смещения;

API для взаимодействия с клиентом

* void startScanner(string path); – запускает сканирования пути;
* void stopScanner(); – останавливает сканирование пути;
* string getScanResult(); – получает результаты или логи сканирования;
* bool getScanStatus(); – получает статус работы сканера;
* void startMonitoring(string path); – запускает мониторинг директории;
* void stopMonitoring(); – останавливает мониторинг директории;
* string logMonitoring(); – получает логи мониторинга;
* bool getMonitoringStatus(); – получает статус мониторинга;
* handlerFiles(List<FileDS> files); – обрабатывает переданные файлы с учетом их fileHandler;
* bool addPlan(PlanDS plan); – добавляет новую запланированную проверку;
* void removePlan(PlanDS plan); – удаляет запланированную проверку;
* List<PlanDS> getAllPlans(); – получает все запланированные проверки;
* List<string> getVirusesFiles(); – получает все пути до файлов, выделенные как опасные;
* List<string> getQuarantineList(); – получает все пути до файлов, находящихся в карантине;

Клиент

Для клиента были реализованы следующие компоненты:

MainWindow – служит для переключения между компонентами программы.

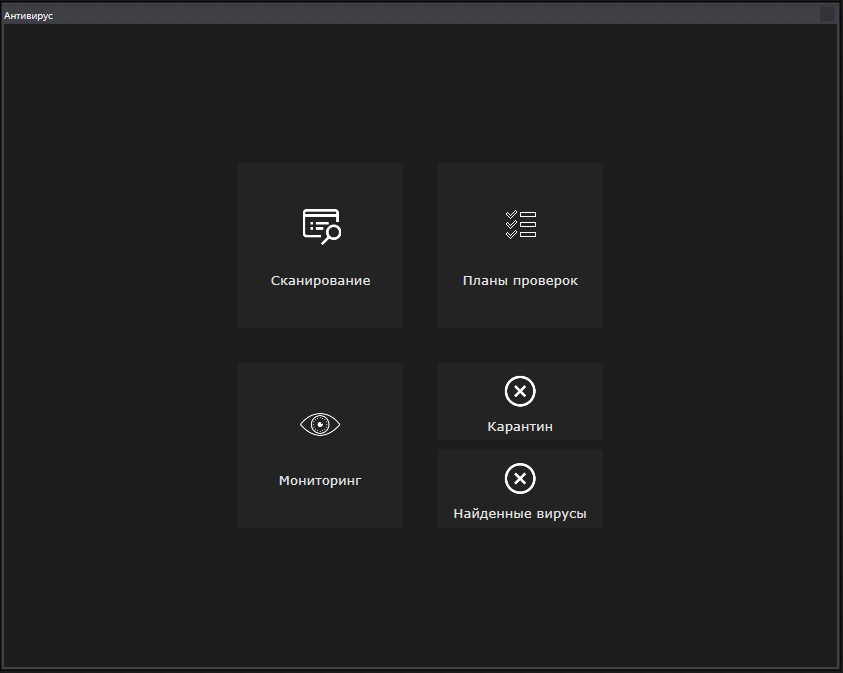


Рисунок 5 – MainWindow control

Scanner – обеспечивает управление сканированием.

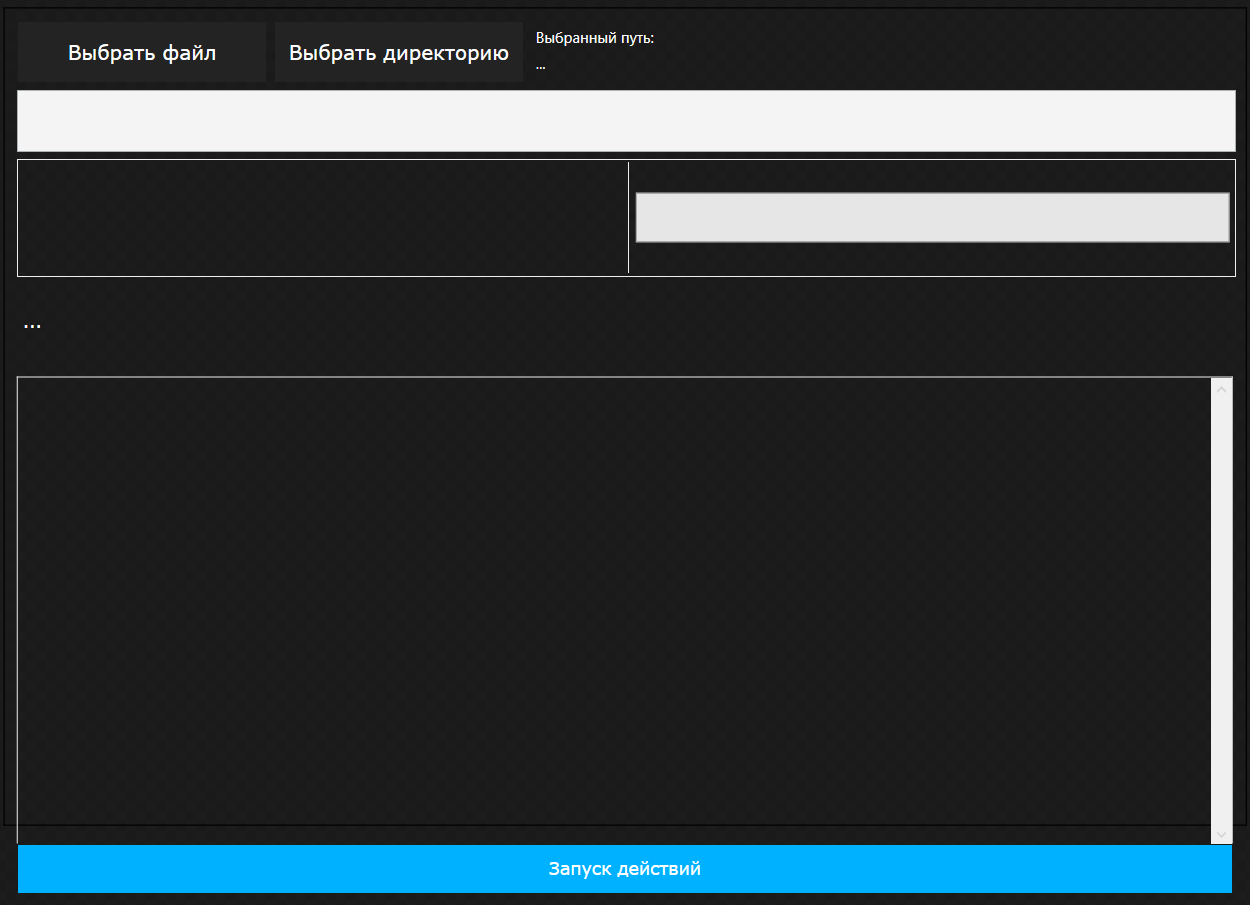


Рисунок 6 – Scanner control

Monitoring – обеспечивает управление мониторингом.

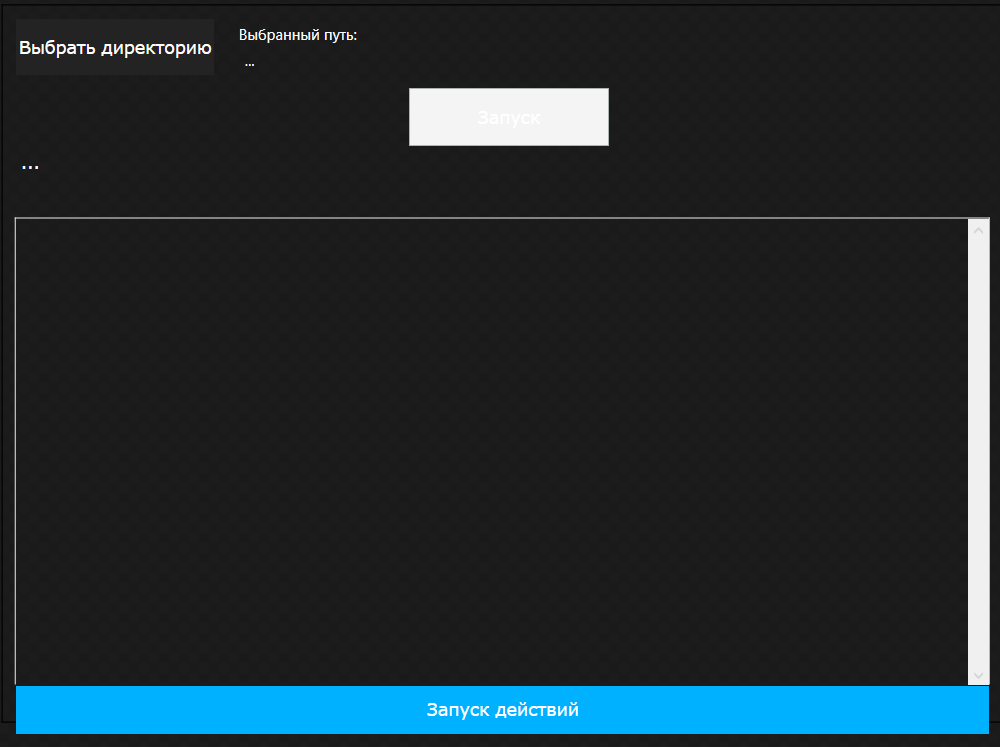


Рисунок 7 – Monitoring control

Plans – обеспечивает управление запланированными проверками и их добавление.

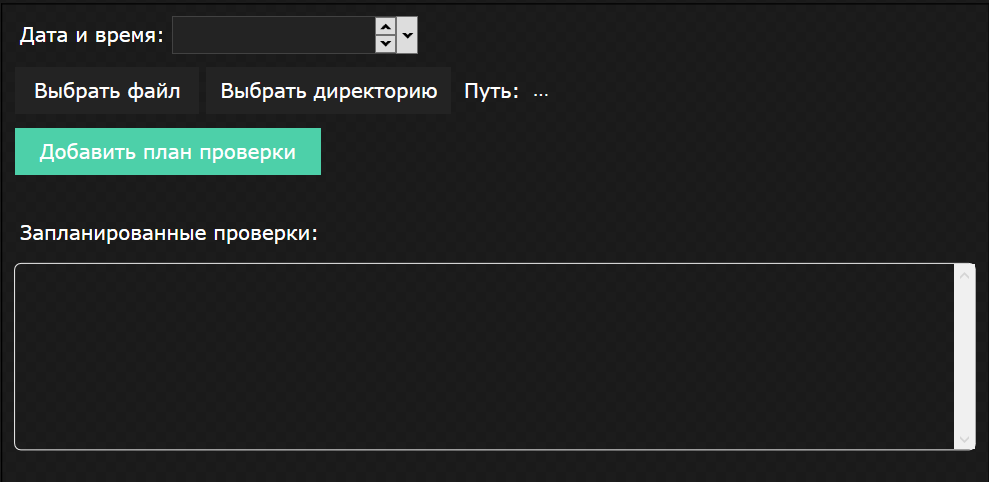


Рисунок 8 – Plans control

Plan – служит для вывода запланированной проверки и позволяет удалить ее.



Рисунок 9 – Plan control

Quarantine – выводит список путей до файлов, находящихся в карантине.

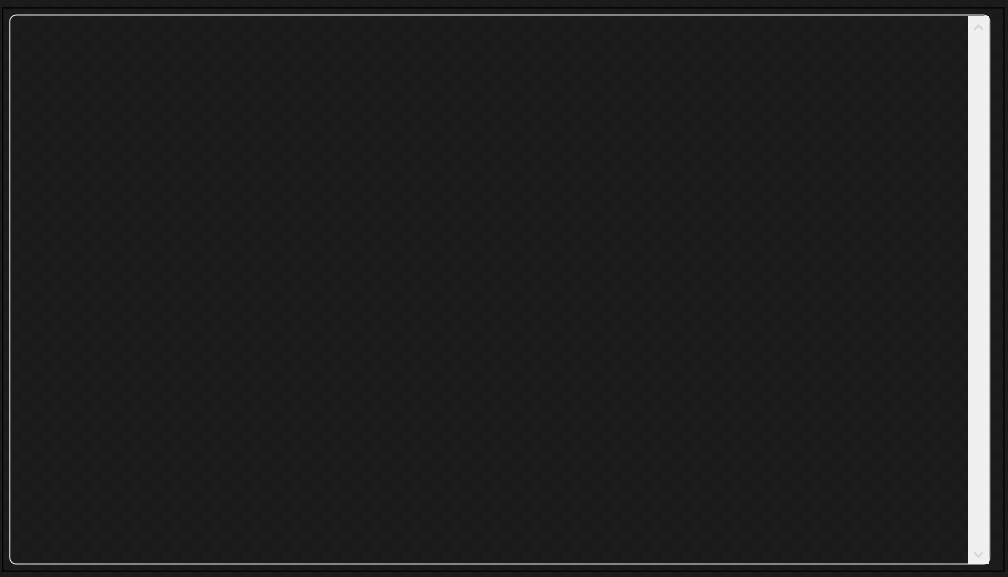


Рисунок 10 – Quarantine control

QuarantineItem – выводит путь до файла, находящегося в карантине и позволяет удалить файл из карантина.



Рисунок 11 – QuarantineItem control

Viruses – выводит список путей опасных файлов, найденных при запланированных проверках.

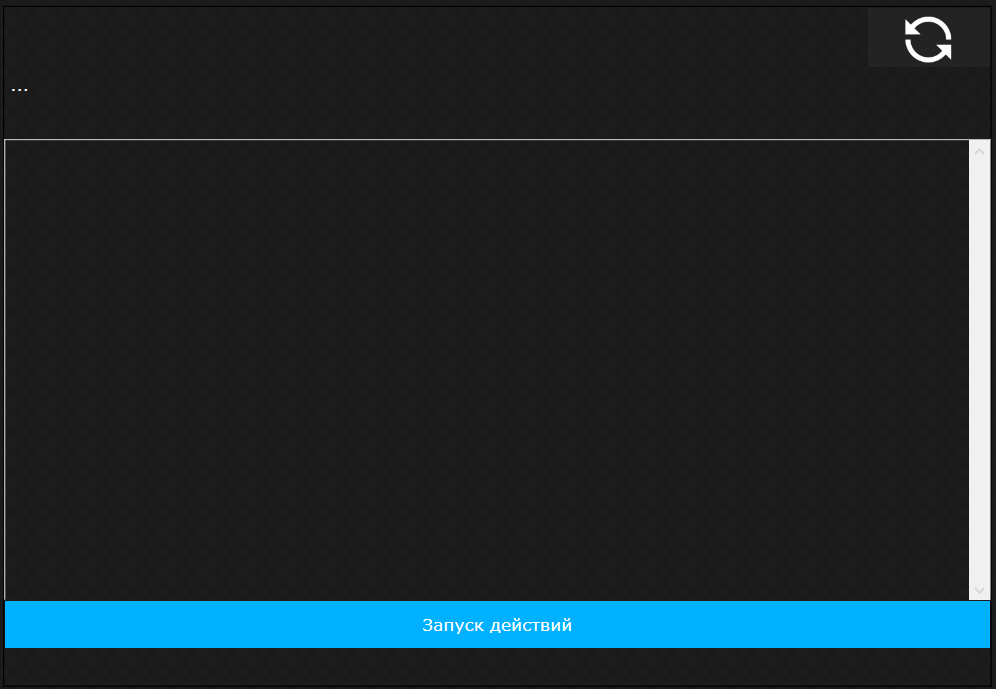


Рисунок 12 – Viruses control

FilesProcessing – выводит список файлов для реализации с ними действий и запускает обработку.

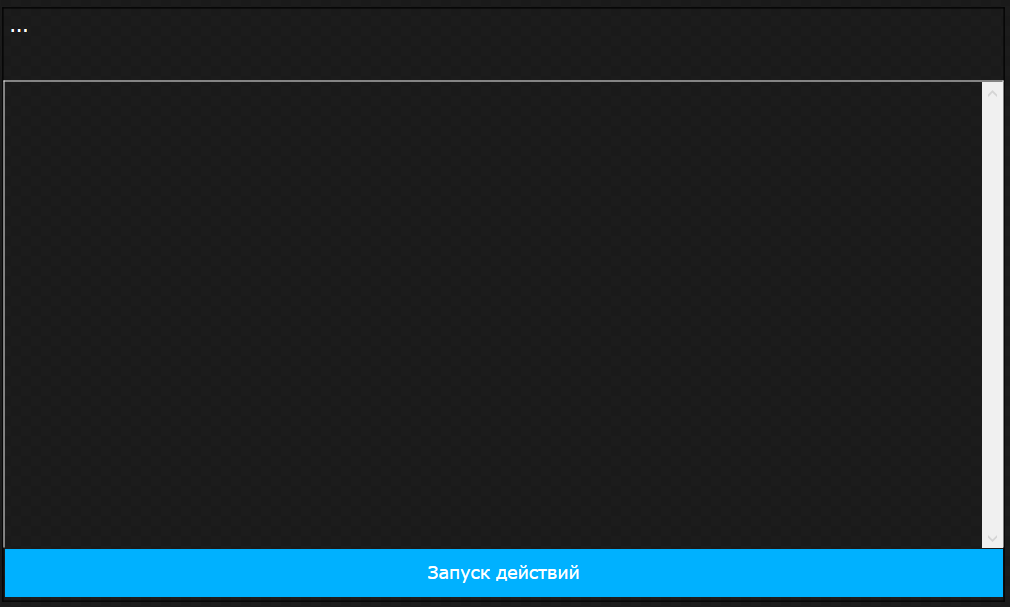


Рисунок 13 – FilesProcessing control

FileProcessing – позволяет указать, что необходимо сделать с файлом и выводит путь файла.

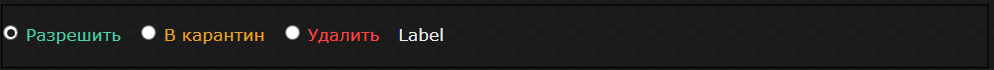


Рисунок 14 – FileProcessing control

Тестирование разработанного комплекса

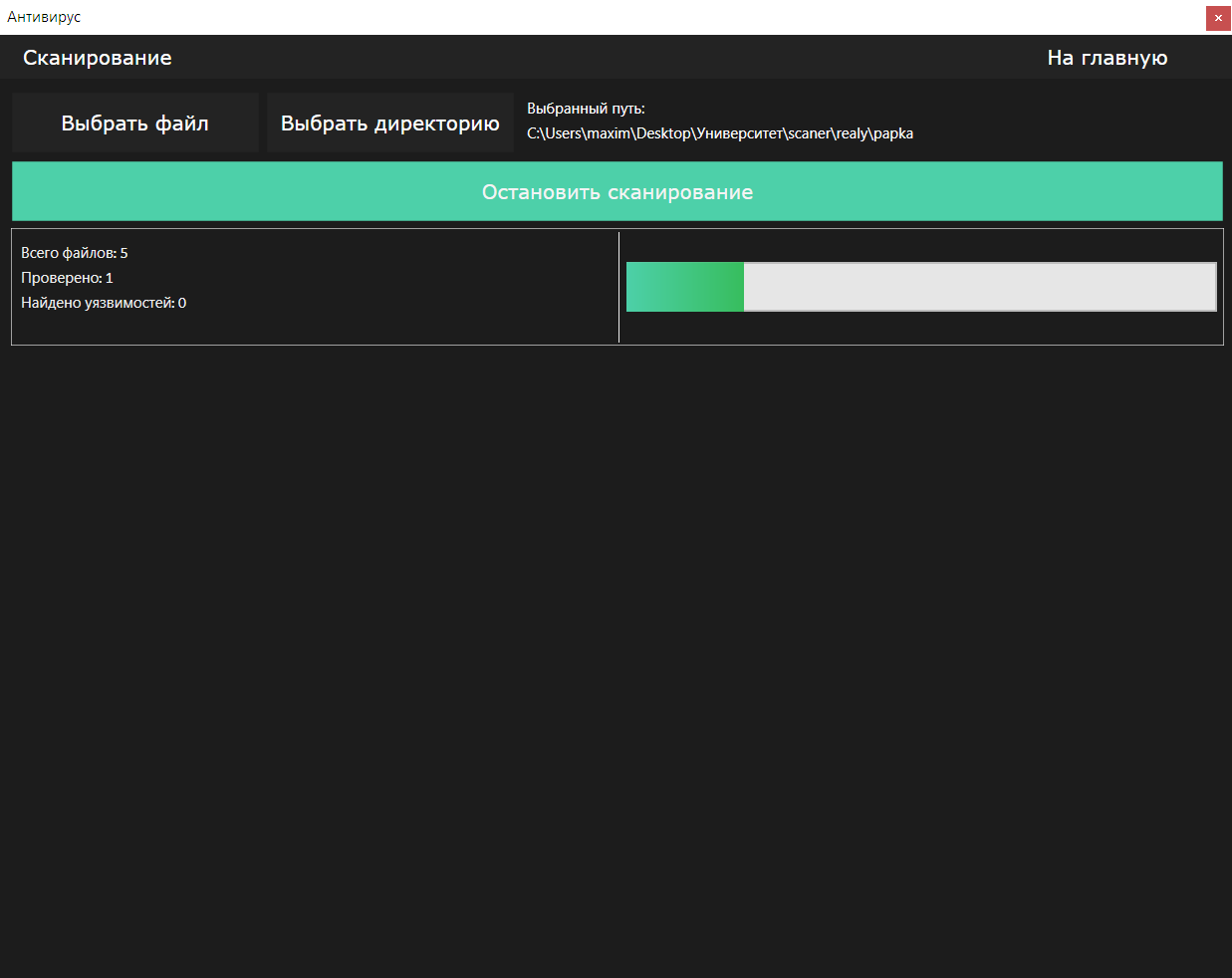


Рисунок 15 – Сканирование файлы

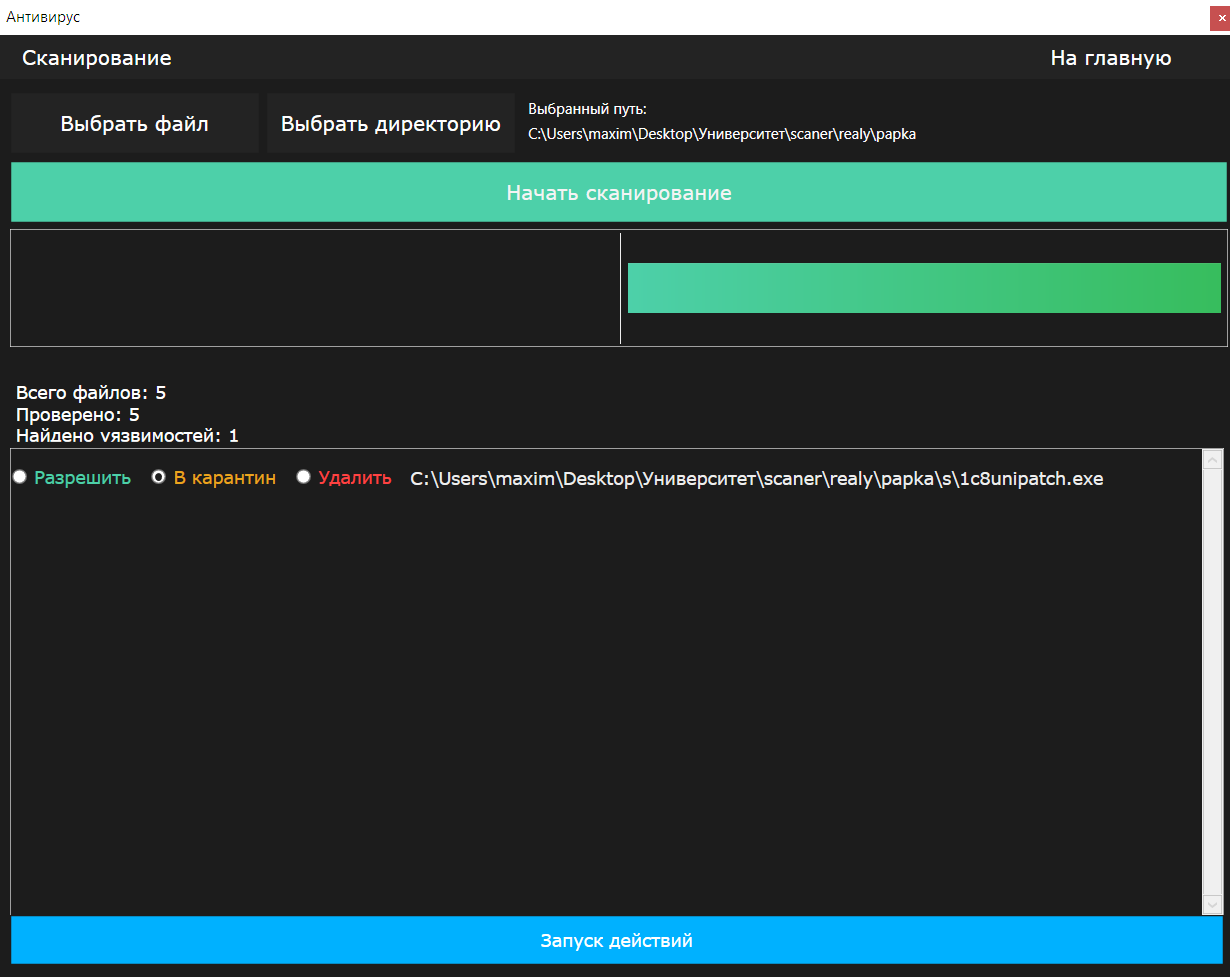


Рисунок 16 – Запуск действий и добавление в карантин файлов

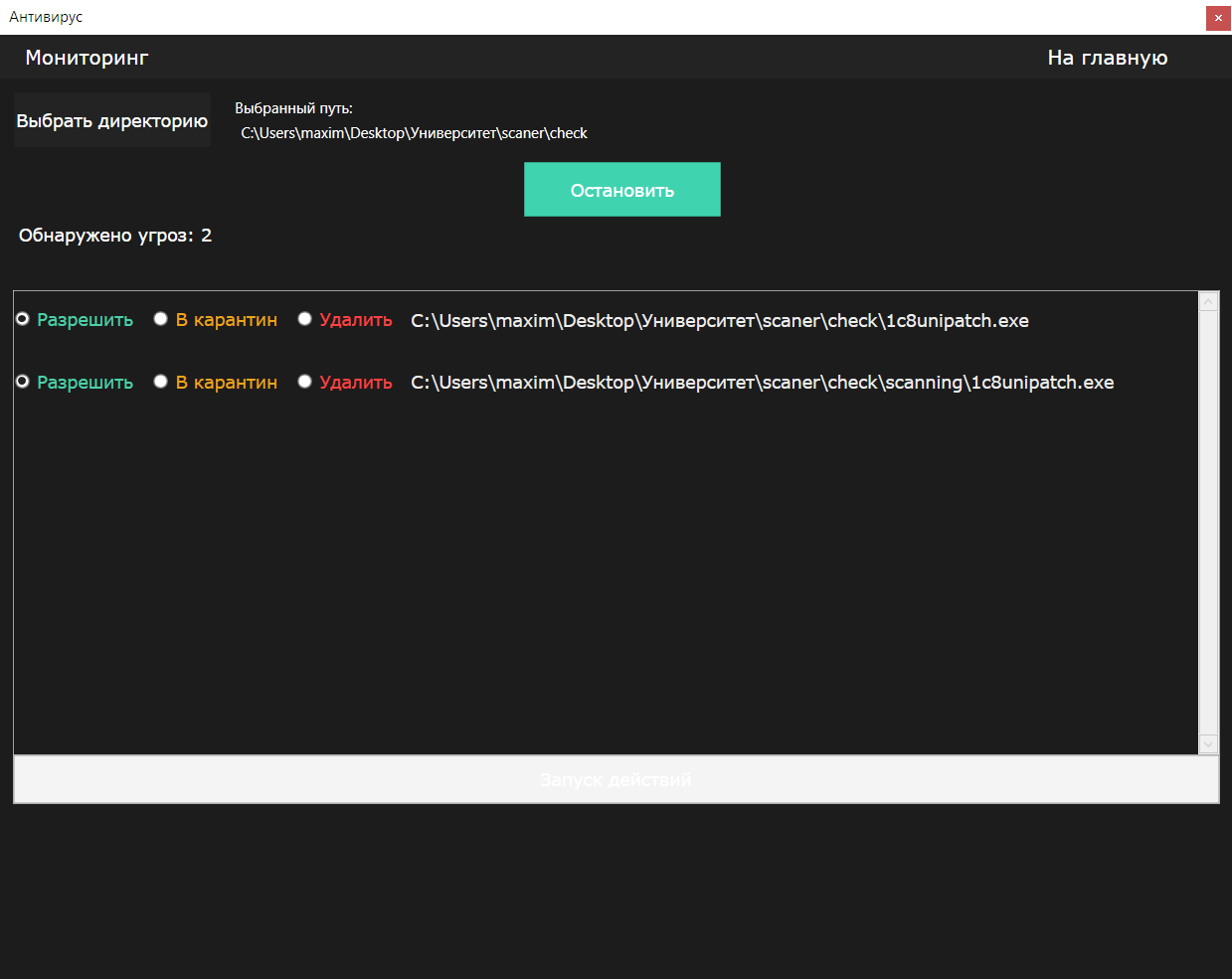


Рисунок 17 – Осуществление мониторинга директории

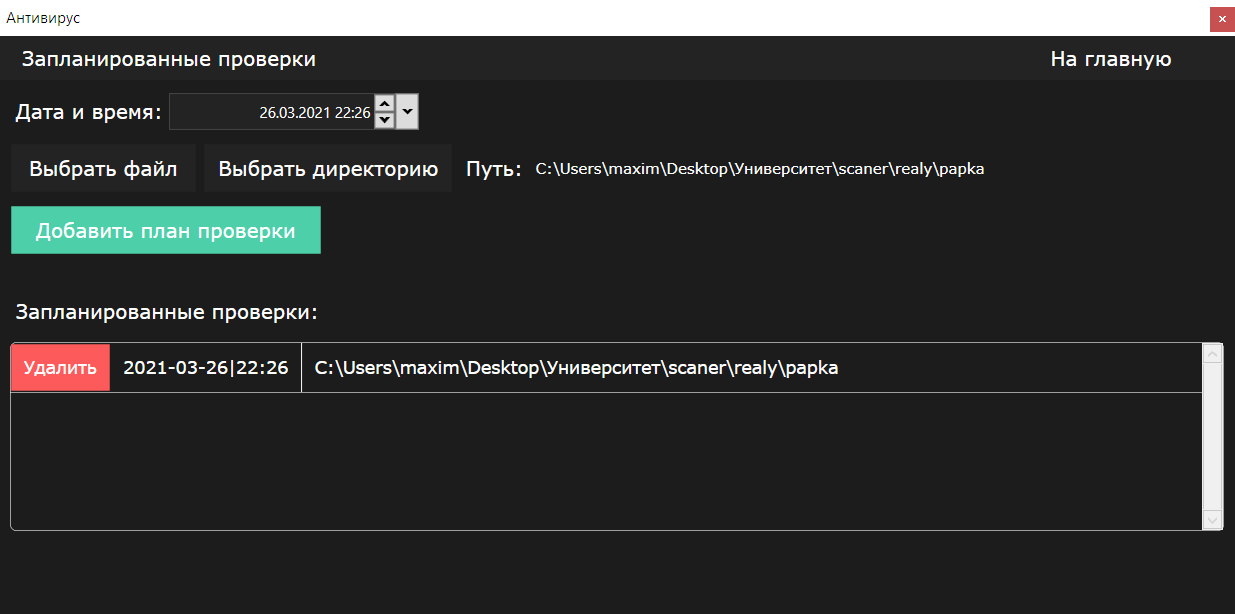


Рисунок 18 – Добавление плана проверки

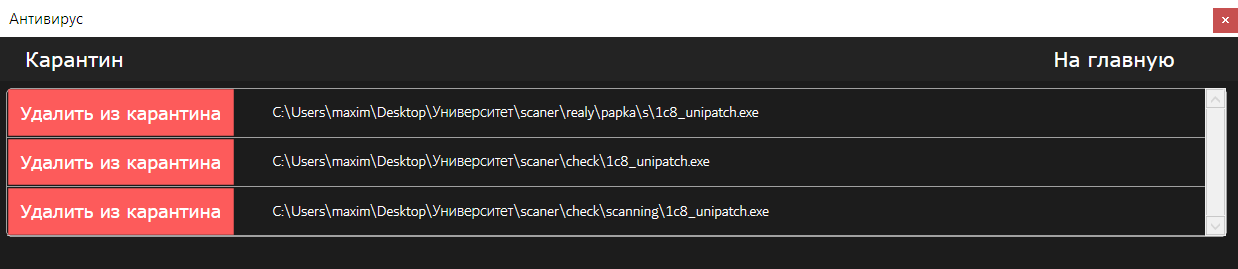


Рисунок 19 – Список файлов, добавленных в карантин

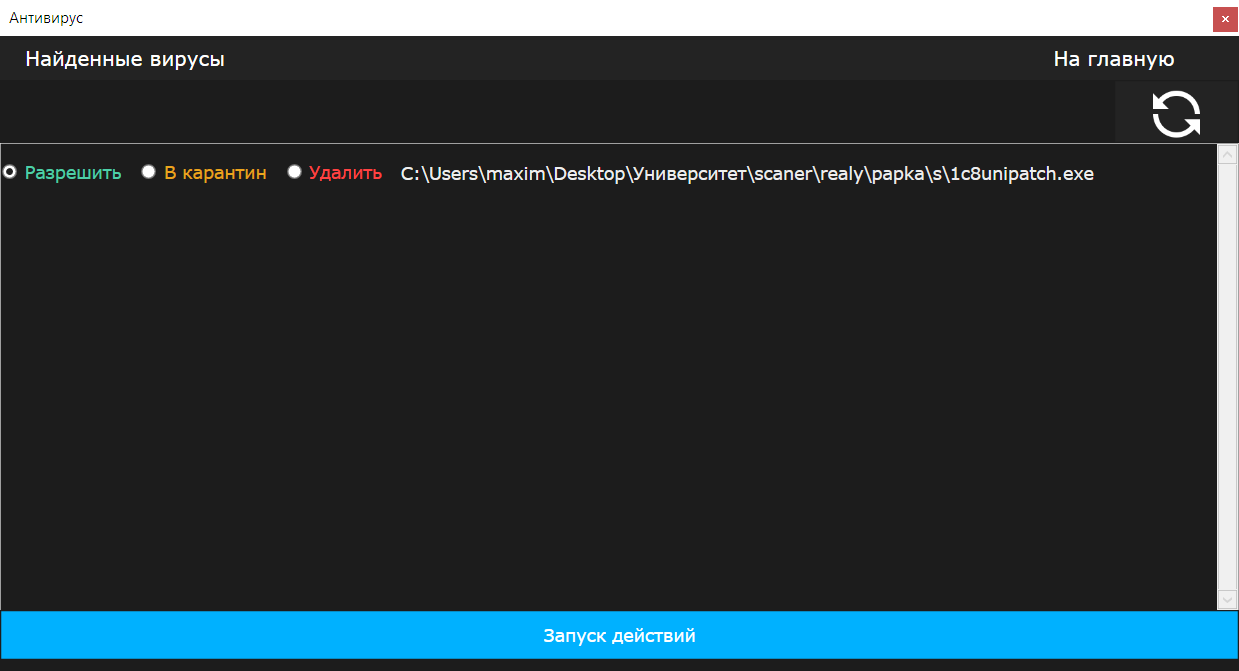


Рисунок 20 – Результат плановой проверки

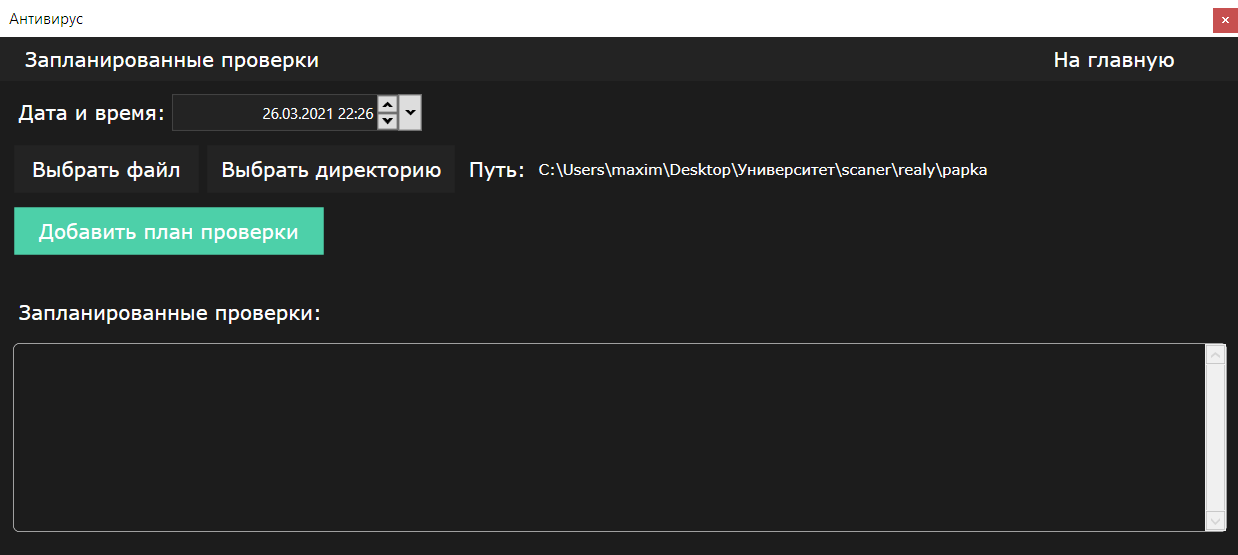


Рисунок 21 – Окно запланированных проверок после выполнения

Вывод:

В результате практической работы был разработан комплекс для антивирусной защиты, предоставляющий функции мониторинга, сканирования, запланированных проверок, работой с удалением, добавлением в карантин, удалением из карантина файлов.