

**Курсов** **проект**

„ Създаване на програма за автоматизирани месечни отчети на преподавателите от ВВМУ “

|  |  |
| --- | --- |
| **Студент:** | Николай Николов |
| **Специалност:** | Киберсигурност |
| **Фак. №** | 626-201 13 |

Съдържание

[Кратко описание и цел на проекта 3](#_Toc90499726)

[Използвани технологии 3](#_Toc90499727)

[Visual Studio Code 3](#_Toc90499728)

[GitHub 4](#_Toc90499729)

[Python 4](#_Toc90499730)

[Pip installer 4](#_Toc90499731)

[pyinstaller 5](#_Toc90499732)

[tkinter 5](#_Toc90499733)

[requests 6](#_Toc90499734)

[re 6](#_Toc90499735)

[configparser 6](#_Toc90499736)

[datetime 6](#_Toc90499737)

[openpyxl 6](#_Toc90499738)

[Логическа реализация 7](#_Toc90499739)

[Техническа имплементация 10](#_Toc90499740)

[Графичен Интерфейс 10](#_Toc90499741)

[Прихващане на заявка 12](#_Toc90499742)

[Експортиране 13](#_Toc90499743)

[Конфигурация и техническа информация. 14](#_Toc90499744)

[Хеширане на файлове 14](#_Toc90499745)

[Обобщение 14](#_Toc90499746)

[Библиография 16](#_Toc90499747)

[Апендикс 17](#_Toc90499748)

[Графичен интерфейс 17](#_Toc90499749)

[Прихващане на заявки 20](#_Toc90499750)

[Експортиране 23](#_Toc90499751)

[Конфигурация 24](#_Toc90499752)

[Логър 25](#_Toc90499753)

# Кратко описание и цел на проекта

Проектът представлява изготвяне на програма, с която лесно, бързо и практично да се изготвят месечни времеви отчети, относно преподадените лекции и упражнения, във формата на таблица на **Microsoft** **Excel**. Считайки, че цялата информация, която се използва е публично достъпна, не се изисква автентификация и не подлежи на нужда от засекретяване, е напълно удачно документирането и да бъде автоматизирано. Написването на програмата ще се случи чрез използването на програмен език **Python**, редом с допълнителни независими библиотеки за по-трудно постижимите функции (пр. създаване и манипулация на таблица в **Excel**). Въпреки че проекта не е пряко свързан с дисциплината киберсигурност, неговото предназначение е да има реално приложение и да спести време на преподавателите от университета. Въпреки това,

# Използвани технологии

Следва списъкът на всички използвани технологии в разработката на този проект, основните такива биват разпределени в главни отдели, а по-малките свързани с тях в подотдели.

## Visual Studio Code

**Visual studio code** (Microsoft) е платформата, която се използва за написването и компилацията на самият код. Без значение на какво е написана една програма, тя трябва да има достъп до интерпретаторите и компилаторите на един език, за да може компютъра да разбере какво трябва да се случи. Това са неговите указания как да разчита написаното от нас. Теоретично погледнато, кодът може да бъде написан и в един елементарен **notepad** и после компилиран отделно, но това би било изключително непрактично и неудобно. Платформата позволява събирането на всички тези елементи на едно място за удобно използване. Освен това тя предлага:

* **Intelisense** – възможност за интелигентно автоматично довършване на кода. Чрез сканиране на вече съществуващите библиотеки, платформата може да предложи всички възможни променливи и функции на един обект, заедно с документация, ако такава съществува. Подобно на стандартният табулатор в една конзола, с разликата, че предлага повече от едно решение, той улеснява работата на програмиста като спестява време от правописни грешки и пълно изписване, както и предлага нужната информация за използването на един обект на място, вместо тя да бъде търсена в първоначалните източници. **Intelisense** е част от всяка подобна платформа в днешно време и е неразделна част от набора инструменти на един програмист.
* **Портативност** – самата програма е портативна версия на основната такава **Visual** **Studio**, тя е превърната в не-натоварваща модулна платформа, която може да бъде използвана за изключително голям набор от езици спрямо желанието и нуждите на използващите я. Не изисква инсталация, а всички файлове се съдържат в папката при изтегляне.
* **Extensions** - Програмата бива допълнена от екстеншъни, които и позволяват да върши стандартната си работа. В този случай, **Python** също играе роля на такъв екстеншън, след което чрез него ще бъдат инсталирани допълни такива. Тези екстеншъни могат да бъдат инсталирани и деинсталирани с няколко клика, което позволява и бързото рефакториране на програмата за нови цели, въпреки че това рядко се налага.

## GitHub

**GitHub** (GitHub, Inc) е разширение с графичен интерфейс и по-големи възможности на системата контролиране на версиите **git** (Software Freedom Foundation). Подобна система поддържа списък от всички налични версии на една програма, т.е. в нея се съхраняват сегашното и състояние, както и всичките и минали такива до началото на нейното съществуване. Тъй като доста програми са не-малки, тази задача би била изключително натоварваща върху операционните системи и паметта на сървърите, ако се помнят всички файлове във всяка нова версия на една програма. Затова git един вид пази „снимка“ на състоянието на файловете използвайки криптиращият алгоритъм **SHA1**, за да придобие символично състояние на всеки един файл.

Освен това **git** разрешава проблема с екипната работа върху еднакви файлове. Той позволява разработката на различни части на програмата в отделни „клони“. По този начин всеки човек от екипа може да работи самостоятелно върху файлове от програмата без да се изисква постоянен хардуерен трансфер на данни, и с минимална опасност от препокриване. Единственият проблем, е че той трудно поддържа файлове с големи размери. Съществуват допълнения, които разрешават това като **git lfs**, но тъй като за проекта това няма да е нужно, той няма да бъде разглеждан.

Последното предимство на използването на тази система, че тя проекта може да бъде закачена към сървъра на **GitHub**, вместо да бъде съхраняван на локална машина. **Git** позволява лесно теглене на проекти, което значи, че всеки преподавател може лесно да достъпи и използва проекта.

## Python

**Python** (Python Software Foundation) или питон е език за програмиране на високо ниво, без специализирано предназначение. Създаден е през 80-те години и в момента е един от най-известните езици за програмиране. Отличава се с леснота на използване и модулност. Предназначен е да се използва и допълва спрямо нуждите и е високо допълним. Също така поддържа множество стилове на програмиране. Особено е, че използва стандарт за кодиране **PEP-8** (Rossum, et al., 2001). Използва табулатори, които служат както за четимост, така и за структура. Чрез използване на табулатори питон разбира къде има логическа или структурна промяна. Тяхната липса не позволява на програмата да се компилира правилно. Също така създателите на питон целят той да е приятен за използване, това е подчертано от името му, което идва от известната британска комедийна трупа Monty Python (Python (Monty) Pictures Limited). **Python** може да бъде инсталиран като extension на **Visual** **Studio** **Code**.

### Pip installer

**Pip installer** (The pip developers) е част от основните модули на **Python** и инсталацията му идва, заедно с тази на питон. Той е главният инструмент, чрез който биват инсталирани всички останали нужни модули от основните и допълнителни библиотеки на питон. Той осигурява достъп до тях, както и лесен мениджмънт. Това се случва чрез няколко кратки команди:



Първата команда е главният начин за инсталиране на пакети чрез инсталатора. Втората е подобна, но съдържа допълнителен аргумент *--upgrade*, който ръчно проверява за наличие на нови версии и, ако такива съществуват, ги инсталира. Ако той не бъде използван ще бъде проверено наличието на пакета като цяло, а ако той е стар това може да не е достатъчно. Последната команда премахва съществуващ модул. Въпреки че тях рядко бива използвана, има допълнителни случаи, в които тя се оказва доста полезна. Един подобен ще бъде описан в този проект. Освен тези команди съществуват и още доста други спрямо нуждите на използващият го, които няма да са нужни за разработката на проекта. Освен това има и различни аргументи при използването на базовите, какъвто е *--upgrade*, те позволяват разнообразни неща като задаване на конфигурационен файл при инсталация, избиране на специфична версия и други.

### pyinstaller

**Pyinstaller** (PyInstaller Development Team, 2021), съкратено от python installer, е модул от допълнителните библиотеки на python, който позволява компилирането на скриптове във изпълними програми, т.е. превръща файловете с *.python* разширения в .*exe*. Има и други такива модули, но този се отличава с поддръжка на новите версии на питон, създаването на по-малки на големина файлове, и идентично действие независимо от платформата, върху която файловете са създадени или изпълнени. Всеки файл може да бъде компилиран еднакво както на **Windows**, така и на **Linux**, след което да бъде изпратен на друга система и да работи по очакваният начин.

За изискванията на този проект, **pyinstaller** ще бъде използван за превръщане на python скриптовете в изпълним файл с графичен интерфейс, който е по-интуитивен за използване от командният ред. Това позволява по-лесно и достъпно използване.

### tkinter

**Tkinter** (Python Software Foundation) или tk interface е стандартно допълнение за функционалност на графичен интерфейс към езикът за програмиране **Tcl** и собственото му разширение за интерфейс – **Tk**. **Tkinter** е **Python** обвивка на съответната библиотека. За да може тя да съответства на езика възможно най-добре, тя включва и голямо количество самостоятелна логика, вместо да е просто тънка обвивка около съществуващата библиотека.

Библиотеката служи за създаване на графични интерфейси за програми. Имайки доста възможности, с нея може да бъде създаден както елементарен интерфейс като калкулатор, така и сложни интерфейси с множество функционалности, различни екрани и логически разделени пътища на изпълнение, както примерно би бил интерфейсът на един инсталатор. Едно от основните преимущества на библиотеката е, че дава възможност за използване на вече съществуващи елементи, наречени widgets. Това са често използвани елементи в дизайна, като надписи, полета за вход на данни, бутони, изображения и други. Това помага за лесното и бързо създаване на прототип или дори на пълна програма за сравнително по-малко време, отколкото би било нужно при написването на всички тези елементи от нулата. Тъй като не би могло да се разчита единствено на тези елементи, библиотеката позволява специфично наместване на съществуващите елементи по множество начини, включително тяхното точно разположение, размер, промяната на такъв при промяна на размера на екрана и други. Всички тези елементи превръщат библиотеката в необходим инструмент при изграждането на проекта.

### requests

Библиотеката за интернет заявки – **requests** (Reitz) е точно това, което звучи. Чрез нея можем да осъществяваме всички съществуващи стандартни заявки с протокол – *HTTP/HTTP*S (пр. *GET*/*POST*). Тя също е допълнителен модул за питон, който може да бъде инсталиран чрез pip installer. Тя ще бъде използвана за прихващане на съществуващата информация в публичният домейн на университета относно седмичните програми на учителите. Резултатът от прихващането на заявката е *HTML* код, което ще рече, че тя трябва да бъде детайлно обработена и интересуващата ни информация да бъде изчистена и структурирана, преди да е възможно тя да бъде пренасочена към файл на **Excel**.

### re

**Re** (Python Software Foundation), съкратено от regex, е основна библиотека в модулите на питон, която служи за извличане и обработка на стандартни изрази от сложни данни. Името на библиотеката буквално идва от съкращението и обединението в една дума на „стандартен израз“ (regular expression). Тя съществува в много близки, почти еднакви, състояния в почти всички езици за програмиране, тъй като е широко използвана за намиране на информация, валидация, и други операции в много програми. Въпреки, че за по-сложни изрази, нивото на четимост рязко пада, скоростта на regex го кара да бъде често избиран (пр. валидация на телефон, пощенски код, имейл). След прихващането на заявка, тази библиотека ще бива използвана за изграждането на регулярни изрази, спрямо които да бъде извлечена необходимата ни информация.

### configparser

**Config** **parser** (Python Software Foundation)е основен модул, който позволява по-детайлна манипулация на файлове от стандартният модул **os**, със специализация към конфигурационни файлове. Той ще бъде използван за съхраняване на настройките на програмата при всеки следващ експорт. Тоест, за улеснение на използването, той ще запаметява и зарежда последно използваните настройки за експорт в програмата. Това ще доведе до създаването на всеки следващ експорт да бъде сведено до отваряне на програмата, смяна на месеца и натискане на бутон „експорт“, което е доста по-удобно от ръчното търсене и въвеждане на цялата информация.

### datetime

Последната библиотека, която е от по-голямо значение е основният времеви модул на python (Python Software Foundation). Тъй като заявките на университета работят с номер на седмица, а времевите отчети са месечни е необходима такава програма, която надеждно да изчислява кои седмици участват в даден месец, след което да извлича информацията за всяка една от тях. Съответната библиотека позволява подобни изчисления да бъдат извършени сравнително лесно, тъй като повечето от тях вече са налични като вградени функции.

### openpyxl

Името **openpyxl** (Gazoni, et al.) идва от комбинацията open source python excel, т.е. свободно разпространена библиотека за работа с екселски файлове използвайки питон. Чрез нея се осъществява манипулация на файлове от съответният тип на множество нива – файл, таблица, ред, колона или клетка. Използвайки тази библиотека е възможно цялата информация от заявките да бъде запаметена и форматирана в екселски файл.

# Логическа реализация

Логическата реализация на проекта се извършва основно от програмата. Потребителят има достъп до избора на параметрите, с които да се изпълняват заявките, но извън това няма нищо общо с изпълнението на програмата. Единственото, което ще се изисква от него е да отвори програмата и да въведе желаните параметри, след което да отвори репорта и да потвърди самостоятелно, че всичко вътре е изрядно.

Системата се грижи за зареждането и използването на всички конфигурационни настройки, както и какво се случва ако някоя от тях, или целият файл, липсва. След което, при изпълняване на експорт, взаимодейства с изброените библиотеки на логически отделени части. Изчислява нужните параметри спрямо зададените такива и изпълнява заявки, докато месечната информация бива пълна. Тя бива филтрирана и изпратена към **openpyxl** библиотеката, която се грижи за структурирането и съхранението на информацията в Excel, както и за менажирането на самият файл.

Има няколко логически проблема с изпълнението на програмата по този начин. Първо, програмата е напълно зависима от състоянието на сайта на университета. Всяка промяна в структурата на заявките или състоянието на сървъра съответно оказва критично влияние върху функционалността на програмата. Считайки, че това са рядко и трудно случващи се събития, рискът от подобен проблем е нисък.

Второ, тъй като заявките винаги връщат резултат, без значение дали такъв съществува за съответното класно отделение, преподавател или зала, експорт винаги ще бъде създаден. Дори ако въведената информация не е валидна, ако параметрите отговарят на типа данни (пр. цифри за класно отделение) заявката ще бъде изпълнена успешно, но резултатът ще е празна таблица. В този случай е невъзможно да бъде направена разлика между липсващи данни за седмица и грешно въведени параметри. Единственият начин това да бъде потвърдено е със проверка на експорта след създаването му. Доста вероятно е резултатът да е такъв при наличието на празна таблица, а очаквани попълнени резултати. С цел минимизиране на шанса от грешни заявки в случай на правописни грешки и др. въвеждането на параметрите ще бъде ограничено или валидирано.

Логическата последователност може да бъде проследена в следната таблица:

Graphical user interface

Description automatically generated

Graphical user interface

Description automatically generated

Graphical user interface

Description automatically generated

# Техническа имплементация

В тази част ще бъдат разгледани техническите детайли на имплементацията на програмата. Референции към кода на програмата могат да бъдат намерени в апендикса. Възможно е логическото обяснение на програмата да е в обратен ред на написаният код, тъй като в **Python** не е възможно да се използва променлива или функция, преди да бъде създадена, т.е. всички детайли на използването трябва да са обявени и инициализирани преди да бъдат извикани от друго място.

## Графичен Интерфейс

Програмата започва логически от графичният интерфейс [(1)](#_Графичен_интерфейс), използвайки библиотеката **Tkinter**, създаваме начален екран и съвместими променливи за всеки желан параметър. Тъй като репортите са месечни, а не седмични, интерфейса показва поле за избиране на месец, вместо седмица. Съществуват съответно основните три параметри, заедно с допълнителен такъв за избиране на папка, в която да бъде създаден крайният експорт.

След това се използва графичната библиотека за създаване на всички нужни елементи – полета за вход на данни и техните наименования, бутони и други, и тяхното разположение и форматиране. Библиотеката съдържа няколко вида инструменти за геометрично подреждане на елементи. Основният е grid geometry manager. Чрез него елементите могат много лесно да бъдат разпределени в квадратна мрежа с избран брой редове и колони, като последният ред е извикването на *root\_frame.mainloop()* [(1, ред 107)](#_Графичен_интерфейс), което казва на библиотеката да поддържа елементите на екрана (т.е. няма да се загаси веднага, след като тръгне). Програмата цели да е разположена по удобен начин като също така по този начин дава възможност за лесно разширение при бъдеща нужна от промяна (пр. избиране на наименование на файл). Разпределението и се води от следният шаблон:



Graphical user interface, application

Description automatically generatedИ в резултат изглежда така:

Спрямо споменатите проблеми с валидацията на параметрите на заявката, избирането на вид на заявка и месец представляват избиране на стойности, заредени от статични списъци с променливи. Списъците съдържат само стойности, които са сигурни да работят, освен ако няма промяна в начина на действие на уебсайта. Потребителят няма право избор на друг елемент и, тъй като точно това са стойностите с най-голяма вероятност за грешка в сравнение с номера, рязко пада възможността за изпращане на заявки съдържащи невалидни елементи. Със същата цел е добавен бутон към избирането на съдържаща папка за финалният експорт. При директно избиране на съществуваща папка няма възможност за правописна грешка. Въпреки това може да има ситуация, при която потребителят копира целият път от файлов браузър, след което го копира директно в програмата. За такъв случай, полето не е направено с единствен достъп от бутона.

От там нататък е счетено за излишно добавянето на валидация на полетата за номер или папка. Отговорността за верни данни се пренася върху потребителят. Считайки, че информацията съдържана в заявките е публична и не крие нищо тайно, а главният и път е твърдо зададен [(2)](#_Прихващане_на_заявки) в недостъпна част на програмата и единственото, което може да бъде променено са параметрите и, се счита, че опасност от кибератака не съществува, или при проба за такава е невъзможно да бъде извлечена полезна информация. След проверка на информацията и натискане на бутон експорт, закачената към него функция се изпълнява, стойностите в конфигурационният файл се презаписват и хода на програмата се предава към прихващача [(2)](#_Прихващане_на_заявки).

## Прихващане на заявка

Прихващането на заявките става от следващата част на програмата [(2)](#_Прихващане_на_заявки), която приема зададените параметри от графичният интерфейс и формулира серия от заявки спрямо тях. Първото нещо, което се случва е да бъде взето името на месеца и да бъде превърнато в индекс с помощта на времевата библиотека. Това се прави, за да бъде използван след това, за намиране на първата седмица от този месец, тъй като функциите на библиотеката работят с номер, а не с наименование на месец. След това по същият начин биват намерени и общият брой седмици в съществуващият месец. Създава се празна структура от данни, в която да се съдържа ежедневната информация и, започвайки от първата седмица, заявки се формулират по следната формула:

*Основният път към сайта + идентификационен номер + вид заявка + номер на седмица*

Или, ако искаме да направим заявка за класно отделение 626201 за първата седмица на Октомври (39), тя ще изглежда така:

*https://nvna.eu/wp/?****group****=626201&****queryType****=group&****Week****=39*

Октомври има 5 седмици, съответно ще бъдат създадени 5 различни заявки в подобна форма. Всяка заявка ще бъде създадена и изпълнена с помощта на модул **requests**. Отговорът от всяка е *HTML* кодът на цялата страница, т.е. това което иначе би видял потребителят в сайта е показано в оригиналният му вид преди браузърът да го превърне в смислен вид. След прихващане информацията се препраща към друга функция, която извлича нужната ни информация и я добавя към временна структура. След края на всяка заявка, временната структура бива добавена към месечната, след което автоматично се унищожава преди създаването на следващата заявка.

Извличането става чрез използването на модул **re**. Използват се няколко различни регулярни израза, самостоятелно или в комбинации за постигане на желаната цел. Най-кратко казано, изразите се състоят от очаквани точни части (пр. при търсене на програмата за понеделник се търси мястото в таблицата, където пише точно „Понеделник“), комбинирано с шаблонни части (пр. след името на всеки ден е неговата дата, ако първият понеделник на Октомври е **2021-10-04**, тогава ще се търси като **[0-9]{4}-[0-9]{2}-[0-9]{2}**, което ще търси поредица от четири цифри, после две, и после още две, с тире между всяка група). По подобен начин са формулирани и останалите изрази, но изглеждат по-сложно и е излишно да бъдат обяснявани детайлно. Съществуват три основни случая, които трябва да бъдат покрити от изразите:

* ден без часове
* академичен час от деня, които е начало на лекция/упражнение
* академичен час от деня, които е продължение на вече започнала лекция/упражнение

Причината случаите да са три, а не два (с/без лекция), е че кодът изглежда по-различен начин спрямо дали един ден е празен, или ако не дали всеки негов час е начало или продължение на лекция, като структурата на празните часове, които не са продължение на лекция съвпадат със стандартният код за начало на такава, но липсва информация в тях. Съществуват 4 различни шаблона изрази:

* Структурата на заглавният ред на таблицата, показващ деня и датата (пр. Понеделник, 2021-10-04).
* Структурата на ден, в които няма лекции.
* Структурата на ден, в които има лекции.
* Структурата на самостоятелен час.

Разделени са по този начин за лесно комбиниране, тъй като се препокриват. За всеки ден от седмицата се извършват следните действия - Тъй като е най-семпло, първо се изпълнява търсенето за заглавен ред, заедно с ден без лекция. Ако върне резултат, към временната структура се добавя списък съхраняващ деня и изречението „Няма занятия“. Ако не, програмата използва комбинация от заглавният ред и структурата на цял ден, в които има лекции. Тук не се интересуваме от самата информация, а дали такъв ден ще бъде намерен, тъй като това улеснява бъдещото извличане на информация да бъде фокусирано върху резултата от това търсене. То ще бъде кодът на един единствен ден, вместо да бъде правено търсене из цялата седмица, тъй като е излишно и може да има грешно намерени изрази. При резултат, програмата използва структурата на самостоятелният час и намира всички такива попадения. Към временната структура се добавят името на деня и информацията за всеки час разделен в четири части – номер на академичният час, в които е започнал, времевото продължение, дисциплина и преподавател. Така временната структура представлява списък от списъци – в случай, че за един ден не е имало лекции, списъка е с дължина два елемента, ако да е с дължина *n*, за която винаги е валидно условието *n-1=4\*x*, където *x* е броят часове. Това ни позволява по-нататък в програмата лесно да предскажем какво е нужно и не е нужно да бъде добавено в крайният файл спрямо големината на всеки списък. Последно, в случай, че програмата не върне отговор за нито един от изразите, нищо не бива добавено. Това се случва само в първата или последната седмица на даден месец, където има дни от други месеци, тъй като индексът на всеки месец участва в търсенето.

След като всички заявки бъдат направени успешно и информацията е събрана в списък, тя се препраща към следващата част на програмата и съхранението и в екселски файл.

## Експортиране

Експортирането е сравнително по-просто от останалите части на програмата. Извикването на функцията за създаване на нов файл става с три параметъра – списъка с информация, месеца, и папката за съхранение. Използвайки модул **openpyxl**, програмата създава нов файл, чието има е съчетание от статична фраза, датата на създаване и името на месеца. Файл, създаден на първи Декември, за месец Октомври би изглеждал така: *NvnaExport\_October\_2021-12-01.xlsx*. Всеки нов файл с подобни параметри би презаписал предишният. Към момента програмата не позволява избирането на име за файла, което значи, че репорти за различни класни отделения с препокриващ се месец направени в един и същи ден изискват ръчно преименуване, за да не бъдат затрити. Но това не е голяма промяна и при нужда може да бъде променено сравнително лесно.

При извикването на функцията, програмата създава името на крайният файл, и за всеки елемент от списъка, добавя информацията в него в последователни колони, като всеки следващ елемент е на нов ред. Ако всички действия до тук са били успешни, тук програмата връща съответният отговор към графичният интерфейс и се показва съответният диалог, потвърждаващ на потребителя, че репортът е създаден без проблеми, заедно с папката, където е създаден.

## Конфигурация и техническа информация.

Освен основната логика съществуват още две части на програмата – конфигурационната и информационната.

Конфигурацията [(4)](#_Конфигурация) се грижи за създаване и запазване на използваните параметри за по-лесно бъдещо използване. Тъй като е най-добре програмата да се разпространява с минимално количество файлове, тя не разчита, че потребителят има или знае как да конфигурира съответният файл. Затова при всеки експорт се изпълнява функция, която запаметява новите стойности на параметрите, но първо проверява дали файлът съществува. Ако не, се създава нов такъв с разширение *.cfg*, след което се добавя секция *[request\_parameters]*, където ще бъдат съхранявани всички желани стойности. След това продължава стандартното изпълнение на програмата.

Информационната част се грижи за събирането на техническа информация за случващото се в програмата по време на изпълнение. Важно е да се отбележи, че тя по никакъв начин не събира или анализира информация за потребителят, тя служи само и единствено за проследяване на държанието на програмата в случай на проблем. Основната логика за конфигурирането на логъра се съдържа в отделен файл [(5)](#_Логър), към които реферират всички останали. Съхранява се информация при стартирането на програмата, при изпълнение на експорт, както и самата информация от заявките, филтрирана и подредена за всеки ден. По този начин може лесно да се проследи точно къде е настъпил проблемът, при наличие на такъв.

За да може всичко да е четимо и на български език, всичко свързано с отваряне, затваряне или модификация на файлове е със статично зададен стандарт за кодиране *UTF-8*.

## Хеширане на файлове

Тъй като проекта ще се разпространява с изпълним файл и оригиналният код, като допълнение против зловредна промяна във файловете е добавен скрипт, които изчислява хешираната сума на всеки *.py* скрипт, както и на самият изпълним файл .*exe* по алгоритъм SHA-256 (Eastlake, et al., 2011). Изчислените суми за всеки файл се съхраняват в текстови файл и могат лесно да бъдат пресметнати с инструменти каквито се намират на всяка операционна система. Промени по кода съответно ще укажат влияние върху сумите. За най-голяма сигурност е добре изчисленията да се правят спрямо файлът в сървъра на **GitHub**, откъдето може да бъде изтеглен проекта.

# Обобщение

В този доклад е представен проект за разработка на програма, която служи за автоматично създаване на месечни отчетни доклади. Целта на проекта е да спести време на учителите във Военноморско училище „Никола Йонков Вапцаров“, като изпълнява това вместо тях, спестявайки време. Отбелязани са всички използвани технологии и са референции към мястото, от където са взети, и където е описан начинът на тяхното използване. Целият използван софтуер по време на проекта е безплатен и свободно-разпространим за не-комерсиални цели. След описание на логическото поведение на проекта е показано реалното протичане на имплементацията. Показани са логическите проблеми идващи от разчитането на външен източник на информация – сайтът на университета, както и специфичните разминавания между стандартното изкарване на седмична програма в <https://nvna.eu/wp>, и начинът, по които това се прави в програмата. Рискът от невалидни данни е намален максимално спрямо допустимото от страна на разработка, останалото зависи от личното потвърждение на потребителя. Счетено е, че поради съществото на използваната информация, риск от кибератака не съществува, или изпълнението на такава е невъзможно да окаже смислено влияние върху потребителят или университета. Въпреки това има шанс от зловредно разпространение на проекта. Той винаги трябва да се взима от първоизточникът му, който е зададен в апендиксът на този доклад. Дори тогава, за по-голяма сигурност е добавен скрипт с резултат от хеширане на всички използвани файлове чрез алгоритъм *SHA-256.*

# Библиография

**Eastlake, Donald и Hansen, Tony. 2011.** *US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF).* неизв. : IETF, 2011.

**Gazoni, Eric и Clark, Charlie.** openpyxl - A Python library to read/write Excel 2010 xlsx/xlsm files. *openpyxl.* [Онлайн] [Цитирано: 15 December 2021 r.] https://openpyxl.readthedocs.io/en/stable/.

**GitHub, Inc.** GitHub. *GitHub.* [Онлайн] [Цитирано: 13 December 2021 r.] https://github.com.

**Microsoft.** Visual Studio Code. *Visual Studio Code.* [Онлайн] [Цитирано: 13 December 2021 r.] code.visualstudio.com.

**PyInstaller Development Team. 2021.** PyInstaller. *PyInstaller.* [Онлайн] PyInstaller Development Team, 11 August 2021 r. [Цитирано: 15 December 2021 r.] https://www.pyinstaller.org/.

**Python (Monty) Pictures Limited.** Monthy Python's Official Website. *Monty Python.* [Онлайн] Python (Monty) Pictures Limited. [Цитирано: 13 December 2021 r.] http://www.montypython.com/.

**Python Software Foundation.** configparser — Configuration file parser. *Python Documentation.* [Онлайн] [Цитирано: 15 December 2021 r.] https://docs.python.org/3/library/configparser.html.

**—.** datetime — Basic date and time types¶. *Python Documentation.* [Онлайн] [Цитирано: 15 December 2021 r.] https://docs.python.org/3/library/datetime.html.

**—.** Python. *python.* [Онлайн] [Цитирано: 14 December 2021 r.] https://www.python.org/.

**—.** re — Regular expression operations. *Python Documentation.* [Онлайн] [Цитирано: 14 December 2021 r.] https://docs.python.org/3/library/re.html.

**—.** tkinter — Python interface to Tcl/Tk¶. *Python Documentation.* [Онлайн] [Цитирано: 15 December 2021 r.] https://docs.python.org/3/library/tkinter.html.

**Reitz, Kenneth.** Requests: HTTP for Humans™. *Requests.* [Онлайн] [Цитирано: 14 December 2021 r.] https://docs.python-requests.org/en/latest/.

**Rossum, Guido van, Warsaw, Barry и Coghlan, Nick. 2001.** Style Guide for Python Code. *Python.* [Онлайн] 05 July 2001 r. [Цитирано: 13 December 2021 r.] https://www.python.org/dev/peps/pep-0008/.

**Software Freedom Foundation.** git. *git.* [Онлайн] https://git-scm.com/.

**The pip developers.** pip documentation v21.3.1. *pip.* [Онлайн] [Цитирано: 13 December 2021 r.] https://pip.pypa.io/en/stable/.

# Апендикс

## Графичен интерфейс







## Прихващане на заявки







## Експортиране



## Конфигурация



## Логър

