|  |
| --- |
| **ТЕХНИЧЕСКИ УНИВЕРСИТЕТ - СОФИЯ**  **Факултет – „Компютърни системи и управление“** |

**ДИПЛОМНА РАБОТА**

**„Наблюдение дейността на студентите по време на изпити провеждани в електронен вариант“**

**Студент: Иван Николаев Колев, фак. № 121213114, гр. 43**

**Специалност: „Компютърно и софтуерно инженерство“**

**Дата: Научен ръководител:**

**/ маг. инж. Д.Андреев/**

**София  
2017**

**СЪДЪРЖАНИЕ**

[Въведение 2](#_Toc493375505)

[1. Постановка на дипломната работа. Цели и задачи. 4](#_Toc493375506)

[1.1. Поток от данни. 4](#_Toc493375507)

[1.2. Вид на събираните данни. 4](#_Toc493375508)

[1.3. Структура на събираните данни. 5](#_Toc493375509)

[1.4. Ресурсоемкост на приложението за събиране на данни. 5](#_Toc493375510)

[1.5. Надеждност и сигурност. 6](#_Toc493375511)

[1.5.1. Балансиране товара на сървърите (Load balancing) 6](#_Toc493375512)

[1.5.2. Менажиране на потребителските сесии 6](#_Toc493375513)

[1.5.3. Защита на вътрешните системи 7](#_Toc493375514)

[1.5.4. JSON Web Token 7](#_Toc493375515)

# Въведение

Една от основните цели на преподавателите в училища и университети е да осигуряват подходящи и равни условия за провеждане на изпити, контролни и упражнения. До голяма степен това се изразява в следене на студентите да не преписват от помощни материали, да работят самостоятелно по възложените задачи и да не се отклоняват от целите на упражнението. Това представлява трудна задача, особено при зали с капацитет повече от пет души, защото фокусът на преподавателя не може да бъде върху повече от един човек едновременно и винаги съществува възможността да изпусне моменти, които няма как да върне и види отново. Именно тези моменти студентите ползват, за да използват забранени материали, да препишат или да отворят сайта на любимата си социална мрежа в която да прекарат времето си.

Изследвания проведени по темата за преписване показват стряскащи числа: процентът на студенти/ученици, които са преписвали поне веднъж започва от 17% и стига до 98% в различни части на света. Част от причините за това са, че за да останеш „конкурентоспособен“ си „длъжен“ да преписваш, защото други, по-слаби ученици, го правят и на резултати изглеждат по-способни от теб.

Компютрите и технологиите като цяло стават все по-достъпни и повече училища и университети започват да разчитат на тях за провеждане на упражнения, изпити и контролни. От една страна, работата по следене дейността на цяла група студенти е непосилна за сам преподавател, но от друга страна, е сравнително лесно да се използват наличните технологии за събиране и анализ на данни свързани с дейностите, които дадено лице извърша на компютъра си.

Основният фокус на текущата дипломна работа е наблюдението дейността на студентите по време на изпити, контролни и упражнения, провеждани в електронен вариант с достъп до интернет, по време на които, изпитваното лице трябва да спазва набор от правила и не трябва да използва забранени канали за комуникация, да търси или публикува въпроси във форуми и други вид информационни сайтове, да използва торент тракери и т.н.

Чрез използване на програмно приложение за Windows OS, студентите ще могат да се „регистрират“ в системата и стартират нова „работна сесия“. По време на работната сесия, приложението ще събира информация за процесите стартирани в операционната система, за текущият фокусиран процес, снимки на дисплея, снимки от уеб камерата, натиснати клавиши, изпратени и получени мрежови пакети.

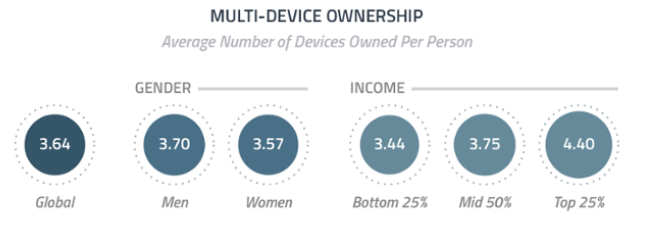
Събраната информация ще може да бъде разгледана и анализирана в реално време или в последващ период.

С предложената имплементация се очаква като резултат, по-ефективно менажиране на учебните и изпитни процеси от страна на преподавателите, по-голяма прозрачност на действията и по-висока ангажираност на студентите.

# 1. Постановка на дипломната работа. Цели и задачи.

## 1.1. Поток от данни.

За последните 20 години, непрестанният технологичен възход промени значително начина по който хората живеят, работят, комуникират и почиват. Масово притежаваме между 3 и 4 мобилни устройства, които ползваме активно през по-голямата част от денонощието. [3] Мобилните устройства могат да бъдат лаптопи, таблети, умни телефони, умни часовници, умни гривни, умни очила, електронни четци за книги, музикални устройства, камери, фотоапарати и т.н. Всяко едно от тези устройства в днешно време има опция за свързване към интернет, към умен телефон или към лаптоп. Това означава постоянен обмен на данни, данни които неусетно могат да станат публични и да бъдат използвани за изграждане на дигитален отпечатък на човека притежаващ устройствата.



***Фигура 1.*** *Статистика за средния брой притежавани мобилни устройства, глобално.*

Потокът от данни включва, но не се ограничава до: уеб сайтовете, които посещаваме и времето прекарано в тях; скоростта, с която натискаме последователно клавиши на клавиатурата; софтуерните приложения, които инсталираме и използваме; имейлите, които изпращаме; хората, с които сме приятели и познаваме; стилът музика, който предпочитаме и любимите ни песни; местата, които посещаваме и пътищата, по които минаваме (благодарение на вградени GPS системи и триангулация от страна на мобилните оператори към които сме абонирани) и т.н.

Всеки ден, използвайки преносими или стационарни устройства, средностатистическият човек може да генерира стотици гигабайти информация. Голяма част от тази информация вече се използва от маркетинг компании с цел продуктово позициониране, показване на подходящи реклами и промоции, целящи да подтикнат човека да закупи вещи, които са в сферата на неговите интереси.

Този голям обем от данни може да бъде използван, както за добронамерени, така и за злонамерени цели.

Затова е важно всеки потребител да използва технологиите с предпазливост и с ясното съзнание, че всеки момент може с действията си да стане уязвим.

## 

## 1.2. Вид на събираните данни.

За целта на приложението, ще се събират данни свързани с начина, по който потребителят използва компютъра, а именно:

* Стартирани процеси/приложения (active processes)
* Текущите процеси на фокус (foreground processes)
* Изпратени и получени мрежови пакети (network traffic)
* Моментни снимки на дисплея, който потребителят вижда (display snapshots).
* Моментни снимки през уебкамерата, ако такава съществува (camera snapshots).
* Натиснати клавиши и скоростта на писане (captured keyboard keys).

## 1.3. Структура на събираните данни.

За целта на системата, данните ще следват един и същ шаблон. Всеки прихванат обект, ще принадлежи на определена „сесия“. Сесия, в контекста на системата, означава всяко ново стартиране на процес за наблюдение.

Всеки прихванат обект, ще се състои от следните 5 полета:

* Име на потребителя
* Идентификационен номер на потребителя
* Идентификационен номер на текущата сесия
* Дата на стартиране на сесията
* Обекта, който е прихванат от приложението (натиснат клавиш, стартиран процес, фокусиране на нов процес, снимка на дисплея, снимка на уебкамерата или мрежови пакет)

## 1.4. Ресурсоемкост на приложението за събиране на данни.

С оглед естеството на данните, които ще се събират, основната част от системата за наблюдение, а именно десктоп приложението, трябва да бъде конфигуруемо.

Следните параметри са от основно значение за ресурсоемкостта му:

* Период, през който ще се правят снимки на дисплея.
* Период, през който ще се правят снимки на уеб камерата.
* Период, през който ще се изпращат агрегираните данни на отдалечения сървър за съхранение.

Информация за активните процеси ще се прихваща, само когато нов процес бива стартиран.

Информация за процеса на фокус ще се прихваща, само когато се фокусира процес, различен от текущия.

Информация за натиснатите клавиши, ще се прихваща при всяко натискане на клавиш.

От прихванатите мрежови пакети ще се записват само първите 24 байта информация, за да се избегне претоварване на мрежата или консумиране на прекалено много RAM памет за временното съхранение на пакетите.

С оглед предоставената информация, само първите три изброени параметъра, ще оказват голямо влияние на разхода на ресурси от приложението, затова те трябва да бъдат избрани внимателно, спрямо машината на която ще работи продукта.

Конфигурациите ще бъдат четени от конфигурационен JSON файл, и ще могат да бъдат променяни спрямо нуждите на различните сесии/потребители/машини.

## 1.5. Надеждност и сигурност.

Първото нещо, за което трябва да мислим, когато работим с потребителска информация е „Законът за защита на потребителските данни“. Всичко което е чувствителна информация, или може да навреди по един или друг начин, ако попадне в злонамерени ръце, трябва да бъде съхранявано по сигурен начин.

За целта, всяко едно предаване на потребителски данни, от компютърът домакин, до сървърите, които ще обработват тази информация, трябва да се случва като информацията се криптира, изпраща се на сървъра приемник, а той я декриптира и използва в оригиналният й вид.

За жалост, все още не съществува 100% сигурен начин за предаване на информация между 2 отдалечени точки и методите, които ще използваме, могат единствено да затруднят по-опитните специалисти.

Второто нещо, което искаме да подсигурим е интегритета на данните и информацията, която постъпва в нашата система.

Понеже ще градим решение, което може да скалира достатъчно добре, за голям обем от потребители, първата линия, която обработва всички потребителски заявки е ферма от front-end уеб сървъри, чиято единствена цел е да получат данните, да идентифицират потребителя, който ги е изпратил, и след това да ги запишат в съответното хранилище, където да бъдат поети за обработка по поточната „тръба“.

Ако при ферма от сървъри, за менажиране на потребителските сесии разчитаме на това състояние да бъде пазено на сървъра, ще си създадем потенциални проблеми и неудобства за самите потребители.

### 1.5.1. Балансиране товара на сървърите (Load balancing)

Има няколко техники за рутиране на потребителските заявки при наличие на ферма от сървъри:

* “Sticky session” е стратегия, при която пред фермата стои load balancer, който прихваща потребителска заявка и винаги за дадения потребител, рутира заявката към един и същ уеб сървър. Създава се таблица с връзки „user” – “server”. По този начин се гарантира, че потребителят няма да има проблеми със сесиите си, както ако бъде рутиран към друг сървър. Тази стратегия не е много надеждна, защото при непланирано спиране на съответния сървър, всички потребителски сесии ще бъдат прекъснати, а load-balancer-ът ще пренасочи потребителите към нов активен сървър.
* “Round robin” е стратегия, при който пред фермата отново стои load balancer, който избира следващият сървър, който да обработи заявката, на ротационен принцип. Това е най-семплият начин за балансиране на натоварването между уеб сървърите, но отново води до проблеми при менажиране на потребителските сесии, ако те се съхраняват на ниво сървър.

### 1.5.2. Менажиране на потребителските сесии

Фазата на идентификация е деликатен момент при наличието на ферма от дистрибутирани уеб сървъри. За решаване на този проблем ще използваме RESTful архитектура, чиято основна цел е да не пазим потребителско състояние/сесия на сървъра, а цялата необходима информация да се предава всеки път, криптирана в бисквитка на браузъра (browser cookie) или като част от тялото на заявката, която ще се декриптира с машинен ключ на ниво уеб сървър и ще се проверява за автентичност.

След като е решен проблема с идентификацията на потребителите, трябва да подсигурим, че всички „входни“ точки в нашите уеб сървъри изискват съответното ниво на автентикациям преди да допуснат каквито и да било данни, напред по поточната линия.

Входните точки още се наричат “endpoint”-и [mark] на уеб сървъра. Това са адресите, към които изпращаме заявки за извличане или създаване на информация.

### 1.5.3. Защита на вътрешните системи

Друга съществена част при защитаване на софтуерната система е подсигуряване комуникацията между отделните модули и валидиране на автентичността на всяка заявка вътрешно в системата.

За решаване на този проблем стандартно се ползват 2 стратегии:

* Конфигуриране на защитна стена (firewall), която да допуска заявки само от определени IP адреси.
* Изпращане на идентификационен токен, който да съдържа идентичността на системата, която прави заявката.

За идентификацията между системите ще използваме много семпла и надеждна концепция, а именно “JWT” (JSON Web Token) [mark].

### 1.5.4. JSON Web Token

Накратко JWT представлява отворен стандарт (**RFC 7519**), който дефинира компактен начин за сигурно предаване на информация (payload) между различни страни, под формата на JSON обект. Предадената информация може да бъде проверена посредством дигитален подпис. JWT токените могат да бъдат подписани използвайки „тайна” ключова дума (за **HMAC** алгоритъм) или двойка публичен/частен ключ (за **RSA** алгоритъм)

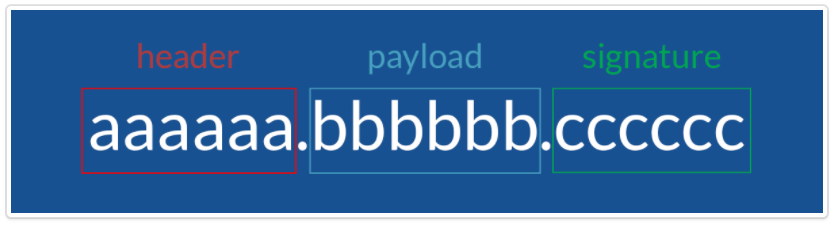
**JWT е подходящ за следните случаи**:

* **Автентикация** – това е най-честия случай за използване на JWT. Веднъж щом потребителят се идентифицира в системата, всяка последваща заявка ще включва токена, позволявайки на потребителят да достъпва пътища, услуги и ресурси, които изискват съответното ниво за достъп. Single Sign On функционалността широко използва JWT в днешно време, заради малкия му размер, бързодействието което осигурява и възможността да се споделя лесно между различни домейни.
* **Предаване на съобщения** – JSON Web Token-ите са добър начин за сигурен пренос на информация между две или повече страни. Поради факта, че тези токени са подписани – можем да сме почти сигурни, че страната изпращаща информацията е тази за която се представя. В допълнение, подписът се генерира използвайки хедърите и самото съобщение, съответно можем да валидираме, че съдържанието не е фалшифицирано.

За надеждна и сигурна работа използвайки този тип токени, ключът, който се използва за подписването им трябва да бъде ръчно сложен в конфигурационния файл на системата, на скрито, непублично място.

Добра практика е през определен период от време да се подменя ключа за подписване.

#### 1.5.4.1 Примерна структура на JWT токен:



***Фигура 2.*** *Структура на base64-encoded JWT.*



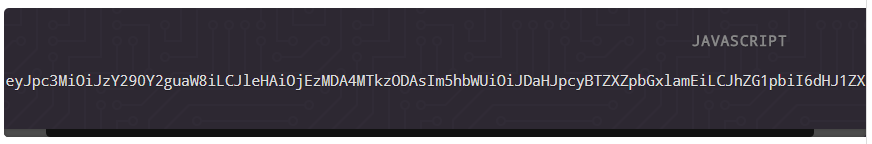
***Фигура 3****. Примерен Header на JWT.*



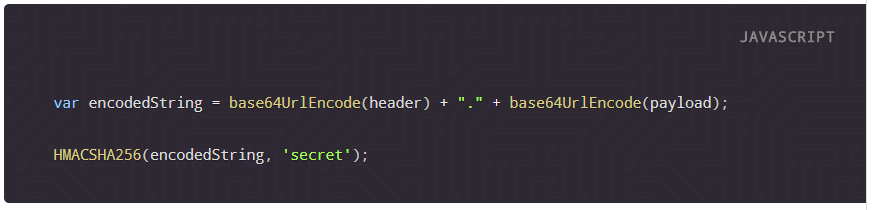
***Фигура 4.*** *Примерен резултат от кодиране на Header в base64 формат.*



***Фигура 5.*** *Примерен Payload на JWT.*



***Фигура 6.*** *Примерен резултат от кодиране на Payload в base64 формат.*



***Фигура 7.*** *Примерно подписване (signature) на JWT посредством JavaScript.*



***Фигура 8.*** *Примерен Signature на JWT.*

. JWT/Cookies/Retry policies/Global exception handling/Firewalls

1.6. Скалируемост на софтуерната система.

1.7. Ресурсоемкост на софтуерната система за обработка и съхранение на данните.

2. Функционално описание на софтуерната система.

2.1. Програмни езици и библиотеки.

2.2.1. Варианти, силни и слаби страни.

2.2. Бази за съхранение на данните (Database servers).

2.2.1. Варианти, силни и слаби страни.

2.3. Поточна линия за пренос на съобщения между системите (Message bus)

2.3.1. Варианти, силни и слаби страни.

Кафка компресира данните посредством gzip за намаляне размера на записваните съобщения.

<http://www.academicintegrity.org/icai/integrity-3.php>

<http://www.cleveland.com/metro/index.ssf/2017/02/cheating_in_college_has_become.html>

Притежание на мобилни устройства

3. <http://blog.globalwebindex.net/chart-of-the-day/digital-consumers-own-3-64-connected-devices/>

JSON WEB TOKEN

4. <https://jwt.io/introduction/>