**Имена:**  Иван Ивов Чучулски **фн:**  *4MI3400043***Начална година:** 2021г. **Програма: магистър ЗИКСМ** **Курс: 1**   
**Тема: Приложение за съхранение на пароли  
Дата: 2022-06-09 Предмет: oopJava2022 имейл: ivan.iv.chuchulski@gmail.com**

**преподавател:** доц. д-р Милен Петров

**Предаване:** Задачата се предава в архив с попълнен настоящия документ, проекта/проектите с кодовете, README.txt файл, който описва съдържанието на архива; папка с допълнителни компоненти и използваниресурси**, архива да се казва oopJava2022\_fn4MI3400043\_FinalProject.zip.**

# ТЕМА: Приложение за съхранение, проверка и генериране на сигурни пароли

(курсов проект по oopJava2022)

## 1. Условие

Целта на проекта е да се проектира и имплементира клиент-сървър приложение, което да позволява на потребителите да записват пароли за използвани от тях уеб сайтове или други услуги . Приложението ще предоставя възможност за генериране на сигурна парола за конкретна услуга, ако потребителите не желаят да измислят такава.

Клиентската част на системата ще поддържа два варианта – десктоп приложение с графичен потребителски интерфейс, което може да се изпълнява на всички основни операционни системи за персонални компютри, а вторият вариант на системата е с конзолен интерфейс, който изисква от потребителя въвеждане на текстови команди.

## 2. Въведение

В днешни времена, когато технологиите са навлезли във почти всеки един аспект от живота ни и едно от най-важните неща е да предпазваме личните си данни от изтичане в Интернет. Паролите са изключително чувствителна информация, която трябва да се знае единствено от потребителя, тъй като чрез нея той удостоверява своята самоличност в услугите, които използва.

Една от често срещаните грешки, която доста хора правят е повторното използване на една и съща парола в множество различни сайтове и услуги. Също така често самите пароли са слаби, тъй като са къси и се състоят от рождени дати и имена. Проблемът се корени в това, че потребителите не могат да помнят много на брой дълги и разнообразни пароли.

Един начин да подобрим своята сигурно е като използваме приложение, което да реши проблема със запомнянето на паролите, а също така и предоставя възможност за генериране на достатъчно сигурна парола вместо потребителя.

## 3. Теория

При съхранението на чувствителна информация като пароли за достъп е необходимо да се предприемат мерки за защита от неоторизиран достъп, модификация или кражба. За целта ще използваме концепциите за криптографска хеш функция и симетрично криптиране на данните.

Паролите, които потребителите подават при регистрация в системата ще преобразуваме с помощта на криптографска хеш функция и ще пазим в базата от данни резултата от пресмятането, който е поредица от байтовете на техния хеш. При опит за влизане в системата се пресмята хеша на въведената от потребителя парола и за успешен вход е необходимо той да съвпада с този в базата данни. Допълнителна сигурност ще ни донесе използването на т.нар. salt – това е поредица от байтове, която произволно генерираме при регистрация, долепяме към подадената парола и така пресмятаме хеша. По този начин се предпазваме от варианта, в който паролите за достъп на потребителите съвпадат и резултата от хеш функцията ще е еднакъв, а също така и на практика обезсмисляме много от brute-force атаките и атаките с rainbow таблици. Ще използваме 16-байтов salt и криптографската хеш функцията SHA-512.



Фигура 1 Хеширане на парола със salt

Тъй като приложението цели да предостави възможността на потребителите да могат да записват пароли за използвани от тях услуги е нужно записаната информация да бъде записана достатъчно сигурно, но да може да възстановена в явен вид. Това се постига с използването на симетрична криптографска функция, която използва един частен ключ, който трябва да остане таен за криптиране и декриптиране на информацията. За генерирането на този частен ключ ще изискваме от потребителя т.нар. “master” парола, която той трябва да въведе при регистрация. При нужда от добавяне или генериране на парола е необходимо потребителят да въведе своята “master” парола отново, като това добавя още едно ниво на сигурност и автентикация след влизането в системата.



Фигура 2 Симетрично криптиране

За образуването на частния ключ ще използваме алгоритъма “PBKDF2”, който дава възможност да добавим и случайно генерирана последователност от байтове(salt на ключа), тъй като отново съществува възможността “master” паролите на някои потребителите да съвпадат, както и правим по-трудни атака за отгатване на ключа, знаейки само “master” паролата. За самото криптиране използваме алгоритъма AES в режим CBC, като той изисква освен текста за криптиране да бъде подадена и случайно генерирана поредица от байтове, наречена инициализационен вектор(IV), която служи за подравняване на данните и увеличава силата на криптирането. В базата данни записваме криптирания текст, байтовете на salt-та за ключа и байтовете на инициализационен вектор.

При декриптиране потребителят въвежда своята “master” парола, която заедно с байтовете на salt-та за ключа формират тайния ключ. После ключа и инициализационен вектор участват в декриптирането на получаването на паролата в явен вид.

## 4. Използвани технологии

За реализирането на приложението е използван езика Java SE, версия 17.0.1. За базата данни е използвана СУБД MariaDB версия 10.4.11, под формата на софтуера “XAMPP for Windows” версия 7.4.2. За изпълнение на програмата, тестовете, както и за набавянето на нужните библиотеки се използва инструментът за автоматизация Gradle, версия 7.3. За изграждане на графичния потребителски интерфейс е използвана софтуерната платформа JavaFX, версия 14.

Използвани са следните Java софтуерни библиотеки :

* “Gson” версия 2.8.9 за работа с JSON тестов формат за сериализация и обмен на данни
* “Mockito” версия 3.7.7 използвана при тестовете на класовете, използващи уеб услугите
* “MariaDB Java client” версия 2.1.2, конектор за осъществяване на връзка с базата данни
* “Apache Commons CLI” версия 1.5.0, за обработване на аргументите при стартиране на програмата
* “Junit” версия 4.13, за изпълнение на тестовете на програмни единици

Използвани са следните Gradle плъгини:

* “Gradle buildconfig plugin” версия 3.0.0 за създаване на сървърна конфигурация, съдържаща чувствителна информация, която не трябва да присъства в кода на програмата
* “OpenJFX JavaFX plugin” версия 0.0.8 за изтегляне и конфигуриране на модулите на JavaFX

Проектът е разделен на три главни части : сървър, клиент и API. С помощта на Gradle са дефинирани зависимостите от API проекта към сървъра и клиента. По този начин е възможно версиите на библиотеките да бъдат управлявани поотделно във всеки един от проектите, което води до по-стабилен и надежден процес на разработка и поддръжка на проекта.

## 5. Инсталация и настройки

Ще опишем инсталация под ОС Windows 10. Първо е необходимо да се инсталира JDK, като препоръчително е тя да е поне версия Java SE JDK 17 или по-нова. След това е нужно инсталирането на приложението XAMPP версия 7.4.2 или по-нова. След това разархивираме архива “oopJava2022\_fn4MI3400043\_FinalProject.zip ” с кода на проекта. В получената папка “ oopJava2022\_fn4MI3400043\_FinalProject” можем да видим директориите, които обозначат отделните части на проекта :

* директория server - сървър частта
* client - клиент частта
* api - споделената част между клиента и сървъра
* gradle - директория за файловете на Gradle системата
* docs – директория с документация и фигурите към нея

## 5.1 Конфигуриране на сървърната част

Портът и адресът на стартиране на сървъра са дефинирани в програмния код и по подразбиране това е адресът на локалната машина, localhost и порт 7777. Преди пускане на сървъра, първо трябва да бъдат създадена схемата на базата данни. Това може да стане чрез изпълнение на съдържанието на файла в сървърната поддиректория “config/db\_init.sql”, който създава базата и съдържа дефинициите на таблиците. В него е поместена и примерна команда за създаване на отделен потребител в СУБД, който да има права за промяна само върху създадената база от данни.

В директорията сървър трябва да се създаде файл gradle.properties, който да съдържа двойки име=стойност, които обозначават конфигурационните параметри за връзка с базата данни и използване на REST уеб услуги. Имената на променливите трябва да бъде спазен, тъй като те се използват за създаване на конфигурационен клас, който по време на компилация се достъпва от сървърната програма за осъществяване на връзка с базата данни и при отправяне на заявки към уеб услугите. Параметрите във файла са следните:

* HAPPY\_DEV\_AUTH\_KEY : стойност на API ключ за отправяне на заявки към услугата за генериране на пароли на адрес https://happi.dev/docs/password-generator
* ENZOIC\_AUTH\_KEY : стойност на API ключ за отправяне на заявки към услуга за проверка на сигурността на пароли на адрес https://www.enzoic.com/docs-passwords-api/
* DB\_URL : адрес на базата данни, включително и порт
* DB\_USER : име на потребител в СУБД с права за достъп до таблиците
* DB\_PASSWORD : парола на потребителя

Примерен формат на gradle.properties файл е

*HAPPY\_DEV\_AUTH\_KEY=*

*ENZOIC\_AUTH\_KEY=*

*DB\_URL=jdbc:mysql://localhost/password\_vault*

*DB\_USER=exampleuser*

*DB\_PASSWORD=exampleuser1234*

## 5.2 Конфигуриране на клиентската част

Портът и адресът на сървъра, към който се свързва клиента са дефинирани в програмния код на класа “Client”. По подразбиране това е адресът на локалната машина, localhost и порт 7777.

## 5.3 Изпълнение на приложението

Първо се стартира сървърната част. За препоръчване е използването на Gradle wrapper скрипта, който позволява изпълнение на команди, дори без инсталация на Gradle системата. Под Windows ОС това е скрипта gradlew.bat. Изпълнението на сървъра може да стане чрез отваряне на PowerShell терминал в директорията на проекта и изпълнение на следните команди :

* компилация на сървърния проект
  + .\gradlew :server:clean build
* изпълнение на тестовете
  + .\gradlew :server:test
* изпълнение на програмата, опциите не са задължителен параметър, по подразбиране програмата стартира с графичен потребителски интерфейс
  + изпълнение на програмата в конзолен режим
    - .\gradlew :server:run --args=”true”
  + изпълнение на програмата с графичен потребителски интерфейс
    - .\gradlew :server:run --args=”false”

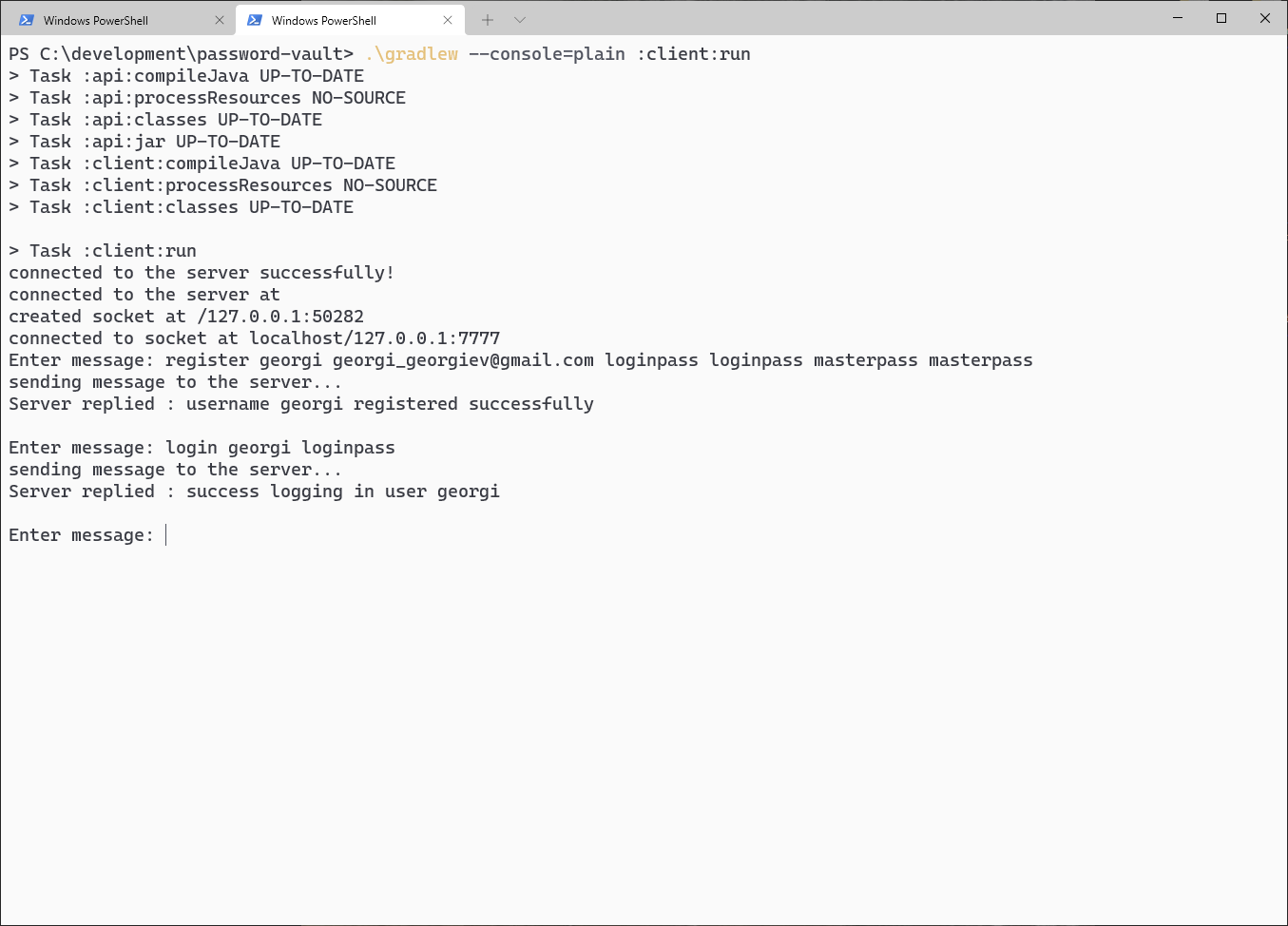
Аналогично изпълнение на клиент може да стане в друг PowerShell терминал в директорията на проекта и изпълнение на следните команди :

* компилация на клиентския проект
  + .\gradlew :client:clean build
* изпълнение на тестовете
  + .\gradlew :client:test
* изпълнение на програмата, опциите не са задължителен параметър, по подразбиране програмата стартира с графичен потребителски интерфейс
  + изпълнение на програмата в конзолен режим
    - .\gradlew :clientr:run --args=”-c=true” --console=plain
  + изпълнение на програмата с графичен потребителски интерфейс
    - .\gradlew :client:run --args=”-c=false”

## 6. Кратко ръководство на потребителя

## 6.1 Регистриране на акаунт и влизане в системата

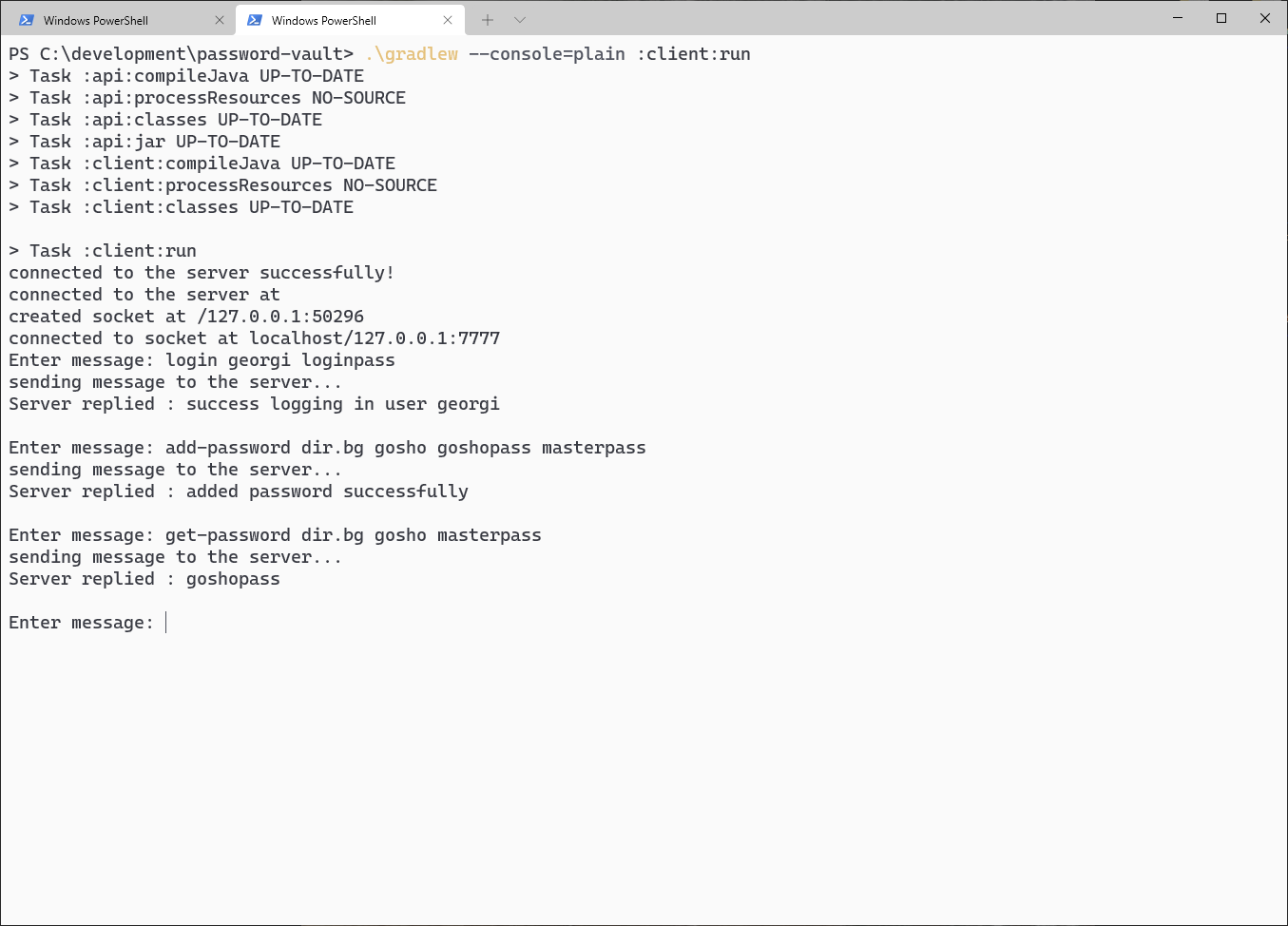
За успешна регистрация потребител трябва да въведе потребителско име, имейл, парола за влизане и “master” парола, която се използва при достъпване на запазените от него пароли.



Фигура 3. Команди за регистриране и вход в системата

## 6.2 Добавяне на парола за сайт

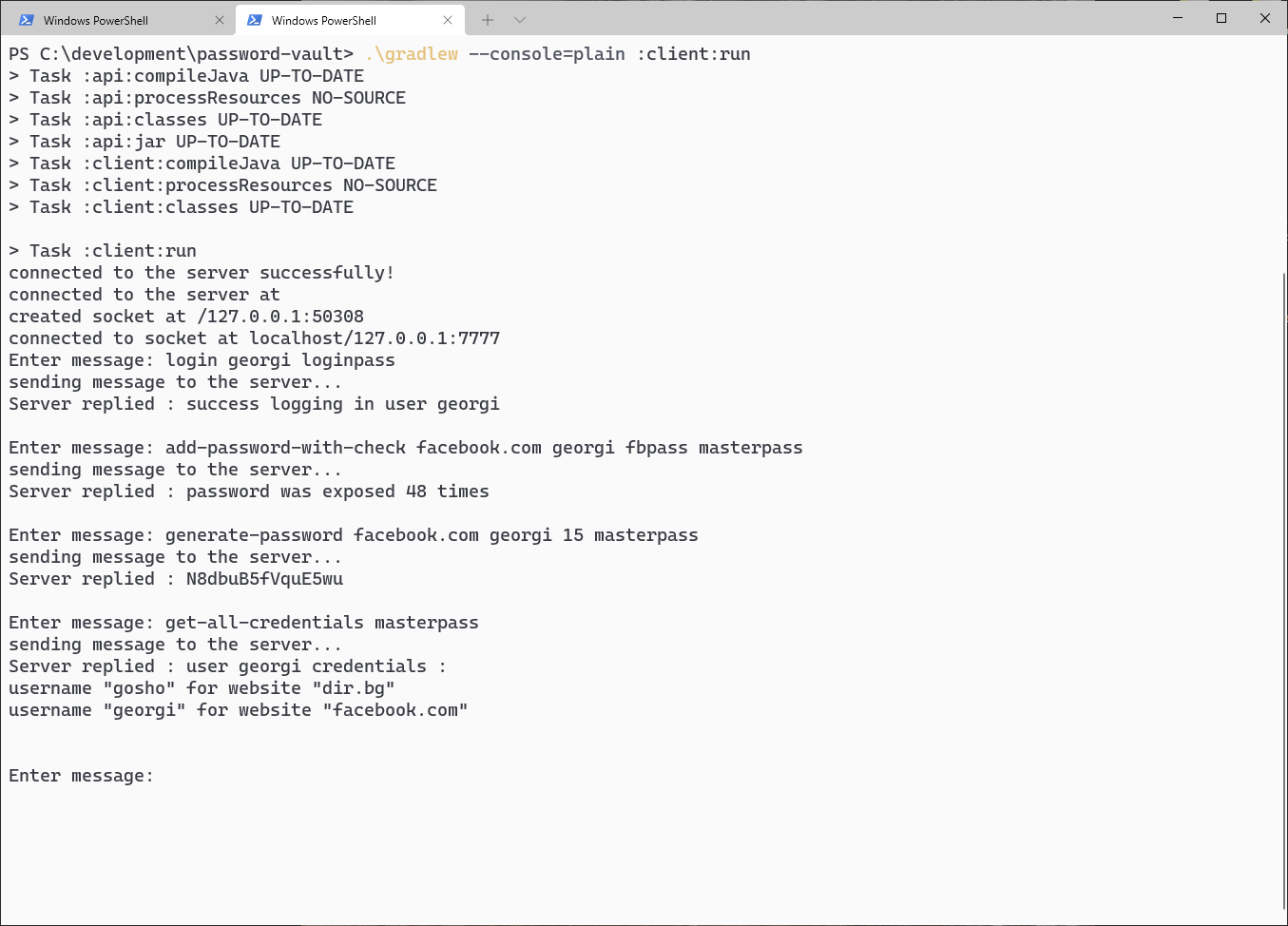
При добавяне на парола за сайт, която потребителя вече има и ползва е възможно той да я запише без да проверява нейната сигурност. Това става като подаде коректната команда с аргументи адреса на уеб сайта, потребителското име за сайта, паролата за сайта и своята “master” парола.



Фигура 4. Команди добавяне на парола и взимане на нейната стойност в явен вид

## 6.3 Добавяне на парола за сайт с проверка на нейната сигурност и генериране на парола

При добавяне на парола за сайт с проверка на сигурността в случай, че паролата е къса или често използвана, то системата уведомява потребителя и не записва несигурната парола. Тогава потребителя може да избере опцията системата да генерира и запише достатъчно сигурна парола, като той трябва да подаде като параметри адреса на уеб сайта, потребителското име за сайта, дължината на генерираната паролата за сайта и своята “master” парола.



Фигура 5. Добавяне на парола с проверка, генериране на парола и извличане на всички пароли

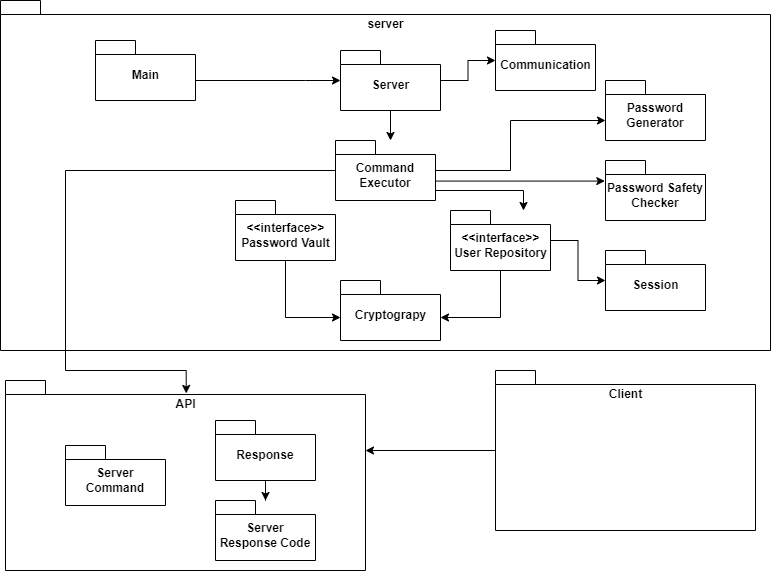
## 7. Примерни данни

Приложението работи с текстови команди, въведени от потребителя. Следните команди са примери и са използвани и при изготвянето на ръководството на потребителя.

* регистрация
  + register georgi [georgi\_georgiev@gmail.com](mailto:georgi_georgiev@gmail.com) loginpass loginpass masterpass masterpass
* вход в системата
  + login georgi loginpass
* показване на помощно меню
  + help
* добавяне на парола без проверка на сигурността
  + add-password dir.bg gosho goshopass masterpass
* добавяне на парола с проверка на сигурността
  + add-password-with-check facebook.com georgi dirpass masterpass
* генериране на парола с дължина 10 символа
  + generate-password facebook.com georgi 10 masterpass
* проверка на списъка с всички пароли
  + get-all-credentials masterpass
* проверка на парола за нейната сигурност
  + check-password unsafepass
* излизане от профила
  + logout
* прекратяване на връзката със сървъра
  + disconnect

## 8. Описание на програмния код

Проекта можем да разделим на три основни модула : сървър, клиент и API. Следната диаграма за употребата на модулите представя зависимостите между трите основни модула, както и подмодулите им.

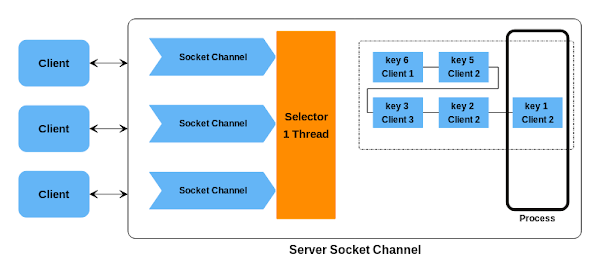


Фигура 6. Употреба на модулите

Както виждаме и на диаграмата клиентската и сървърната част имат зависимост от API модула, в който са дефинирани формата на заявка към сървъра и формата на неговия отговор.

Реализиран е еднонишков и неблокиращ сървър с класовете от Java NIO интерфейса, който позволява осъществяване на мрежова комуникация, поддържаща голям брой клиенти с максимално оползотворяване на системните ресурси. По-важните системни класове от този интерфейс са

* “Channel”: абстракция за комуникационна връзка, в нашия случай ни трябват класовете “ServerSocketChannel”, “SocketChannel”, които са съответно за сървърния сокет и конкретна клиентска връзка
* “Selector”: този компонент служи за ефективно определяне готовността на операцията, която дадена комуникационна връзка може да извърши, съответно приемане на други комуникационни връзки, прочитане или записване на данни
* “Buffer”: съхранява данни, които се прочитат от или записват в дадена комуникационна връзка

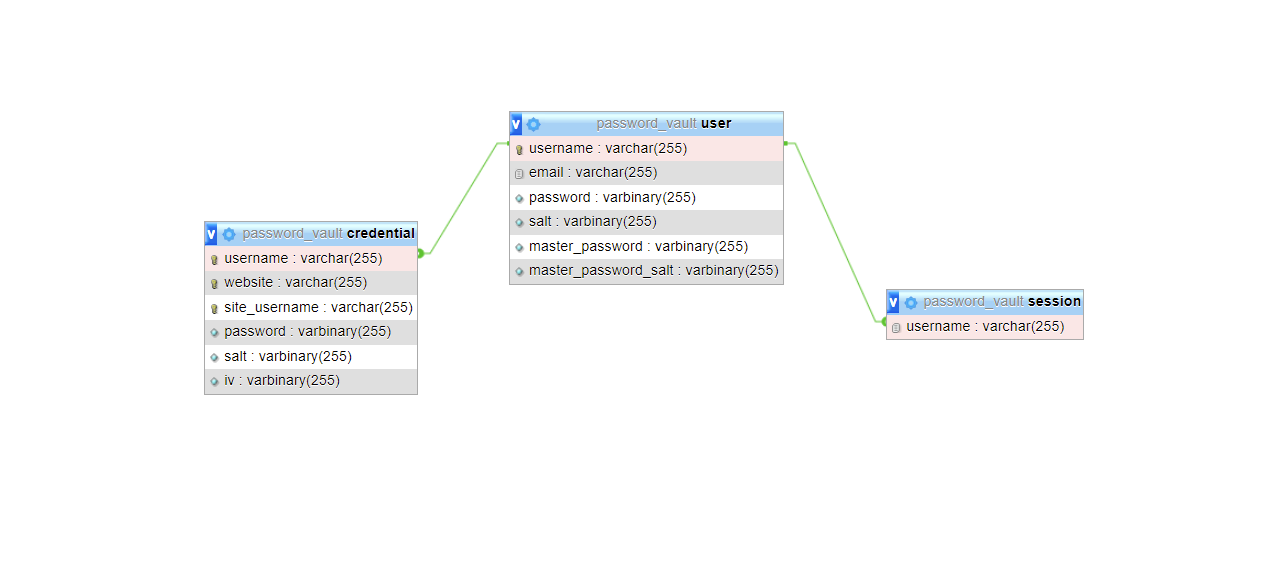


Фигура 7. Архитектура на Java NIO сървър

По-значимите модули в сървърната част са:

* “Main”: инициализира сървърния модул, всички негови зависимости и го стартира
* “Server”: представлява сървърния процес, който приема заявки, обработва ги чрез извикване на модула “Command Executor” и връща отговора към съответния клиент.
* “Command Executor”: определя вида на заявката и при нужда извиква съответния подмодул, който обработва. След това отговора се връща на “Server”, който го изпраща към клиента.
* “User Repository”: грижи се за регистрирането и автентикацията на потребителите в системата. Използва модула за хеширане и съхранение на потребителска сесия.
* “Password Vault”: реализира логиката по съхранението и извличането на пароли, които потребителите желаят да запишат за своите услуги. Използва модула за криптиране.
* “Password Generator”: грижи се за отправяне на заявки към уеб услугата за генериране на пароли. Използва “HttpClient” класа на Java.
* “Password Safety Checker”: грижи се за отправяне на заявки към уеб услугата за проверка сигурността на дадена парола. Използва “HttpClient” класа на Java.
* “Cryptography”: реализира конкретните криптографските алгоритми за сигурно хеширане и симетрично криптиране на текст. Използва класовете дефинирани в Java Cryptography Architecture, които се предлагат от стандартната версия на Java (Java SE).

За съхранение на данните е използвана релационна база от данни, като основните таблици представляват потребител, данните за удостоверение в уеб услуга(credential) и потребителска сесия.



Фигура 8. Схема на базата данни

## 9. Приноси на студента, ограничения и възможности за бъдещо разширение

Възможни разширения на функционалността на приложението може са

* добавяне на графичен-потребителски интерфейс на клиентското приложение
* създаване на администраторско приложение с графичен-потребителски интерфейс за следене на системата
* възможност потребител да експортира/импортира записаните пароли
* възможност за автоматично попълване на форма със съответната парола и копиране на парола на clipboard-а на потребителя
* възможност за съхранение на друг вид удостоверителна информация като API ключове

## 10. Какво научих

С направата на този проект научих как да се имплементират добрите практики за съхранение на чувствителна информация със средствата на криптографията, работа с база данни чрез Java JDBC интерфейса, реализиране на ефективна мрежова комуникация с Java NIO интерфейса и използване на Mockito framework за изолирано тестване на програмните единици.

## 11. Използвани източници

[1] "Java NIO", Ron Hitchens, O'Reilly Media 2002

[2] "Pro Java 7 Nio.2", Anghel Leonard, Apress 2011

[3] "Hashing a Password in Java", Eugen Paraschiv, 12.01.2022, https://www.baeldung.com/java-password-hashing

[4] "Java AES Encryption and Decryption", Eugen Paraschiv, 14.11.2021, https://www.baeldung.com/java-aes-encryption-decryption

[5] "Java AES encryption and decryption", Yong Mook Kim, 02.06.2020, линк: https://mkyong.com/java/java-aes-encryption-and-decryption/

[6] "Cryptography 101 for Java developers by Michel Schudel", Michel Schudel,

линк: <https://github.com/MichelSchudel/crypto-demo>, видео презентация: https://www.youtube.com/watch?v=1925zmDP\_BY

[7] "Java Cryptography Architecture (JCA) Reference Guide", линк: https://docs.oracle.com/en/java/javase/17/security/java-cryptography-architecture-jca-reference-guide.html

[8] Услуга за проверка на сигурността на пароли, линк: https://www.enzoic.com/docs-passwords-api/

[9] Услуга за генериране на пароли, линк: https://happi.dev/docs/password-generator

[10] "Exploring the New HTTP Client in Java", Eugen Paraschiv, 31.12.2021, линк: https://www.baeldung.com/java-9-http-client

[11] "Java 11 HttpClient Examples", Yong Mook Kim, 17.05.2020, линк: https://mkyong.com/java/java-11-httpclient-examples/

[12] "Sharing Build Logic between Subprojects", https://docs.gradle.org/current/userguide/sharing\_build\_logic\_between\_subprojects.html

[13] "Declaring Dependencies between Subprojects", <https://docs.gradle.org/current/userguide/declaring_dependencies_between_subprojects.html>

[14] Фигура 1. линк: https://www.okta.com/blog/2019/03/what-are-salted-passwords-and-password-hashing/

[15] Фигура 2. линк: <https://sectigostore.com/blog/5-differences-between-symmetric-vs-asymmetric-encryption/>

[16] Фигура 7. линк: https://dzone.com/articles/java-io-and-nio