Wykonała:
Maryna Savchenko

Web site in Django which provides with some cryptographic functions from java library. Connected via RabbitMQ.

https://github.com/msavchen/CryptographyWebApplication

To run:

Server
Import cryptoAppServer as Maven project, JRE 1.8 and above is required.
Run 'cryptoAppServer\src\main\java\com\rabbitmq\cryptoAppServer\RPCServer.java'

Client
RabbitMQ Service, Django are required.
Go to CryptographyWebClient and run 'python manage.py runserver'.
Open home page: http://127.0.0.1:8000/cryptoApp/

- **lista funkcjonalności:**
  - *Generate key;*

    After a user provides input, a client sends a request to the server to generate a key with a chosen algorithm and size. After the server answers - client shows generated key.

  - *Hash text;*

    After a user provides input, a client sends a request to the server to hash a text with a chosen algorithm. After the server answers - client shows hashed text.

  - *MAC (Message authentication code);*

    After a user provides input, a client sends a request to the server to generate a key with a chosen algorithm and size. After this server sends a request to create mac using a specified algorithm and created key. After the server answers - client shows Message authentication code and generated key.

  - *Encrypt text;*

    After a user provides input, a client sends a request to the server to generate a key with a chosen algorithm and size. After this server sends a request to encode text using a

specified algorithm and created key. After the server answers - client shows encrypted text and generated key.

- *Encrypt file;*

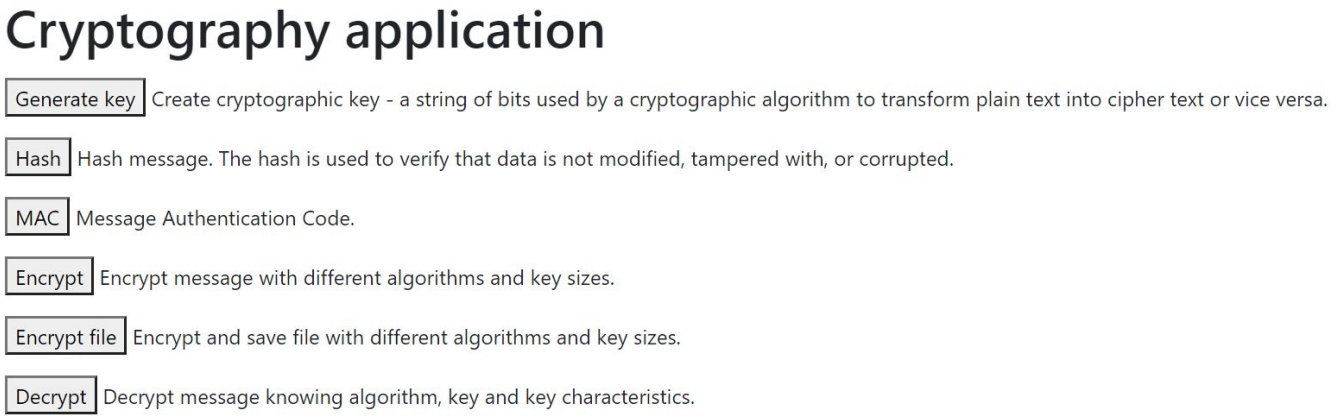
  After a user uploads a file, a client sends a request to the server to generate a key with a chosen algorithm and size. After this server sends a request to encode a file using a specified algorithm and created key. After the server answers - client shows encrypted text and generated key. Optionally the user can save the file to the folder encryptedFiles in CryptographyWebClient folder.

- *Decrypt text - doesn't work properly.*

  After a user provides input, a client sends a request to the server to create a key object with a chosen algorithm and key text. After this server sends a request to decode text using a specified algorithm and generated key. After the server answers - client shows decrypted.

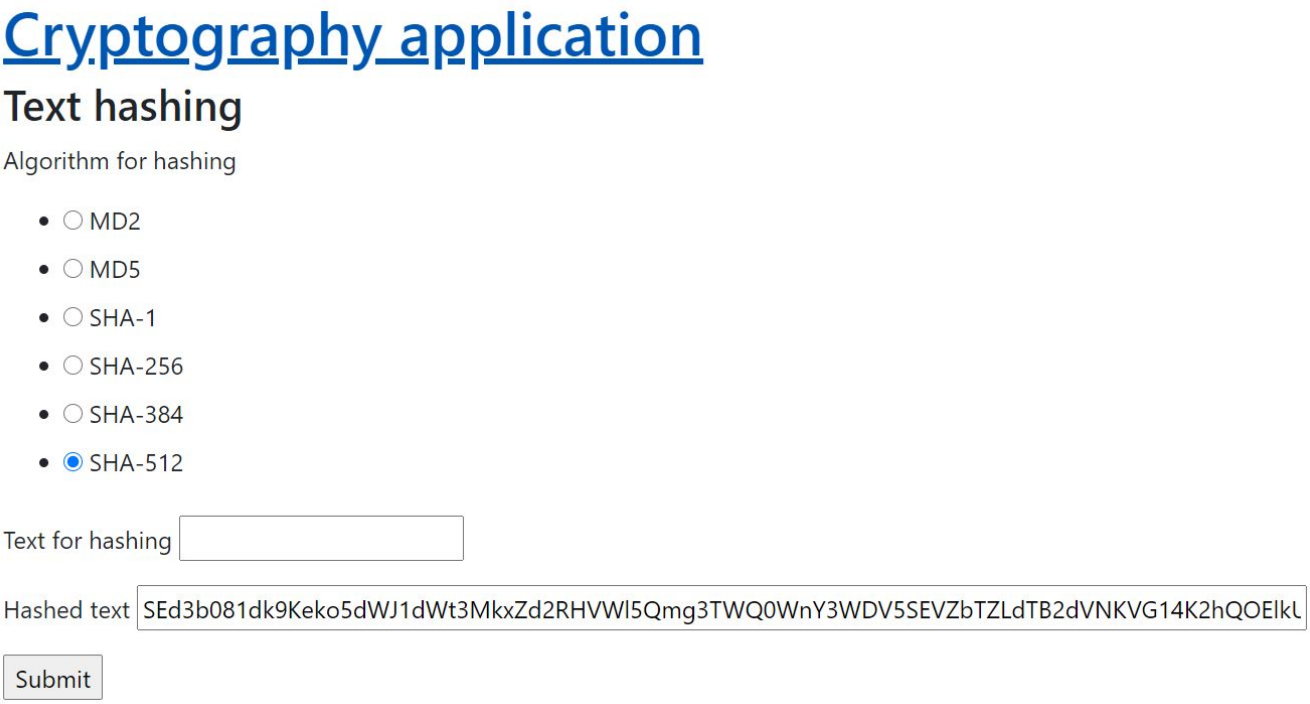
- **projekt interfejsu użytkownika:**

  - Home page:



- Generate key

- Hash



- MAC

Cryptography application

Message authentication code

Algorithm for key

- (•) AES
- ( ) DES
- ( ) DESede
- ( ) HmacSHA1
- ( ) HmacSHA256

Key size 128

Algorithm for text ecryption

- ( ) HmacMD5
- ( ) HmacSHA1
- (•) HmacSHA256

Text to encrypt

Submit

- Encrypt

Cryptography application

Messages encryption

Algorithm for key

- (•) AES
- ( ) DES
- ( ) DESede
- ( ) HmacSHA1
- ( ) HmacSHA256

Key size 128

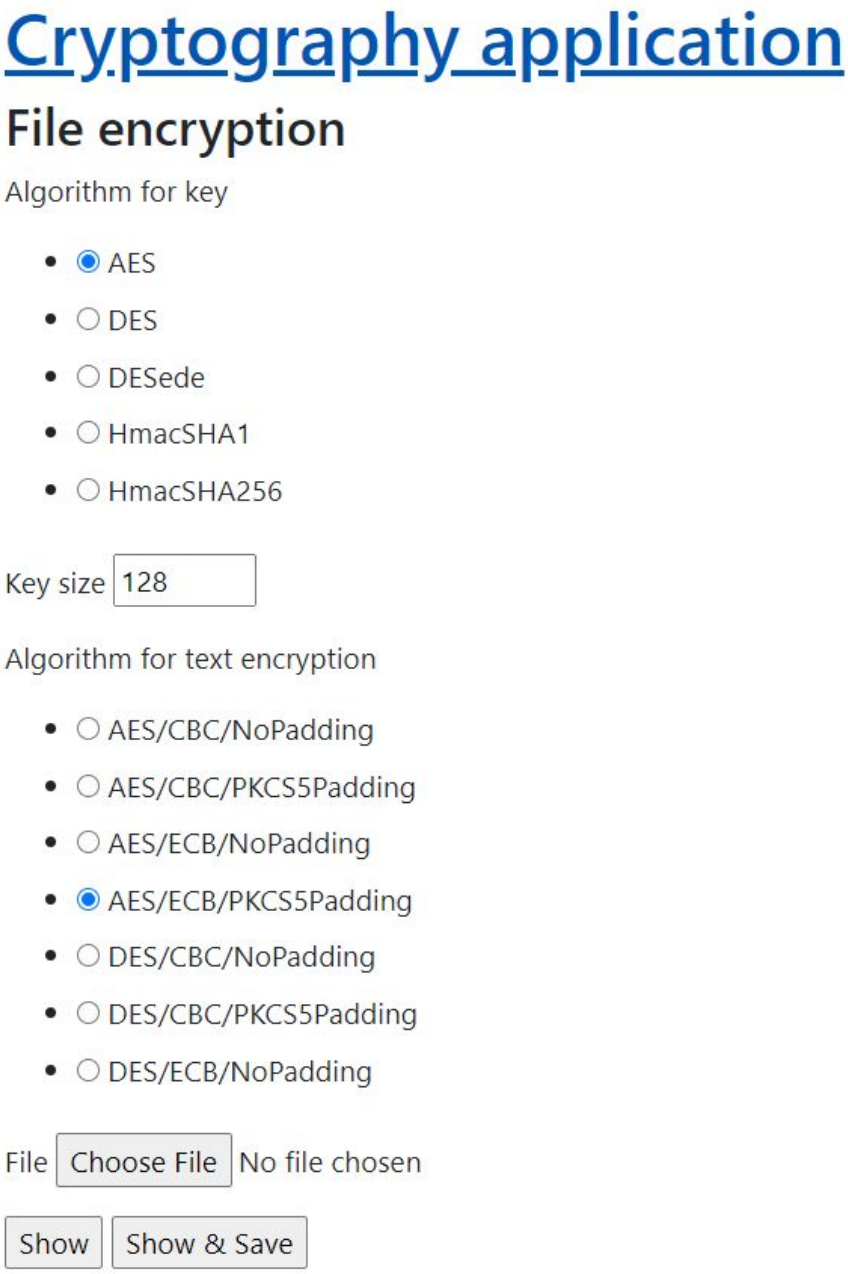
Algorithm for text encryption

- ( ) AES/CBC/NoPadding
- ( ) AES/CBC/PKCS5Padding
- ( ) AES/ECB/NoPadding
- (•) AES/ECB/PKCS5Padding
- ( ) DES/CBC/NoPadding
- ( ) DES/CBC/PKCS5Padding
- ( ) DES/ECB/NoPadding

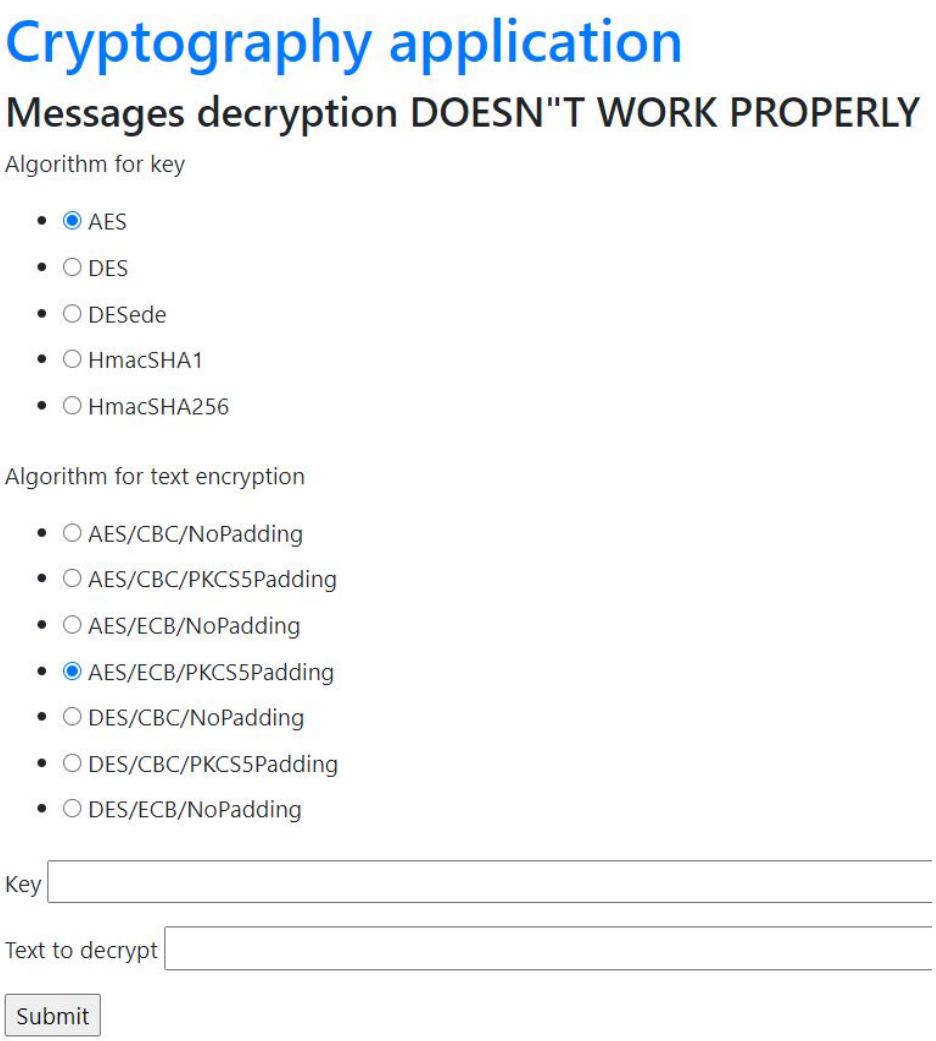
Text to encrypt

Submit

- File encryption



- Text decryption



- **projekt architektury:**
  - Java server which provides some basic functionality from Package javax.crypto;
  - Django client which gets input and tasks from a user and calls for Java server to perform them;

- **specyfikacja protokołu komunikacyjnego:**
  To communicate between Java Server and Python client RabbitMQ is used. Pattern Remote procedure call (RPC) is implemented for this purpose. The client sends a request and waits for a response, the server gets a basic acknowledge.