TÉCNICO LISBOA

BlingBank

MSc. Computer Science and Engineering

Network and Computer Security

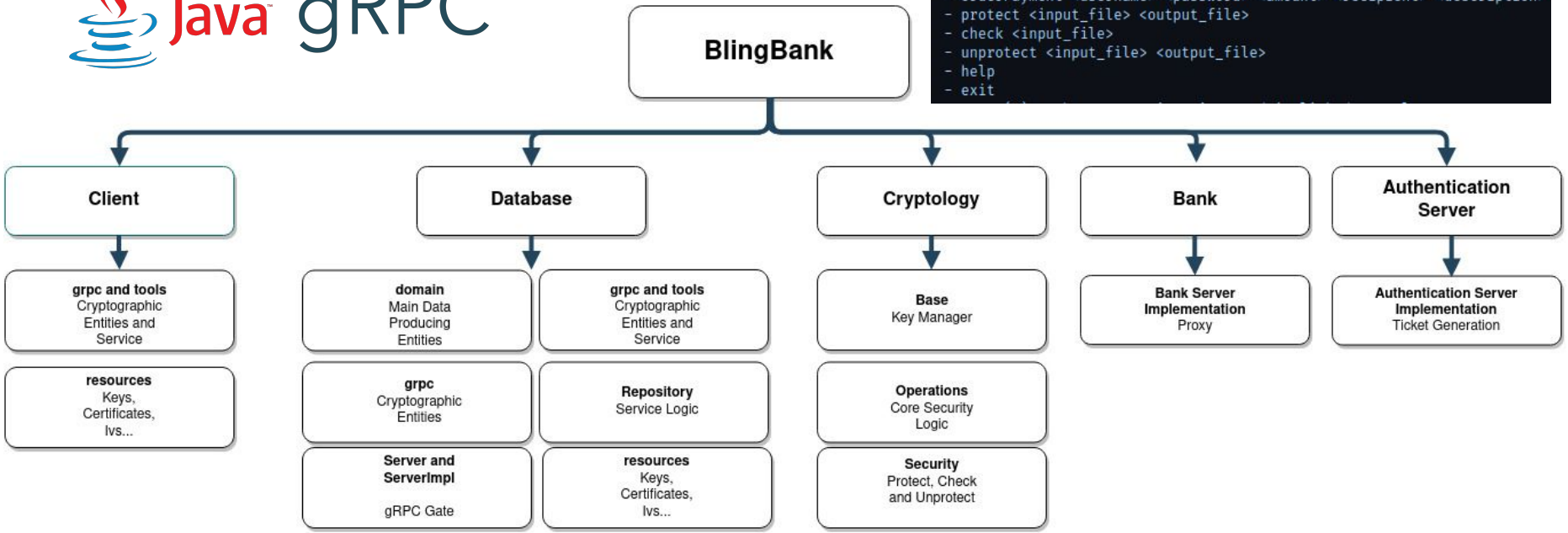
Winter Semester 2023/2024

Alameda 28
Alexandre Coelho 100120
Pedro Lameiras 99540

Built Infrastructure

Software Used


```
Usage:
- createAccount (<username> <password>)*
- deleteAccount <username> <password>
- balance <username> <password>
- getMovements <username> <password>
- addExpense <username> <password> <amount> <description>
- orderPayment <username> <password> <amount> <recipient> <description>
- protect <input_file> <output_file>
- check <input_file>
- unprotect <input_file> <output_file>
- help
- exit
```



Secure Document Format

TÉCNICO LISBOA

DatabaseServerImpl.java

```java
public void orderPayment(OrderPaymentRequest request, StreamObserver<OrderPaymentResponse> responseObserver) {
    try {
        if (isDebug()) System.out.println("\tDatabaseServerImpl: order payment");

        if (!crypto.check(request)) throw new TamperedMessageException();
        JsonObject requestJson = Utils.deserializeJson(crypto.decrypt(request).getRequest().toByteArray());
        String username = requestJson.getString(name:"username");
        byte[] password = crypto.decryptPassword(requestJson.getString(name:"password"));
        LocalDateTime date = LocalDateTime.parse(requestJson.getString(name:"date"));
        BigDecimal amount = new BigDecimal(requestJson.getString(name:"amount"));
        String description = requestJson.getString(name:"description");
        String recipient = requestJson.getString(name:"recipient");
        OffsetDateTime timestamp = OffsetDateTime.parse(requestJson.getString(name:"timestampString"));

        if (isDebug()) System.out.printf("\t\tUsername: %s\n\t\tPassword (Hex): %s\n\t\tRecipient

        databaseManager.orderPayment(username, password, date, amount, description, recipient, ti

        responseObserver.onNext(crypto.encrypt(OrderPaymentResponse.newBuilder().build()));
        responseObserver.onCompleted();
        if (isDebug()) System.out.println("\tDatabaseServerImpl: order payment successful");
```

UserService.java

```java
public void getMovements(String username, String password, String timestampString) {
    try {
        if (debug) System.out.println("\tUserService: encoding show expenses request");
        byte[] requestJson = Utils.serializeJson(
            Utils.createJson(
                List.of("username", "password", "timestampString"),
                List.of(username, crypto.encryptPassword(password), timestampString)
        ));

        if (debug) System.out.println("\tUserService: making rpc");

        GetMovementsResponse getAccountMovementsResponse = bankingServiceStub.getMovements(crypto.encrypt(
            GetMovementsRequest.newBuilder()
            .setRequest(
                ByteString.copyFrom(requestJson)
            ).build())
        );

        if (!crypto.check(getAccountMovementsResponse)) throw new RuntimeException("Message contents were tampered with");

        if (debug) System.out.println("\tUserService: processing get account movements response");
        JsonObject responseJson = Utils.deserializeJson(crypto.decrypt(getAccountMovementsResponse).getResponse().toByteArr
        for(int i = 0; i < responseJson.getJsonArray(name:"movements").size(); i++) {
            JsonObject movement = responseJson.getJsonArray(name:"movements").getJsonObject(i);
            System.out.printf("Movement %d\n\tCurrency: %s\n\tDate: %s\n\tValue: %s\n\tDescription: %s\n", i + 1, movement.
```

```java
public final class Base {

    public interface KeyManager {…

    public interface CryptographicCore {
        default boolean check(byte[] input, String secretKeyPath, String publicKeyPath, String ivPath)
                throws Exception {
            return Security.check(input, Base.readSecretKey(secretKeyPath), Base.readPublicKey(publicKeyPath), Base.readIv(ivPath));
        }

        default JsonObject decrypt(byte[] input, String secretKeyPath, String ivPath)    The type JsonObject from module javax.json may
                throws Exception {
            return Utils.deserializeJson(Security.unprotect(input, Base.readSecretKey(secretKeyPath), Base.readIv(ivPath)));
        }

        default byte[] encrypt(byte[] input, String secretKeyPath, String privateKeyPath, String ivPath)
                throws Exception {
            return Security.protect(input, Base.readSecretKey(secretKeyPath), Base.readPrivateKey(privateKeyPath), Base.readIv(ivPath));
        }
```

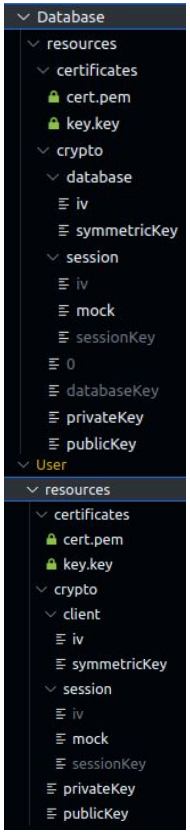
Secure Document Format

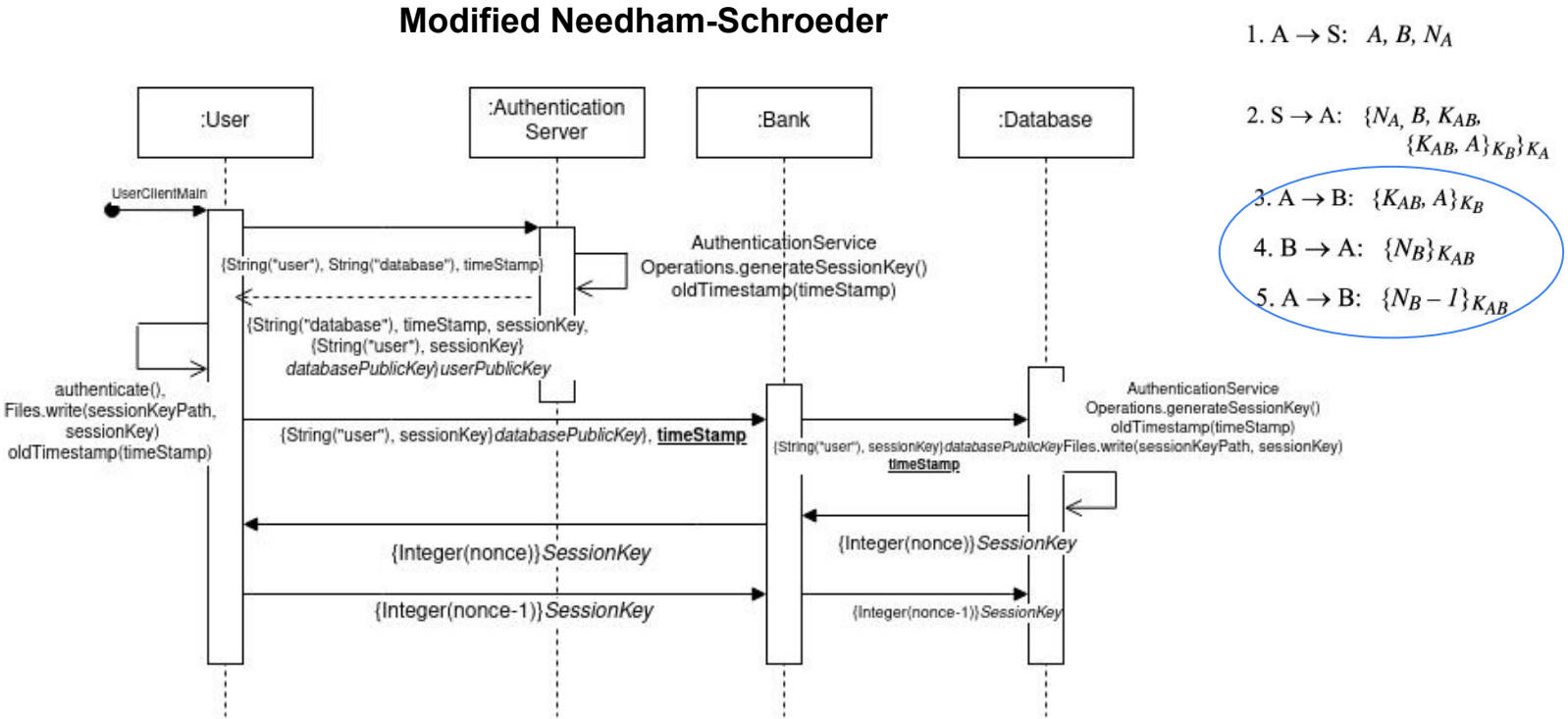
TÉCNICO LISBOA

```java
public final class Security {

    public static byte[] protect(byte[] message, SecretKey secretKey, PrivateKey privateKey, byte[] iv)
            throws SignatureException, InvalidKeyException, NoSuchAlgorithmException,
            InvalidAlgorithmParameterException, NoSuchPaddingException, IllegalBlockSizeException, BadPaddingException {
        byte[] signature = Operations.messageSignature(privateKey, message);

        byte[] protectedDocument = new byte[signature.length + message.length];
        System.arraycopy(signature, 0, protectedDocument, 0, signature.length);
        System.arraycopy(message, 0, protectedDocument, signature.length, message.length);

        return Operations.encryptData(secretKey, protectedDocument, iv);
    }

    public static boolean check(byte[] cryptogram, SecretKey secretKey, PublicKey publicKey, byte[] iv)
            throws NoSuchPaddingException, SignatureException, NoSuchAlgorithmException, InvalidKeyException,
            IllegalBlockSizeException, BadPaddingException, InvalidAlgorithmParameterException {
        byte[] protectedDocument = Operations.decryptData(secretKey, cryptogram, iv);
        byte[] signature = Arrays.copyOfRange(protectedDocument, 0, 256);
        byte[] message = Arrays.copyOfRange(protectedDocument, 256, protectedDocument.length);

        return Operations.messageValidation(publicKey, message, signature);
    }

    public static byte[] unprotect(byte[] cryptogram, SecretKey secretKey, byte[] iv)
            throws NoSuchPaddingException, NoSuchAlgorithmException, InvalidKeyException,
            InvalidAlgorithmParameterException, IllegalBlockSizeException, BadPaddingException {
        byte[] protectedDocument = Operations.decryptData(secretKey, cryptogram, iv);
        byte[] message = Arrays.copyOfRange(protectedDocument, 256, protectedDocument.length);
        byte[] document = new byte[message.length];
        System.arraycopy(message, 0, document, 0, message.length);
        return document;
    }

}
```

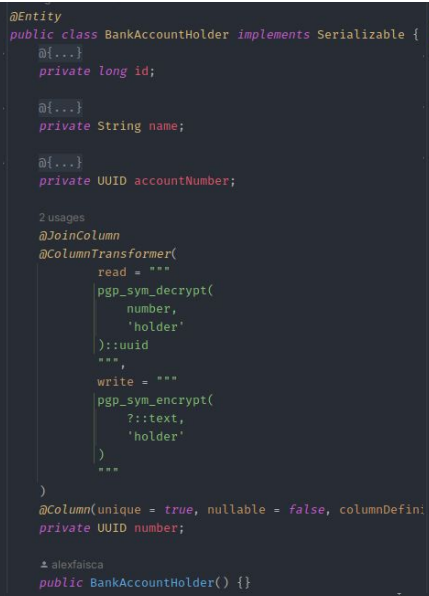
```java
public final class Operations {

    public static byte[] encryptData(SecretKey secretKey, byte[] message, byte[] iv)…

    public static byte[] decryptData(SecretKey secretKey, byte[] cipherText, byte[] iv)…

    public static byte[] hash(byte[] message) throws NoSuchAlgorithmException {…

    public static byte[] messageSignature(PrivateKey privateKey, byte[] message)…

    public static boolean messageValidation(PublicKey publicKey, byte[] message, byte[] messageSignature)

    public static byte[] generateSessionKey() {…

    public static byte[] generateIV(Integer id, byte[] secretKey, String secret) {…
}
```

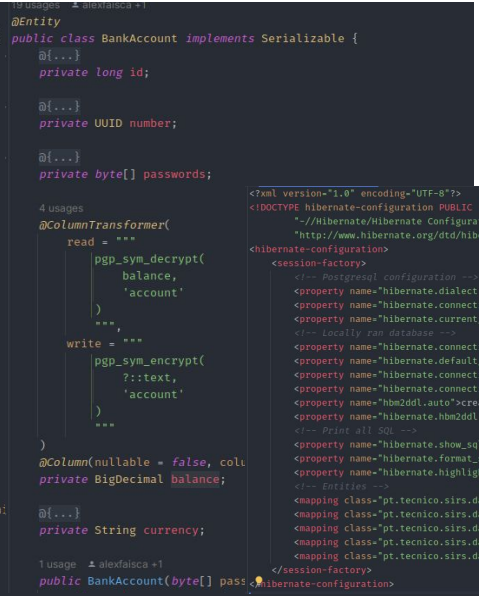
Initial Key Distribution

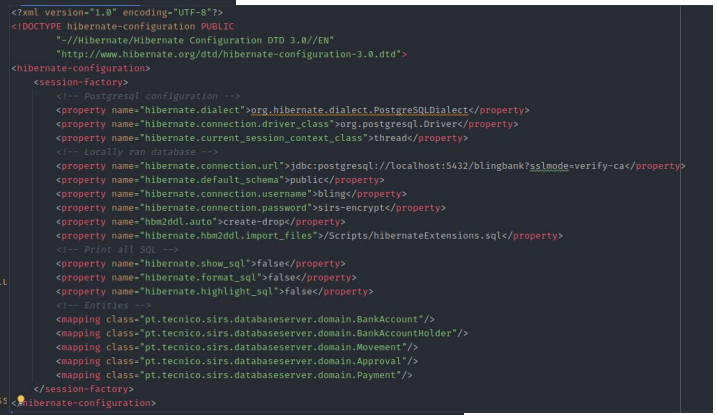


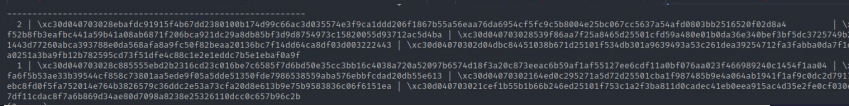


TÉCNICO LISBOA



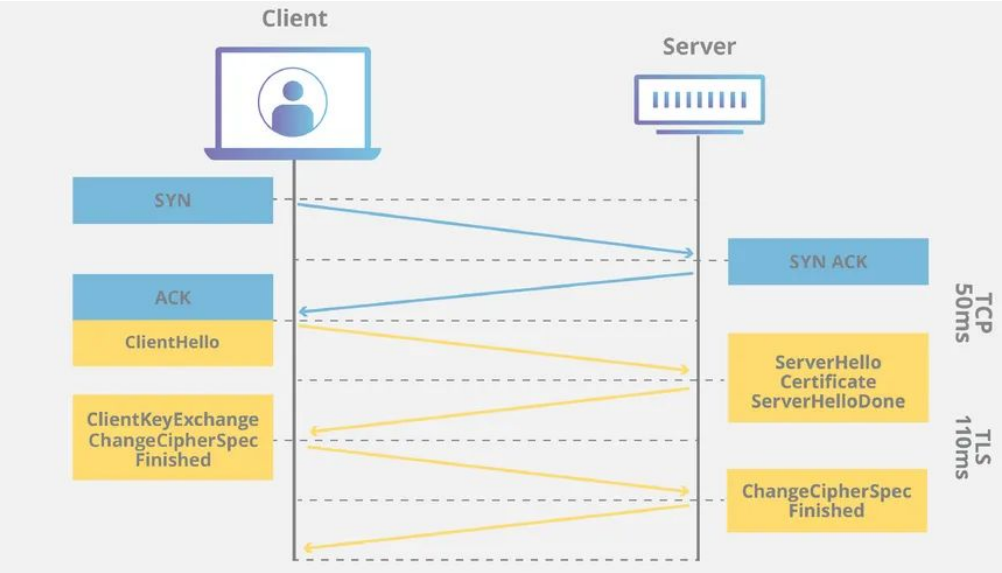






Database Encryption

Configured Secure Channels - TLS

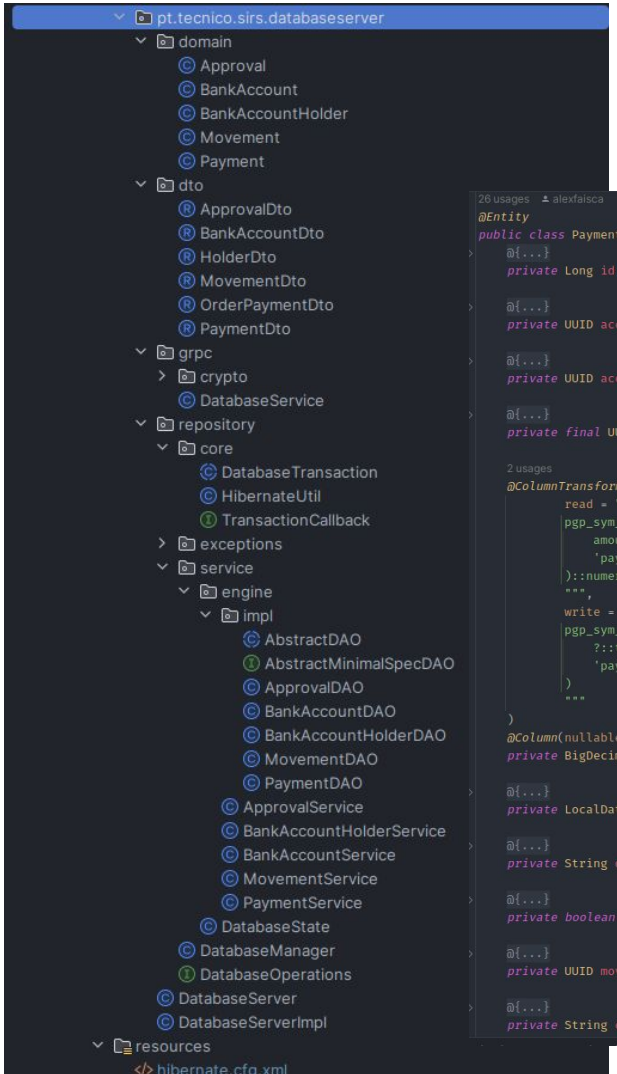


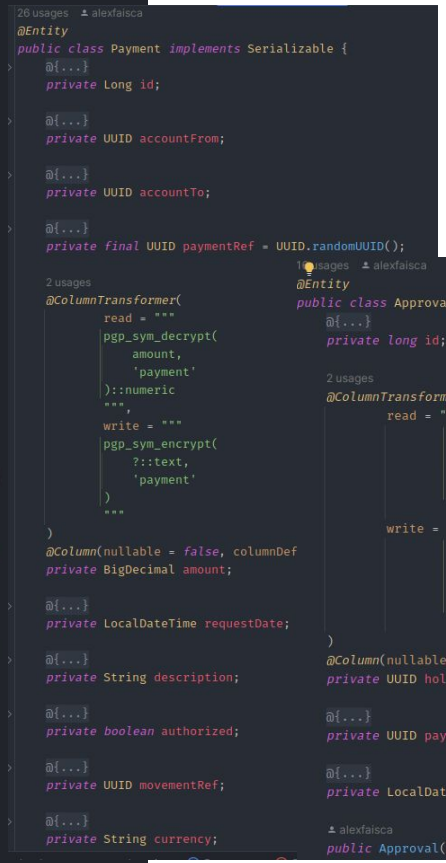

```
this.server = Grpc.newServerBuilderForPort(
        databasePort,
        TlsServerCredentials.newBuilder().keyManager(new File(args.get(9)), new File(args.get(10))).build()
).addService(new DatabaseServerImpl<>(state, cryptoCore, debug)).build();
```

```
1 usage  alexfaisca +1
public UserServiceBuilder(
        String host,
        Integer port,
        String authenticationServerAddress,
        Integer authenticationServerPort,
        String ivPath,
        String secretKeyPath,
        String publicKeyPath,
        String privateKeyPath,
        String trustCertCollectionPath,
        boolean debug
) throws Exception {
    this.debug = debug;
    this.host = host;
    this.port = port;
    this.authenticationServerAddress = authenticationServerAddress;
    this.authenticationServerPort = authenticationServerPort;
    this.crypto = new BankingClientCryptographicManager(
            ivPath,
            secretKeyPath,
            publicKeyPath,
            privateKeyPath
    );
    this.credentials = TlsChannelCredentials.newBuilder()
            .trustManager(new File(trustCertCollectionPath))
            .build();
}

1 usage  Pedro Lameiras +1
public UserService build() {
    this.authenticationServerChannel = Grpc.newChannelBuilderForAddress(
                    this.authenticationServerAddress,
                    this.authenticationServerPort,
                    this.credentials
    ).build();

    this.bankChannel = Grpc.newChannelBuilderForAddress(
                    this.host,
                    this.port,
                    this.credentials
    ).build();

    return new UserService( builder: this);
}
```

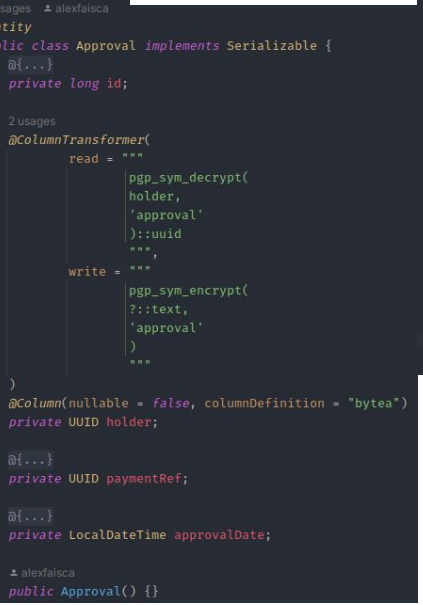
Security Challenge

TÉCNICO LISBOA

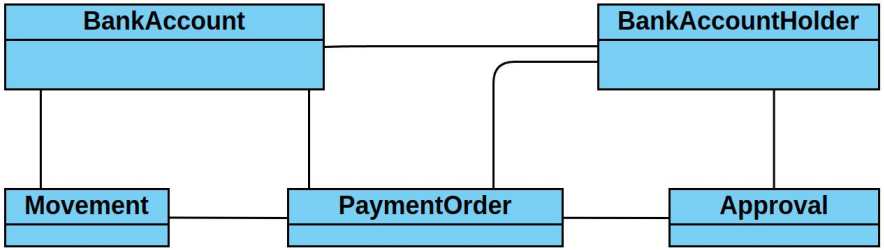




```
26 usages  alexfaisca
@Entity
public class Payment implements Serializable {
    @{...}
    private Long id;

    @{...}
    private UUID accountFrom;

    @{...}
    private UUID accountTo;

    @{...}
    private final UUID paymentRef = UUID.randomUUID();

    2 usages
    @ColumnTransformer(
            read = """
            pgp_sym_decrypt(
                amount,
                'payment'
            )::numeric
            """,
            write = """
            pgp_sym_encrypt(
                ?::text,
                'payment'
            )
            """
    )
    @Column(nullable = false, columnDef
    private BigDecimal amount;

    @{...}
    private LocalDateTime requestDate;

    @{...}
    private String description;

    @{...}
    private boolean authorized;

    @{...}
    private UUID movementRef;

    @{...}
    private String currency;
```



```
1 usages  alexfaisca
@Entity
public class Approval implements Serializable {
    @{...}
    private long id;

    2 usages
    @ColumnTransformer(
            read = """
                    pgp_sym_decrypt(
                    holder,
                    'approval'
                    )::uuid
                    """,
            write = """
                    pgp_sym_encrypt(
                    ?::text,
                    'approval'
                    )
                    """
    )
    @Column(nullable = false, columnDefinition = "bytea")
    private UUID holder;

    @{...}
    private UUID paymentRef;

    @{...}
    private LocalDateTime approvalDate;

    alexfaisca
    public Approval() {}
```



```
private void addPaymentApproval(Payment payment, HolderDto holderDto, LocalDateTime date) {
    ApprovalDto ignored = approvalService.addPaymentApproval(payment.getPaymentRef(), holderDto.number(), date);
}

1 usage  alexfaisca
private boolean checkPaymentApproval(Payment payment) {
    HashSet<UUID> approvals = new HashSet<>(
            approvalService.getPaymentApprovals(payment.getPaymentRef())
                    .stream().map(ApprovalDto::holder).toList()
    );
    return approvals.containsAll(
            holderService.getHolderByAccountNumber(payment.getAccountFrom())
                    .stream().map(HolderDto::number).toList()
    );
}

1 usage  alexfaisca
private void approvePayment(Payment payment, LocalDateTime date) {
    MovementDto movementDto = movementService.addPayment(
        new OrderPaymentDto(
            payment.getAccountFrom(),
            payment.getAccountTo(),
            date,
            payment.getAmount(),
            payment.getDescription(),
            payment.getCurrency()
    ));
    payment.setMovementRef(movementDto.movementRef());
    payment.setAuthorized(true);
}
```

Conclusion

- Overall, our work assured the client's data security needs
- Maven's security features were the main development bottleneck

```
#!/bin/bash

set -x

firewallAddress="10.0.2.2"
firewallPort="8000"
bankAddress="192.168.0.2"
bankPort="22"
databaseAddress="192.168.1.2"
databasePort="22"

# Configure adapters
sudo ip addr add 192.168.0.1/24 dev enp0s3
sudo ip link set dev enp0s3 up

sudo ip addr add 192.168.1.1/24 dev enp0s8
sudo ip link set dev enp0s8 up

# Allow the system to act as router
sudo sysctl net.ipv4.ip_forward=1

# Reload the Network Manager service
sudo /etc/init.d/network-manager force-reload

# Flush all existing rules
sudo ./flush_rules.sh

# Allow forwarding of packets that are part of an already established connection
sudo iptables -I FORWARD  -m state --state RELATED,ESTABLISHED -j ACCEPT
sudo iptables -I INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT

# SSH rate limit
sudo iptables -A INPUT -p tcp --dport 8000 -m state --state NEW -m recent --set
sudo iptables -A INPUT -p tcp --dport 8000 -m state --state NEW -m recent --update --seconds 10 --hitcount 5 -j DROP

# Redirect external connections to the bank
sudo iptables -t nat -A PREROUTING -i enp0s9 --dst 10.0.2.2 -p tcp --dport 8000 -j DNAT --to-destination 192.168.0.2:23

# Bank -> Database (Careful)
sudo iptables -A FORWARD -i enp0s3 -p tcp --sport 23: --dport 23 -m state --state NEW --source 192.168.0.2 -d 192.168.1.2 -j ACCEPT
```

```
<plugin>
            <groupId>org.codehaus.mojo</groupId>
            <artifactId>exec-maven-plugin</artifactId>
            <version>3.0.0</version>
            <executions>
                <execution>
                    <id>exec-start-script</id>
                    <phase>validate</phase>
                    <goals>
                        <goal>exec</goal>
                    </goals>
                    <configuration>
                        <executable>Scripts/init.sh</executable>
                        <arguments>
                            <argument>database</argument>
                        </arguments>
                    </configuration>
                </execution>
            </executions>
        </plugin>
    <plugin>
            <groupId>org.codehaus.mojo</groupId>
            <artifactId>exec-maven-plugin</artifactId>
            <version>3.0.0</version>
            <executions>
                <execution>
                    <id>exec-end-script</id>
                    <phase>install</phase>
                    <goals>
                        <goal>exec</goal>
                    </goals>
                    <configuration>
                        <executable>terminate.sh</executable>
                        <arguments>
                            <argument>database</argument>
                        </arguments>
                    </configuration>
                </execution>
            </executions>
        </plugin>

    </build>

</project>
```

TÉCNICO LISBOA

Live Demo