Portfolio:

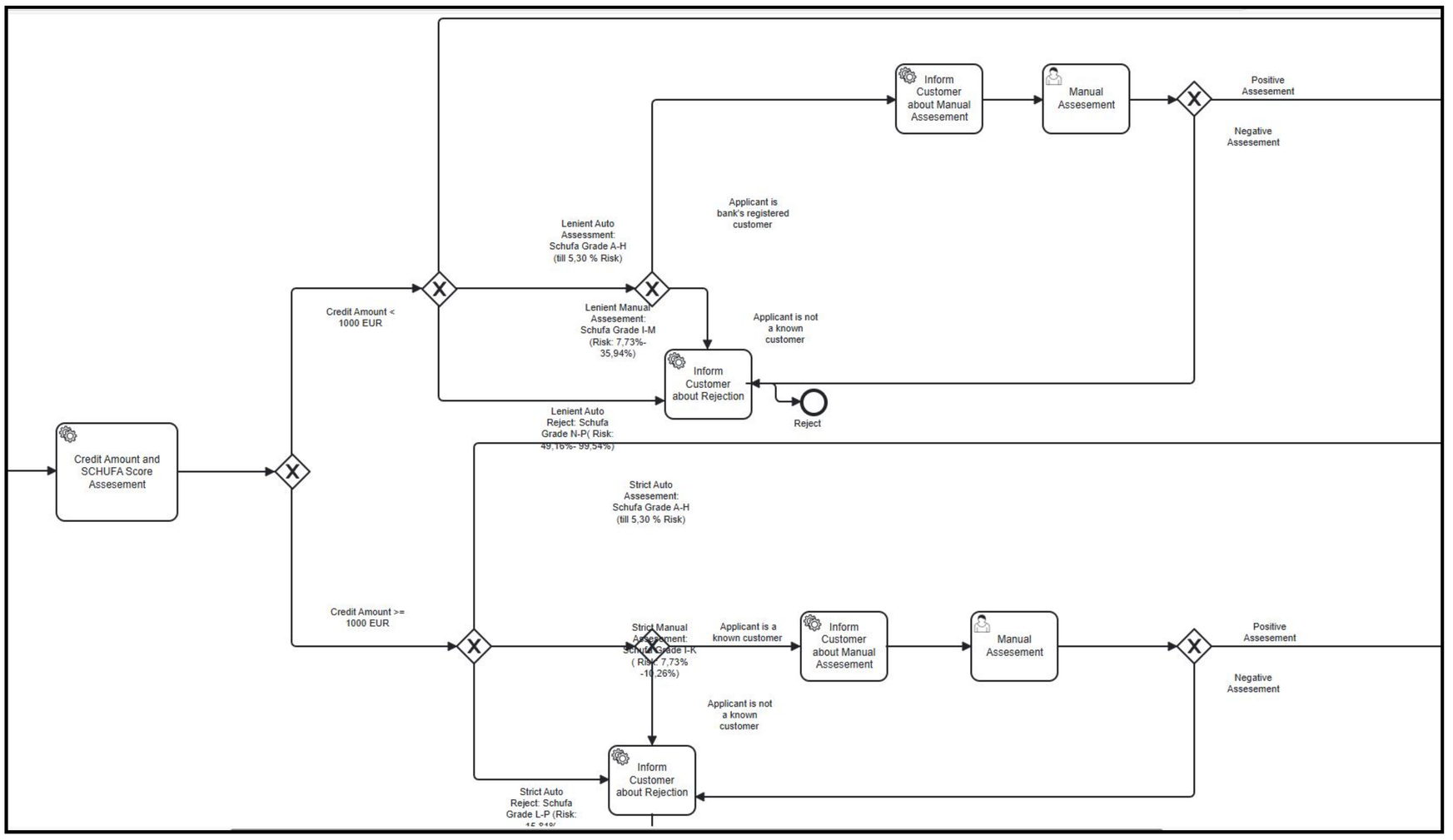
Spring Boot Learning Project:

Credit Assessment Backend Platform with Flowable BPMN Automation Engine

Kaan Bora Karapınar

A look at the current phase of project

Github Repo for better code review:
KaanBoraKarapinar/CreditAssesementApplication



A part of BPMN Schema

Online marketplaces often offer **microcredits** to customers at checkout, while banks typically extend these credits to a customer base about whom they lack detailed information. To address this, I designed a credit application schema using the BPMN framework and implemented it in Flowable, an open-source automation engine (based on Activiti, similar to Camunda).

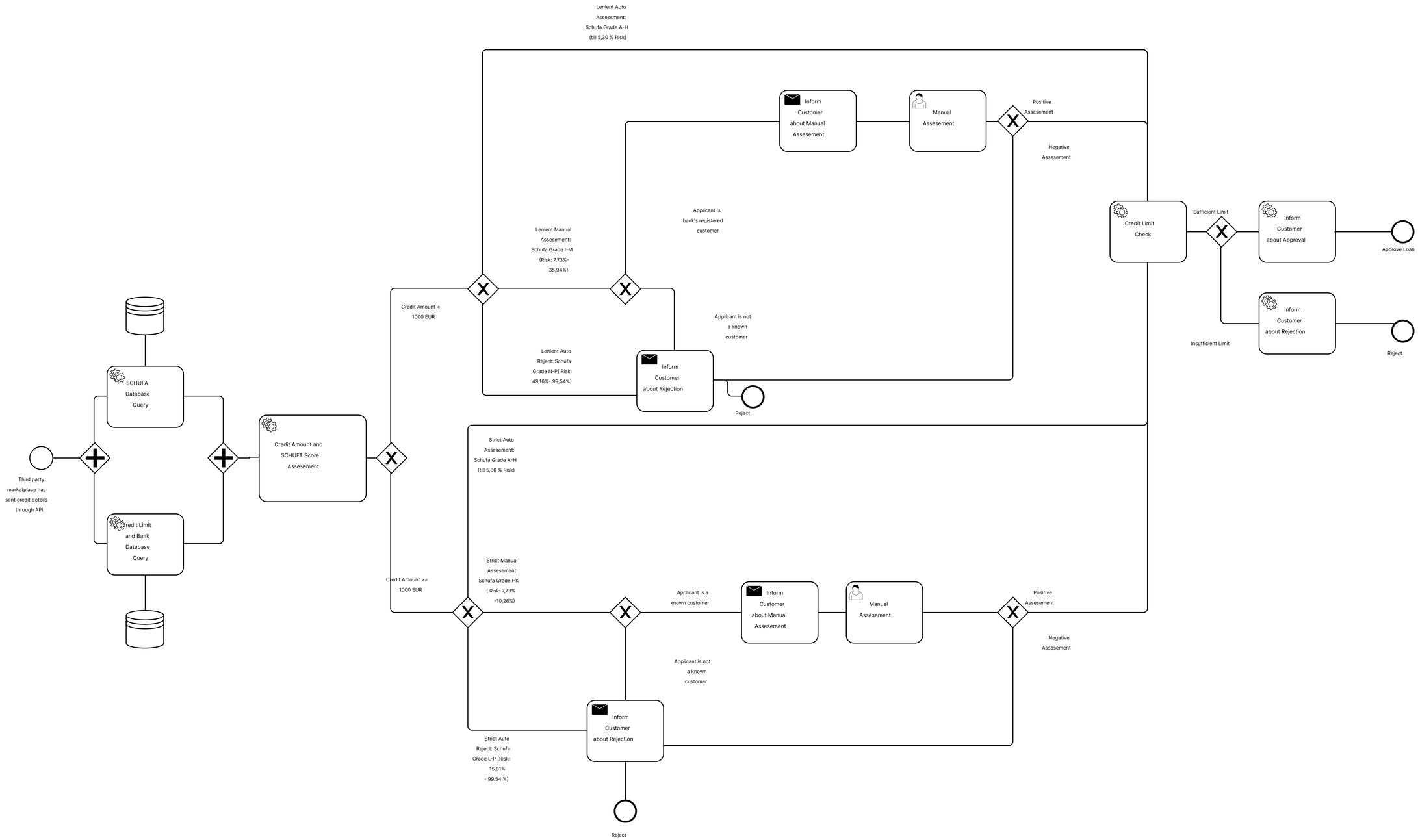
My goal was to explore **Spring Boo**t, a widely-used framework, and **Flowable** because **BPMN engines** like Flowable (after implemented correctly) enable **colleagues without technical expertise** to interact with system variables, **define directly integrable business processes, update them, and audit them** with the version-specific structures and logic.

I am still adding modules to further explore the depths of the Spring framework.

In current phase of my application:

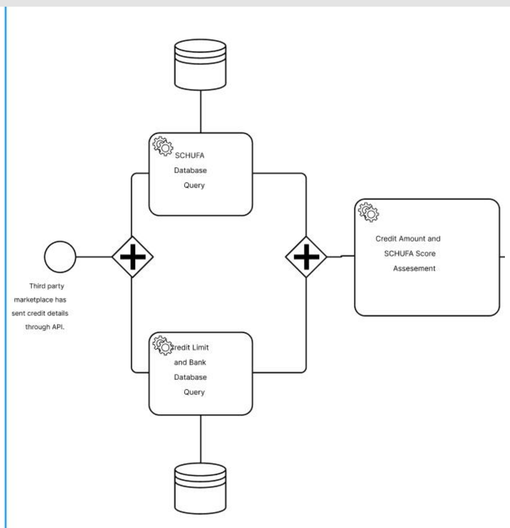
- Bank internal data and Schufa data are gathered (randomly determined in the implementation, as I don't have access to real data).
- These data are saved in a database using JPA.
- The JPA entity classes' object variables are accessed via Flowable's support for Jakarta Common Expression Language ensuring **no shallow copies are created, thus avoiding discrepancies.**

Complete BPMN Schema

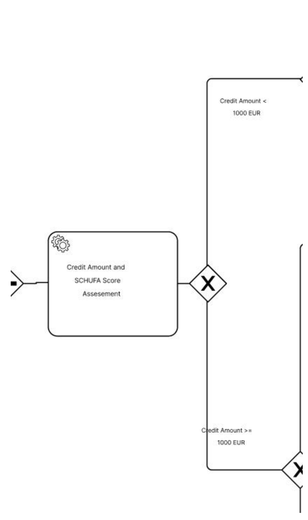


BPMN Schema

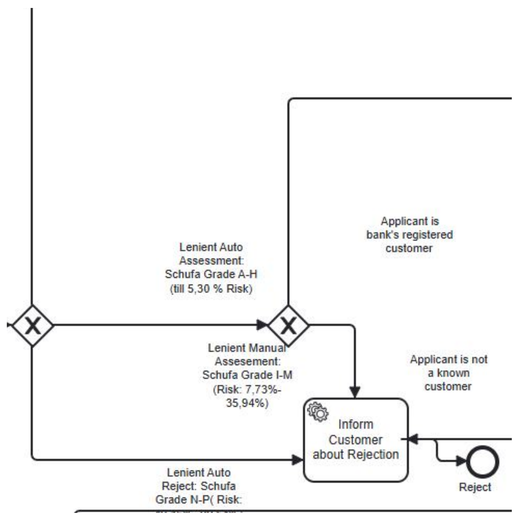
Since my primary focus was on learning how BPMN engines and Spring Boot work, and I didn’t have access to real-life credit data, **this process is oversimplified.** If tasked with implementing a real risk assessment schema, I would also consider other critical metrics such as age, monthly income, employment status (including length and type), educational background, number of accounts, DTI ratio (Debt-to-Income) in our and SCHUFA's systems, insurance policies in our and SCHUFA's systems, product type, shopping behavior in the originating marketplace, and other partner marketplaces etc.



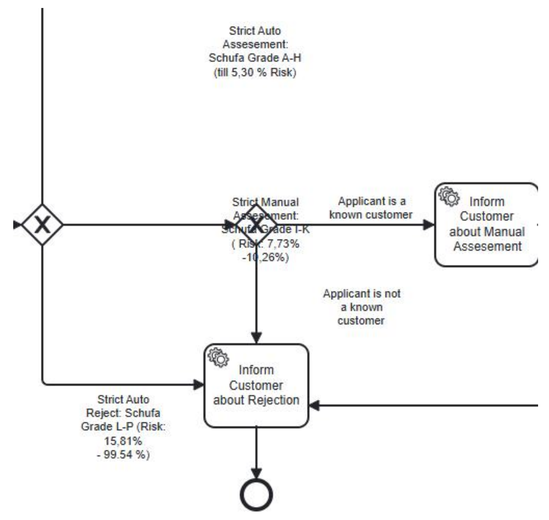
- The assessment system receives an API call for credit assessment. It retrieves the SCHUFA score via an API call and communicates with the bank's mainframe for additional details stored in the database, primarily to determine if the applicant is a known customer.



- For credits less than 1000 EUR, the "Lenient Path" is triggered; otherwise, the system follows the "Strict Path."



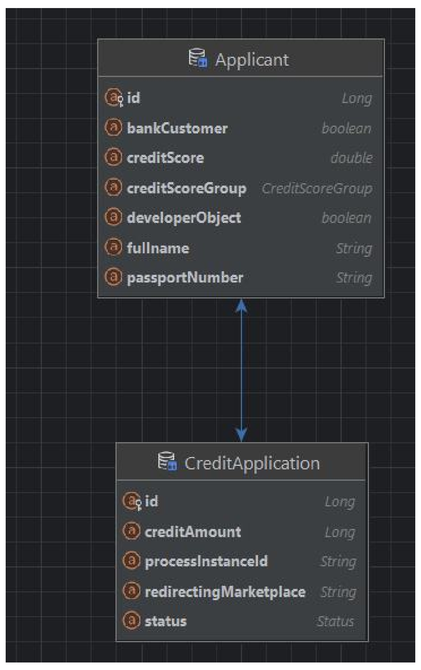
- Lenient Path:
- SCHUFA A-H (risk ≤ 5.30%): Approve, if there is sufficient credit limit
- SCHUFA N-P (risk 49.16%-99.54%): Reject
- For anything in between, if customer is a customer of the bank the credit application is manually controlled. Otherwise it is rejected

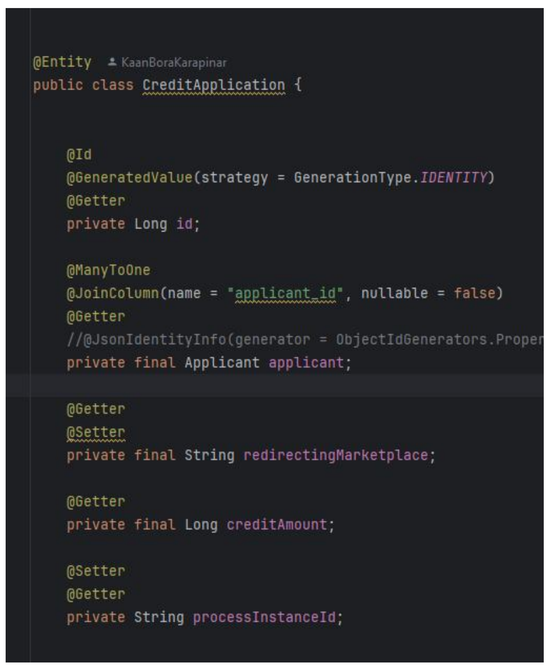


- SCHUFA A-H (risk ≤ 5.30%): Approve. if there is sufficient credit limit
- SCHUFA L-P (risk 15.81%-99.54%): Reject.
- For anything in between, if customer is a customer of the bank the credit application is manually controlled. Otherwise it is rejected

- For credits over 1000 EUR, stricter criteria apply. If the applicant is a known customer, a specialist evaluates and decides on the credit status.

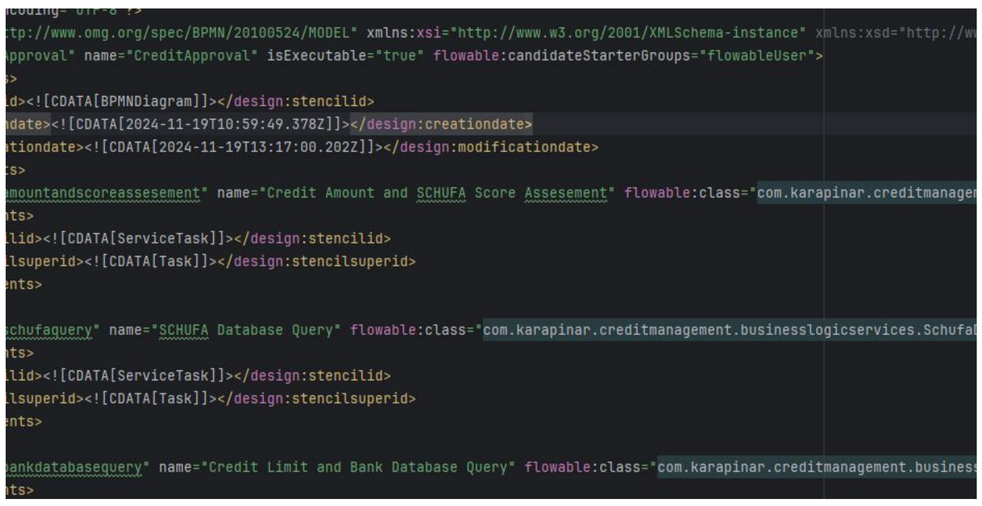
Snippets from source code





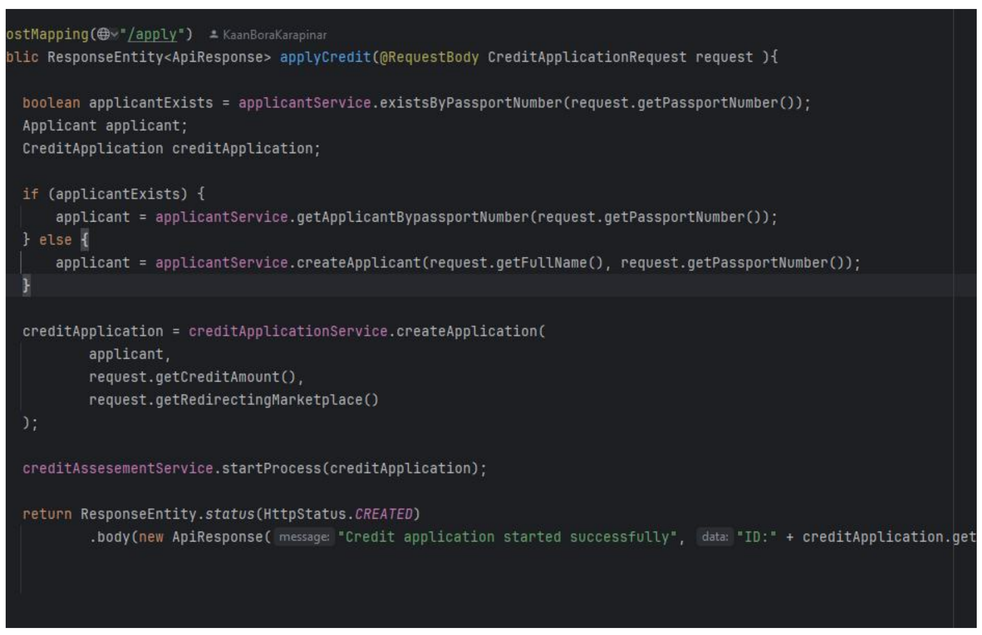
```
@Entity  KaanBoraKarapinar
public class CreditApplication {


    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Getter
    private Long id;

    @ManyToOne
    @JoinColumn(name = "applicant_id", nullable = false)
    @Getter
    //@JsonIdentityInfo(generator = ObjectIdGenerators.Proper
    private final Applicant applicant;

    @Getter
    @Setter
    private final String redirectingMarketplace;

    @Getter
    private final Long creditAmount;

    @Setter
    @Getter
    private String processInstanceId;
```

- Data is managed using JPA, simplifying database operations with object–relational mapping.



```
ctp://www.omg.org/spec/BPMN/20100524/MODEL" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://ww
pproval" name="CreditApproval" isExecutable="true" flowable:candidateStarterGroups="flowableUser">
id><![CDATA[BPMNDiagram]]></design:stencilid>
date><![CDATA[2024-11-19T10:59:49.378Z]]></design:creationdate>
tiondate><![CDATA[2024-11-19T13:17:00.202Z]]></design:modificationdate>
mountandscoreassesement" name="Credit Amount and SCHUFA Score Assesement" flowable:class="com.karapinar.creditmanage
lid><![CDATA[ServiceTask]]></design:stencilid>
lsuperid><![CDATA[Task]]></design:stencilsuperid>

chufaquery" name="SCHUFA Database Query" flowable:class="com.karapinar.creditmanagement.businesslogicservices.Schufa
lid><![CDATA[ServiceTask]]></design:stencilid>
lsuperid><![CDATA[Task]]></design:stencilsuperid>

ankdatabasequery" name="Credit Limit and Bank Database Query" flowable:class="com.karapinar.creditmanagement.busines
```

- BPMN Diagram and Business Logic is stored in an XML file



```
ostMapping("/apply")  KaanBoraKarapinar
blic ResponseEntity<ApiResponse> applyCredit(@RequestBody CreditApplicationRequest request ){

  boolean applicantExists = applicantService.existsByPassportNumber(request.getPassportNumber());
  Applicant applicant;
  CreditApplication creditApplication;

  if (applicantExists) {
      applicant = applicantService.getApplicantBypassportNumber(request.getPassportNumber());
  } else {
      applicant = applicantService.createApplicant(request.getFullName(), request.getPassportNumber());
  }

  creditApplication = creditApplicationService.createApplication(
          applicant,
          request.getCreditAmount(),
          request.getRedirectingMarketplace()
  );

  creditAssesementService.startProcess(creditApplication);

  return ResponseEntity.status(HttpStatus.CREATED)
          .body(new ApiResponse( message: "Credit application started successfully",  data: "ID:" + creditApplication.get
```

- Third party marketplace calls the API (managed by CreditApplicationController)

```
@Autowired
private RuntimeService runtimeService;

@Autowired
private TaskService taskService;

@Transactional  2 usages  KaanBoraKarapinar
public void startProcess(CreditApplication creditApplication) {
   Map<String, Object> variables = new HashMap<>();
    variables.put("creditApplicationJavaObject", creditApplication);

    runtimeService.startProcessInstanceByKey( processDefinitionKey: "creditApproval", variables);

}
```

- All necessary objects are created, and CreditAssessmentService activates Flowable with a CreditApplicationObject.

Snippets from source code

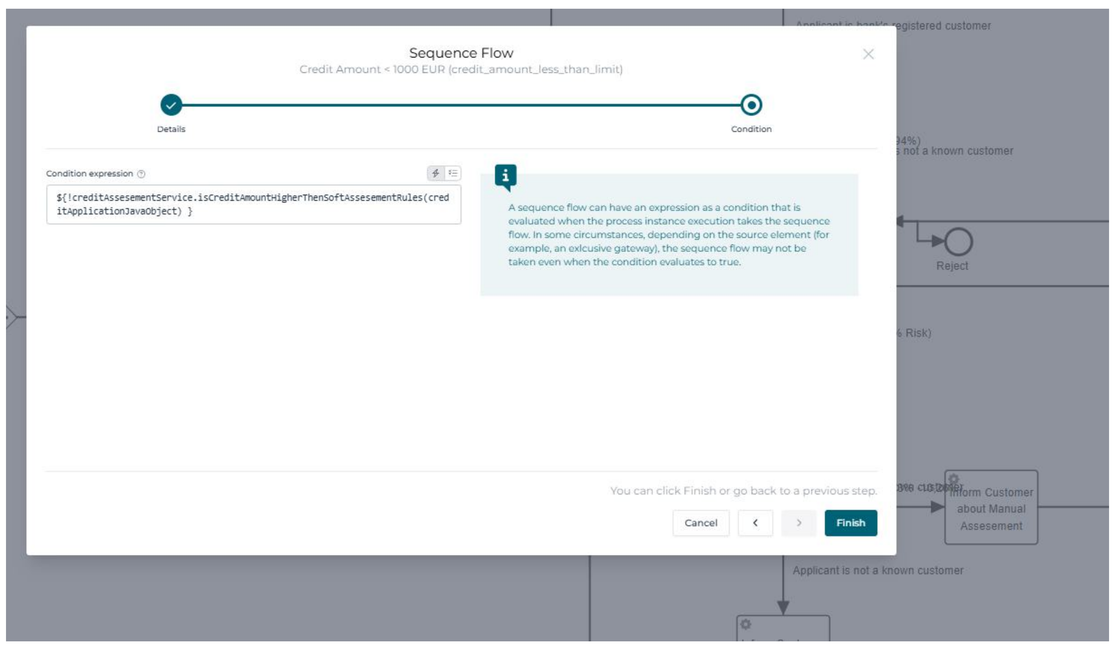
```
<sequenceFlow id="bpmnSequenceFlow_39" name="Schufa Grade I-M (Risk: 7,73%-35,94%)" sourceRef="bpmnGateway_30" targetRef="b
  <extensionElements>
    <design:stencilid><![CDATA[SequenceFlow]]></design:stencilid>
    <design:display_ref_in_diagram><![CDATA[true]]></design:display_ref_in_diagram>
  </extensionElements>
  <conditionExpression xsi:type="tFormalExpression"><![CDATA[${creditAssesementService.
isCreditScoreSufficientForManualCheckForSoftAssesement
(creditApplicationJavaObject) }]]></conditionExpression>
</sequenceFlow>
```

```
public boolean isCreditAmountHigherThenSoftAssesementRules(CreditApplication creditApplication){  2 usages  KaanBoraKarapinar

    final int limit = 1000;

    return creditApplication.getCreditAmount() >= 1000;

}


public boolean isCreditScoreSufficient(CreditApplication creditApplication,  6 usages  KaanBoraKarapinar
                                       CreditScoreGroup lowerBorder,
                                       CreditScoreGroup upperBorder) {
    return CreditScoreGroup.isBetween(creditApplication.getApplicant().getCreditScoreGroup(), lowerBorder, upperBorder);
}

//** soft assesement, when under the limit
public boolean isCreditScoreSufficientForAutoApproveForSoftAssesement(CreditApplication creditApplication) {  1 usage  KaanBo
    return isCreditScoreSufficient(creditApplication, CreditScoreGroup.A, CreditScoreGroup.H);
}
```

- Gateways
  call CreditAssessmentService methods with the CreditApplicationObject as a parameter (Jakarta EL) using to evaluate if the application and applicant meet the criteria to proceed.

Gateways can manually access API data using Jakarta EL, for example, expressions like CreditApplication.creditAmount < 1000 or CreditApplication.Applicant.creditScore > 1000.

Alternatively, they can use an internal method returning a boolean, such as CreditApplicationService.isCreditScoreSufficientForManualCheckForStrictAssessment. This enables non-technical employees to use a more user-friendly interface for BPMN design and implement their own ready-to-use business logic without requiring any prior coding knowledge.



- Flowable's condition design interface. Like other BPMN engines, it provides a straightforward screen for low-code business logic design, while data management and variable creation are handled by the IT department in the background.

Snippets from source code

```java
public class InformCustomerAboutRejectionService implements JavaDelegate {  3 usages  KaanBoraKarapinar

    public void execute(DelegateExecution execution) {  KaanBoraKarapinar
        Map<String,Object> map = execution.getVariables();
        CreditApplication application = (CreditApplication) map.get("creditApplicationJavaObject");

        application.setStatus(Status.REJECTED);
        execution.setVariables(map);


    }

}
```

- Upon reaching the end of the process, , the CreditApplication.Status value is set immutably (or temporarily before a manual check), allowing the marketplace to check the status via an API call.

```java
@GetMapping("/getStatusOfLatestApplication")  KaanBoraKarapinar *
public ResponseEntity<ApiResponse> getStatusOfLatestApplication(@RequestParam String passportNumber,
                                                                @RequestParam Long creditAmount){

    //with form
    if(!applicantService.existsByPassportNumber(passportNumber)){
        return ResponseEntity.status(HttpStatus.NOT_FOUND).body(new ApiResponse( message: "There is no applicant
    }
    Applicant applicant = applicantService.getApplicantBypassportNumber(passportNumber);
    CreditApplication cA = applicant.getApplications().getLast();

    if(!Objects.equals(cA.getCreditAmount(), creditAmount)){
        return ResponseEntity.status(HttpStatus.NOT_FOUND).body(new ApiResponse( message: "Applicant's last cred
    }

    return ResponseEntity.status(HttpStatus.FOUND).body(new ApiResponse( message: "",cA.getStatus()));
```

The customer can thereby proceed with our bank’s microcredit or choose another payment method.

Github Repo for remaining code, DTOs and controllers:
KaanBoraKarapinar/CreditAssesementApplication

Thank you for your attention.

Vielen Dank für Ihre Aufmerksamkeit.

I look forward to your feedback.
Ich freue mich auf ihre Rückmeldung.

Kaan Bora Karapınar