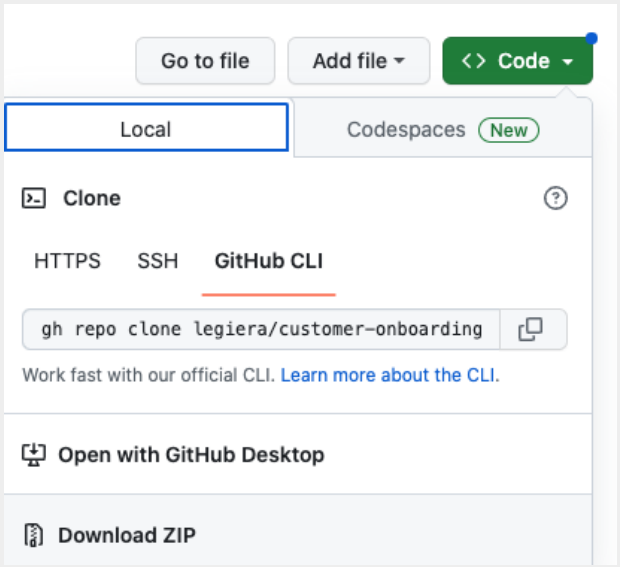
Camunda workshop instruction

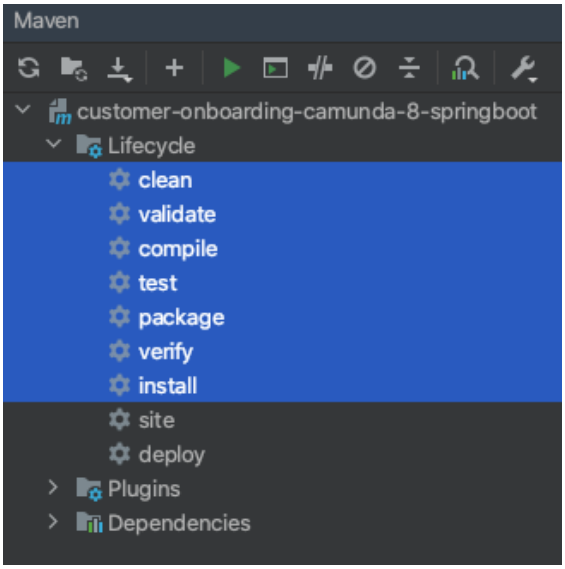
I Clone and build project

1. Clone repository:

   git clone https://github.com/legiera/customer-onboarding-camunda-8-springboot.git
   or download it as a zip file. Once the zip file is downloaded, extract it.



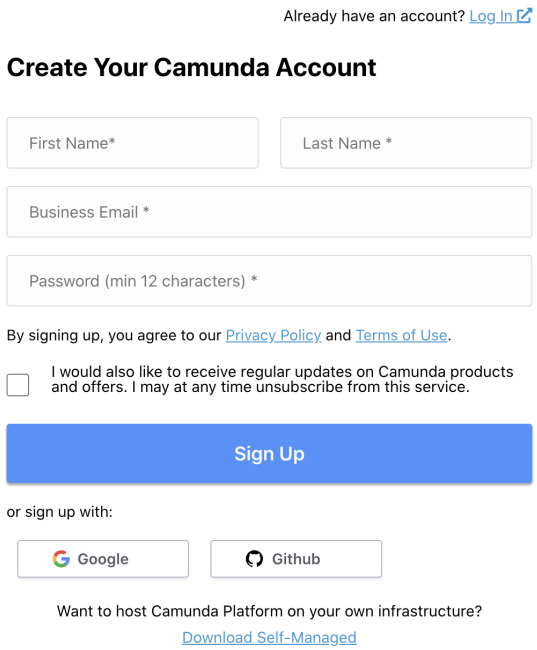
2. Open project in Intellij. Projects> Open> select folder with downloaded project.
3. Build the project: **double click CTRL** and type **maven clean install** or click **shift** and in the maven menu on the right **select clean and install:**



II Camunda Cloud

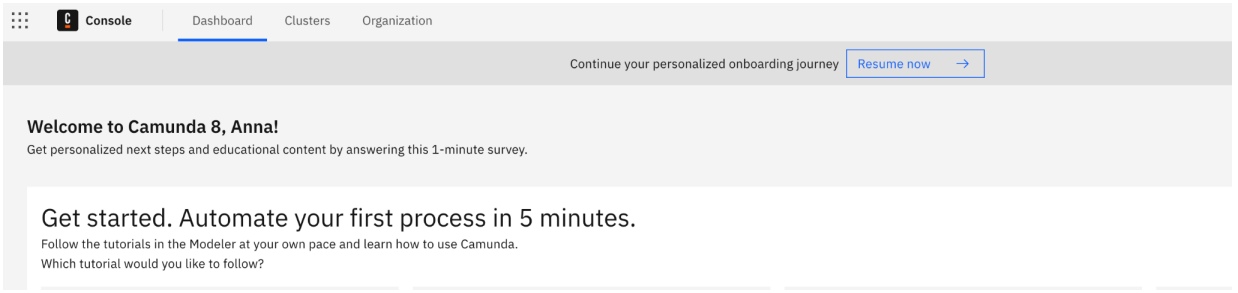
1. Create a free account on Camunda Cloud.
   a. Go to https://camunda.com/platform/ and click 

   b. Provide your details:

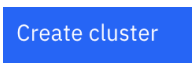


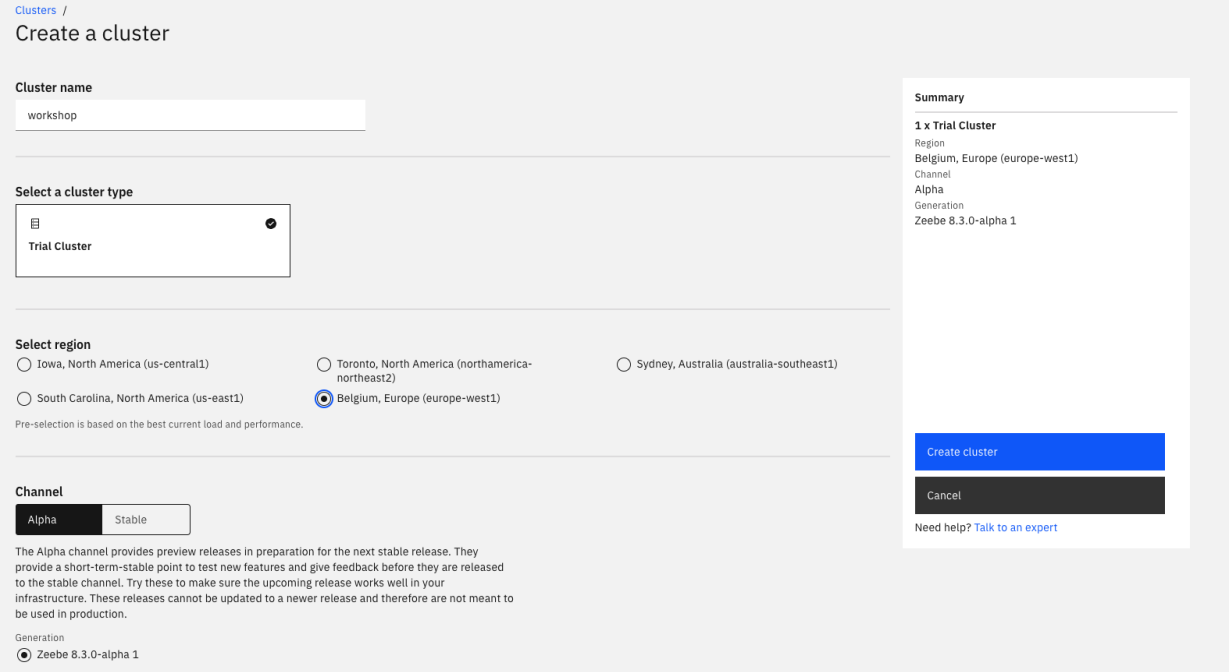
   c. You should receive confirmation on your email. Check your email and verify your account.
   d. Login on Camunda Cloud: https://camunda.com/platform/

2. Create Cluster on Camunda Cloud
   a. In the menu on the top **select Clusters** and then **Create new cluster:**



   b. Then click 

   c. Provide all details for your cluster as below and then click 

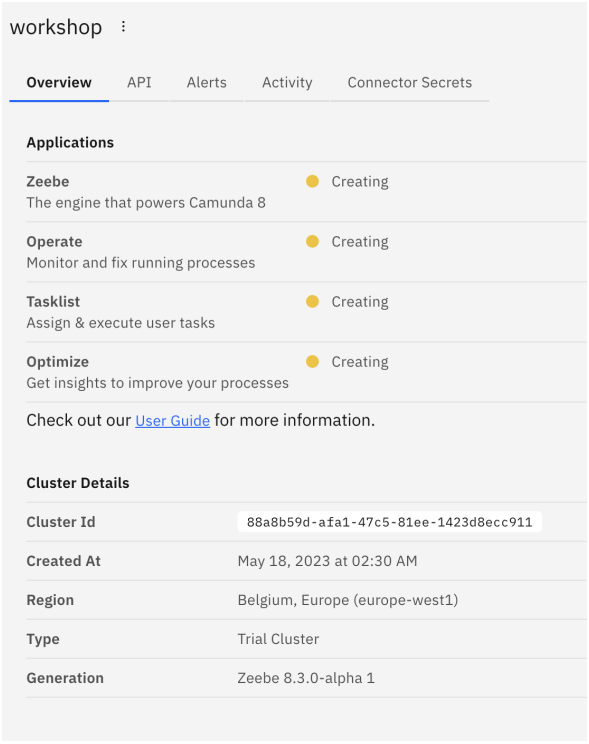


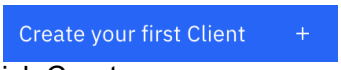
d. You should be able to see you cluster in with Creating status:

Clusters

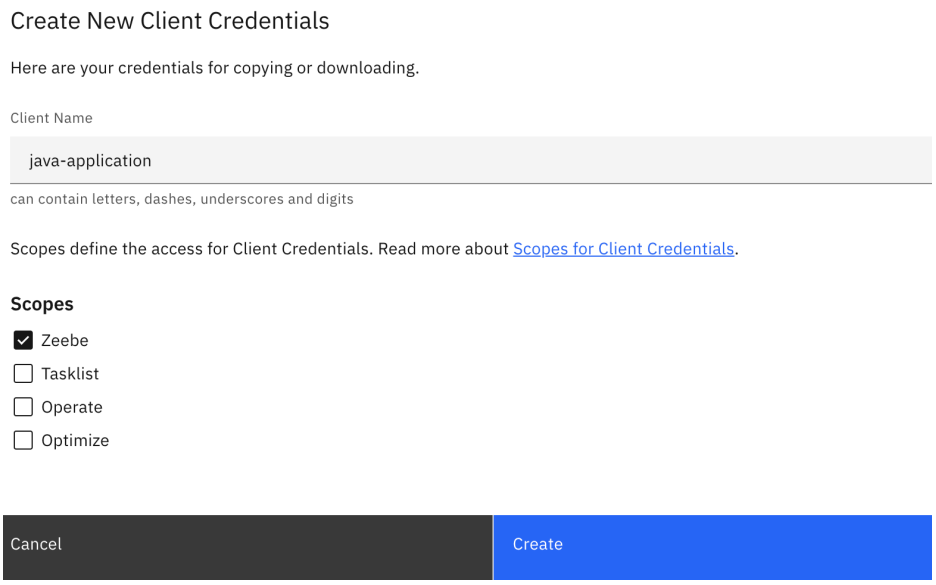
Name	Region	Generation	Type	Status
workshop	Belgium, Europe (europe-west1)	Zeebe 8.3.0-alpha 1	Trial Cluster	Creating

3. Create Client Credential
   a. Go to the Clusters tab and click the name of your cluster. You should see below details:

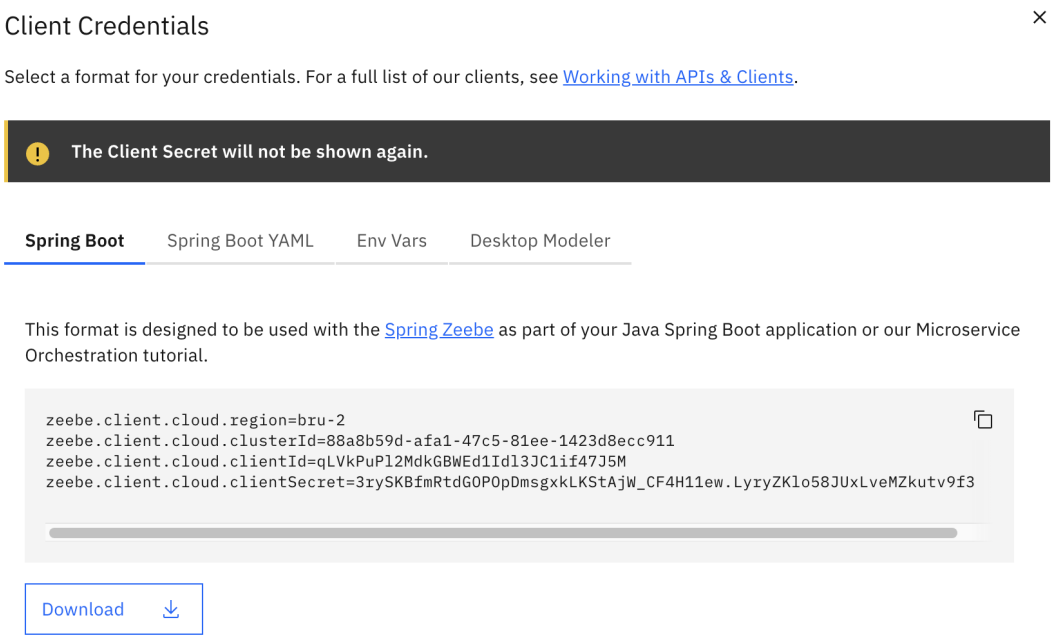


b. Then select **API tab** and click: 

c. Provide details as below and click Create:

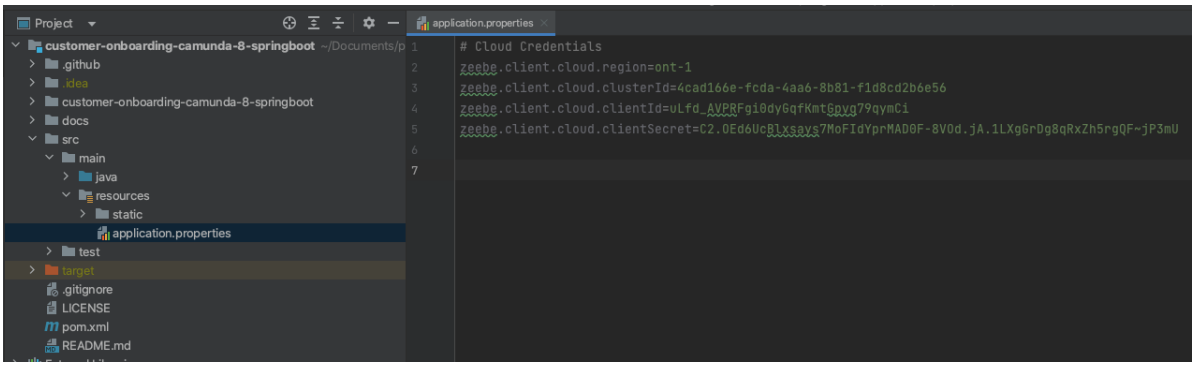


d. You should see details of your **Client Secret**:



e. Download your **Client Secret** and copy it to the clipboard.

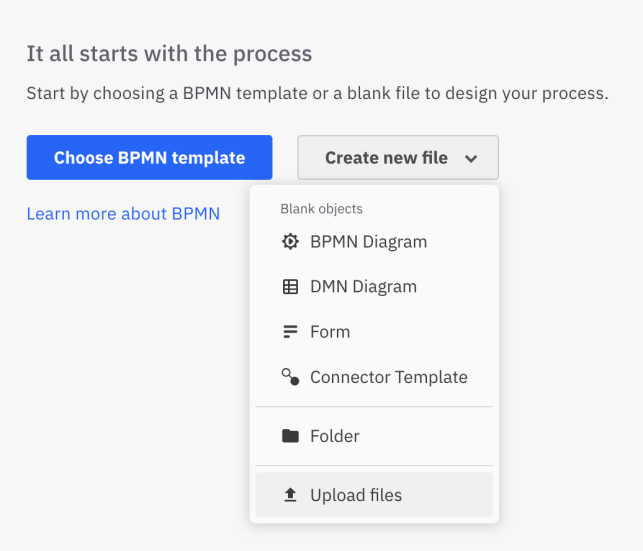
4. Go to your project in Intellij and update application.properties with your details:



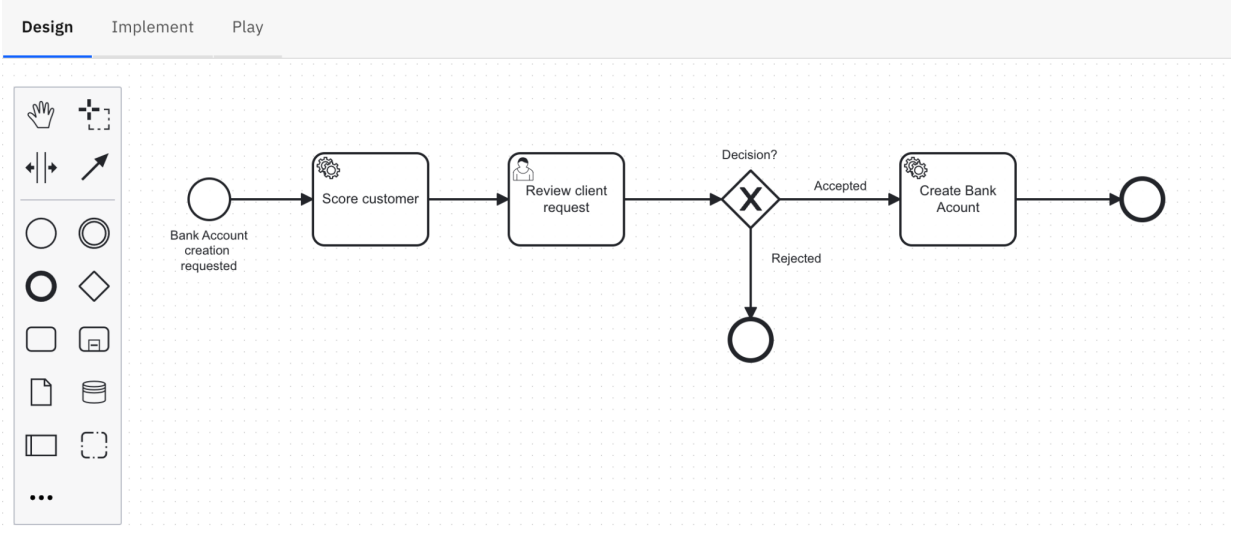
III First Exercise - Simple Bank Account System

Define a process that needs to **automate bank account creation**. Customer requires a bank account. The external system needs to score the customer. Next, the bank **employee needs to review the request** in order to decide whether an account for this Client can be created. If the decision is **successful, the request is sent to microservice** to provision the bank account. **Otherwise, the request is rejected** and the bank account is not created.

1. In the project find **bank-account-simple.bpmn**
2. In Camunda cloud go to Modeler and create new project and select Create new file and upload bank-account-simple.bpmn:

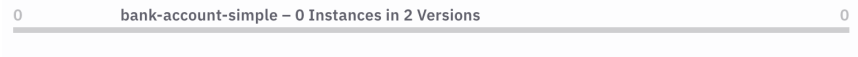


3. Now you see our model in Modeler:



4. Then click Implement tab and then 

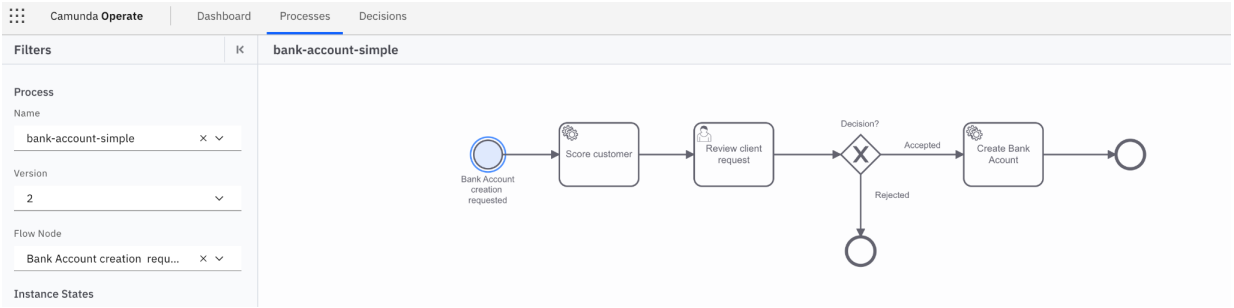
5. Then go to the Operate tab. You should see Instance of your process:

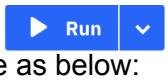


6. Run application. Right click on **BankSpringbootApplicatio**n and then run. You should see below logs in the console:

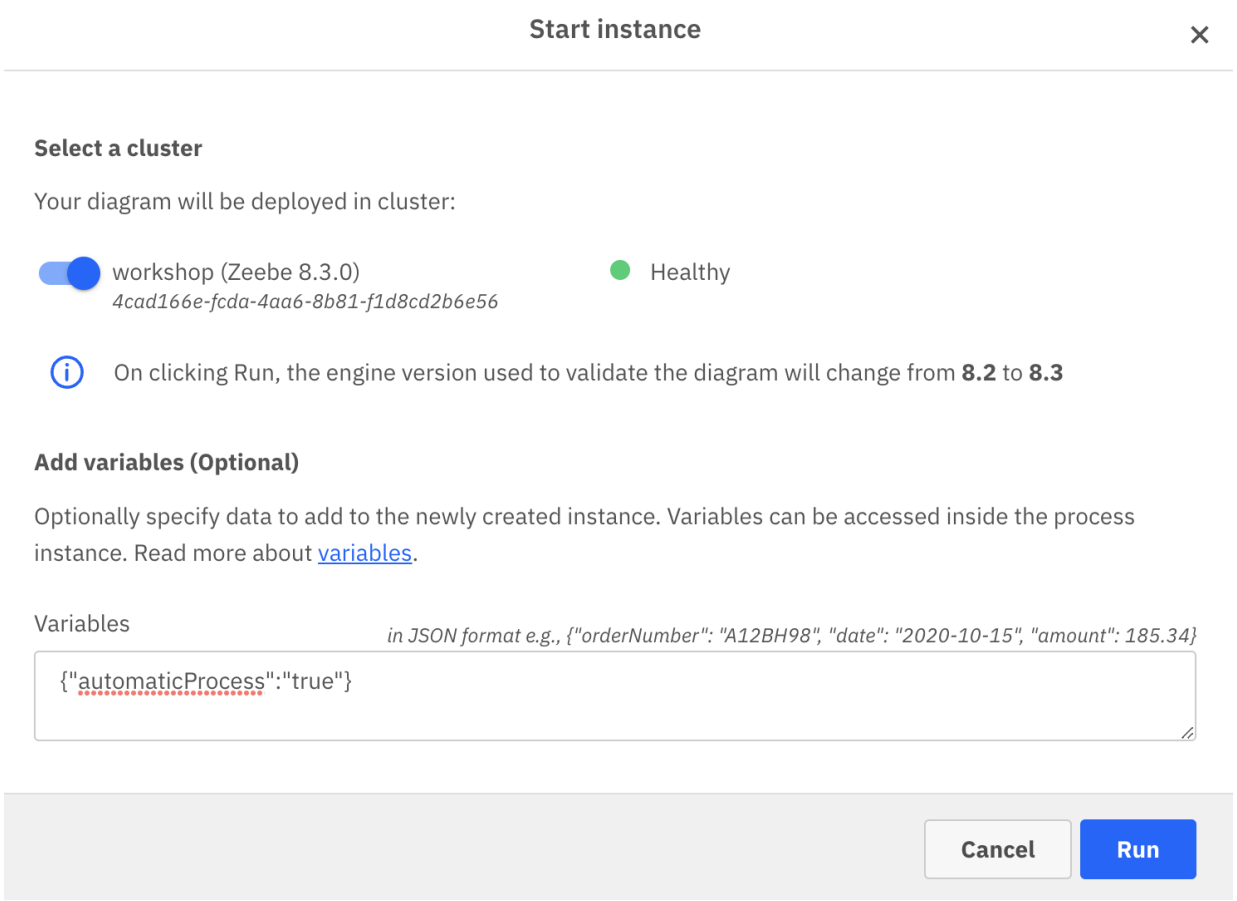
```
2023-05-18T02:49:11.747+02:00  INFO 13438 --- [           main] i.c.z.s.c.c.OutboundConnectorManager     : Reading environment variables or parsing SPI to find connectors that are not Spring beans
2023-05-18T02:49:11.748+02:00  INFO 13438 --- [           main] .u.o.OutboundConnectorRegistrationHelper : Parsing SPI to find connectors that are not Spring beans
2023-05-18T02:49:11.749+02:00  WARN 13438 --- [           main] i.c.z.s.c.c.OutboundConnectorManager     : No outbound connectors configured or found in classpath
2023-05-18T02:49:11.773+02:00  INFO 13438 --- [           main] i.c.z.s.c.jobhandling.JobWorkerManager   : . Starting Zeebe worker: ZeebeWorkerValue{type='createNewBankAccount', name='customerOnboardingGlueCode#createNewBank
2023-05-18T02:49:11.774+02:00  INFO 13438 --- [           main] i.c.z.s.c.jobhandling.JobWorkerManager   : . Starting Zeebe worker: ZeebeWorkerValue{type='scoreCustomer', name='scoringAdapter#scoreCustomer', timeout=-1, maxJ
2023-05-18T02:49:11.817+02:00  INFO 13438 --- [           main] o.s.b.web.embedded.netty.NettyWebServer  : Netty started on port 8080
2023-05-18T02:49:11.826+02:00  INFO 13438 --- [           main] i.b.o.c.BankSpringbootApplication        : Started BankSpringbootApplication in 2.364 seconds (process running for 2.679)
```

7. Click on it. Instance of our process should be the same as in the Modeler:

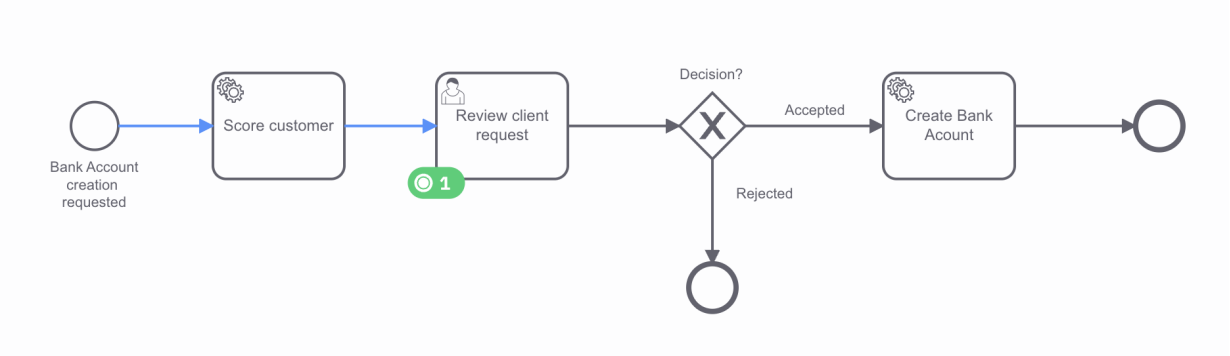


8. In the second tab go to our model in Modeler.
9. Select **Implement Tab** and click 

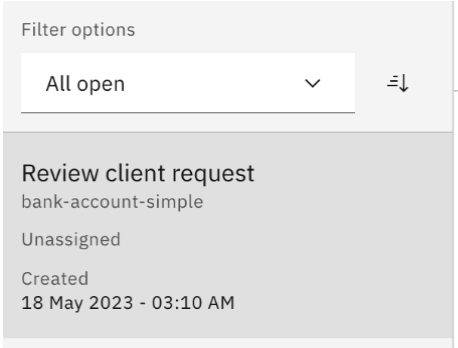
10. To start instance provide variable as below:

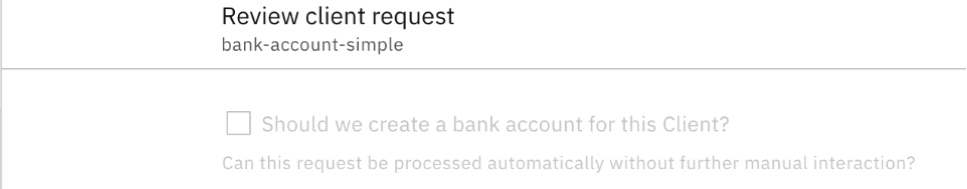


11. Now go again to your Instance. Now you should see new event that we created:

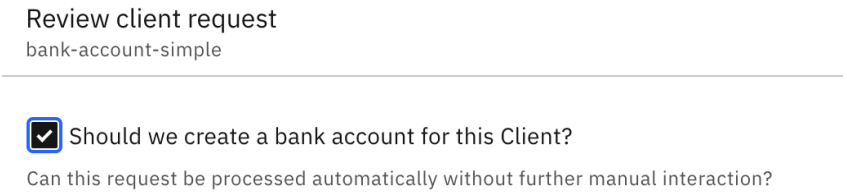


12. Go to Tasklist in the main menu. You should see the Review **client request** task.

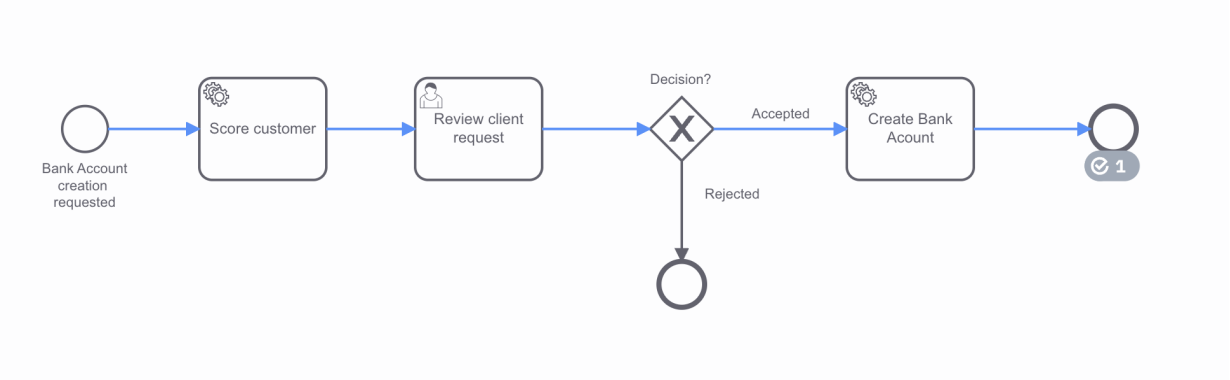




13. Click on that new task and choose:
14. Select box and Complete Task:



15. Go back to your instance and now you should see that token is processed and account is created:

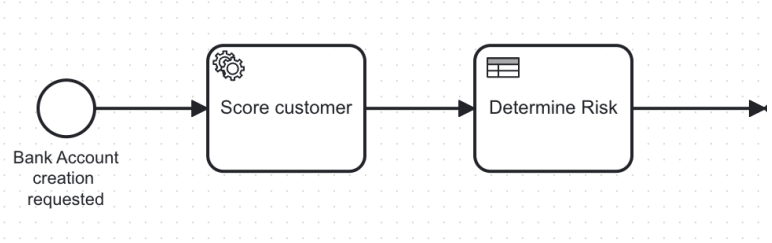


16. Check also logs in the Intellij. What you can see? What happened?
17. Now lets create event from our machine: **curl -X PUT http://localhost:8080/customer**
18. We should see the same effect as before.

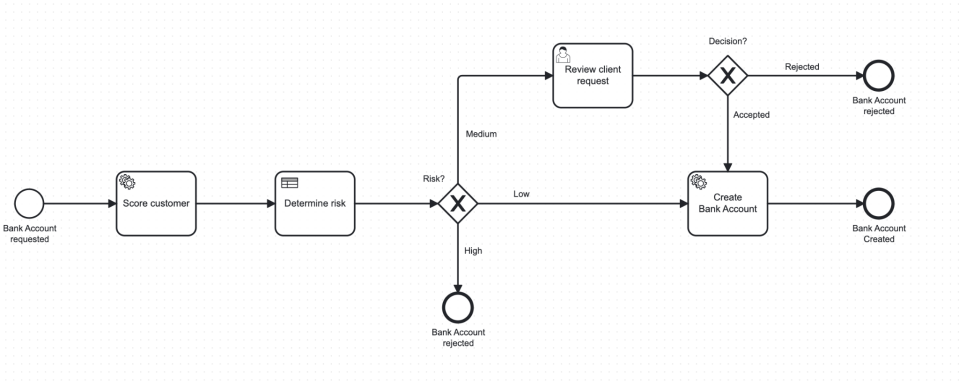
IV Second exercise - extended Bank System

Define a process that needs to automate bank account creation. Customer requires a bank account. The external system needs to score the customer. **If clients earnings are above 10k PLN and age is above 18 years old, a bank account creation request is sent to Create Bank Account service. If age is under 18 and earnings are negative the request is rejected automatically.** If not, then the bank employee needs to review the request in order to decide whether an account for this Client can be created. If the decision is successful, the other employee has to perform some steps in order to provision the bank account bank account creation request is sent to Create Bank Account service. Otherwise, the request is rejected and the bank account is not created.

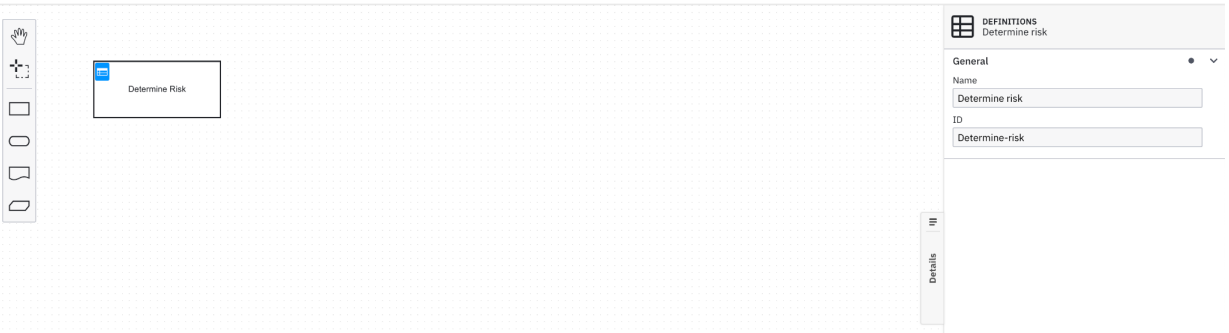
1. Let's start with the BPMN model. Let's modify our existing model.
2. Go to Modeler, find the bank-account-simple model and duplicate it and name it bank-account-extended.
3. Open the model in Moder, then click ••• in the edit menu on the left and search for **Business Rule Task.** Add it to your model behind **Score Customer Service Task** with name **Determine risk and ID TaskValidateCustomerApplication**



4. Next modify the exclusive gateway to have three outputs with Medium, High and Low risk. If risk is Low we want to send the account creation request directly to the microservice responsible for account creation. If risk is Medium, we want the employee to review the request and then send it to the microservice responsible for account creation. If risk is high the account creation request should be rejected.



5. Create New DMN Diagram in Modeler with name Determine risk and id determine-risk

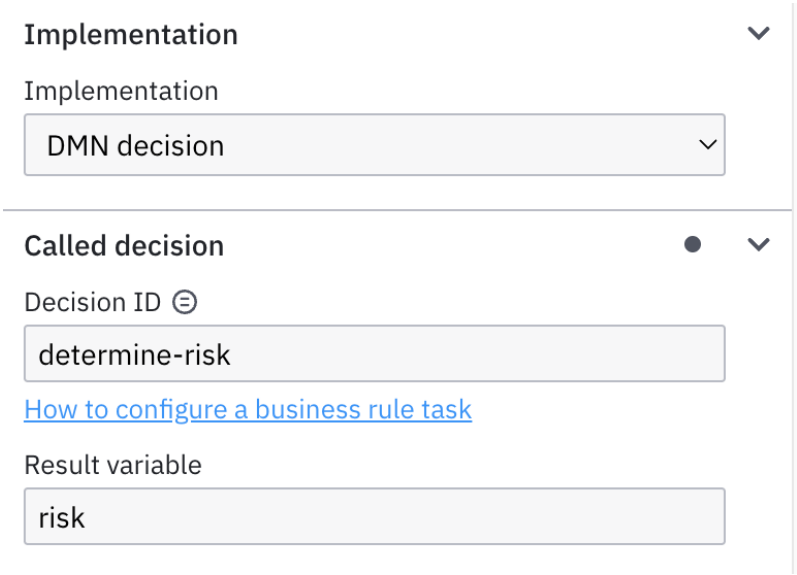


6. Then click on to define your logic:

Determine Risk | **Hit Policy:** Unique

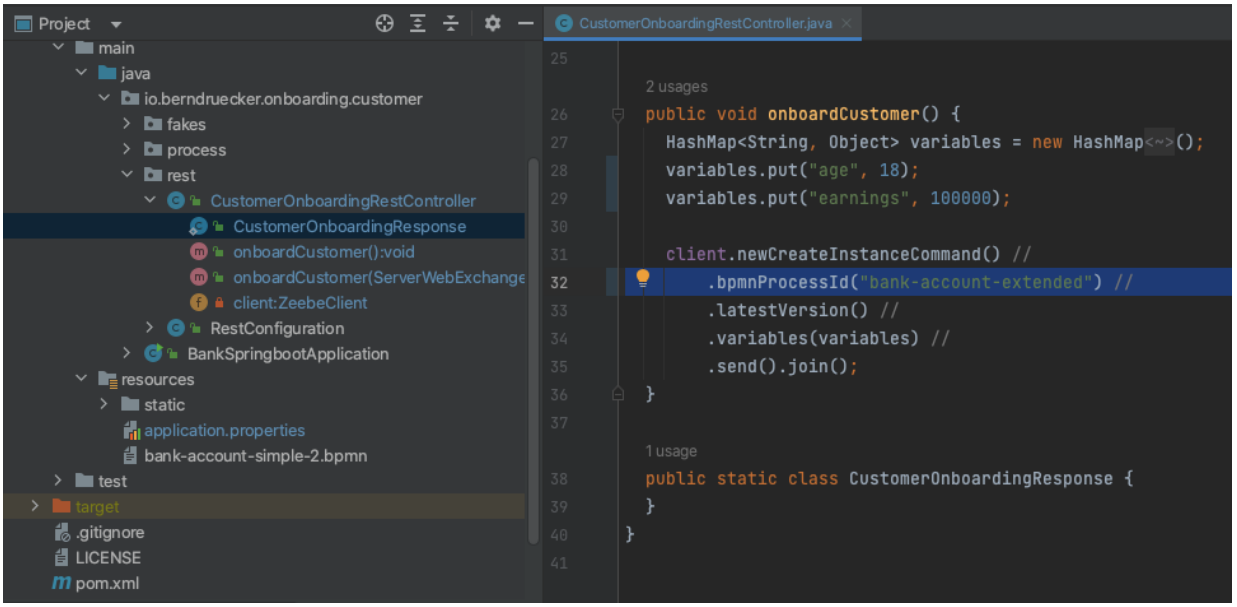
	When age string	And earnings string	Then risk string	Annotations
1	>18	>10000	low	
2	<18	<0	high	
3	>18		medium	
4	-	-		
+				

7. In bank-account-extended Model link Business Rule task to our created DMN. Add below properties:



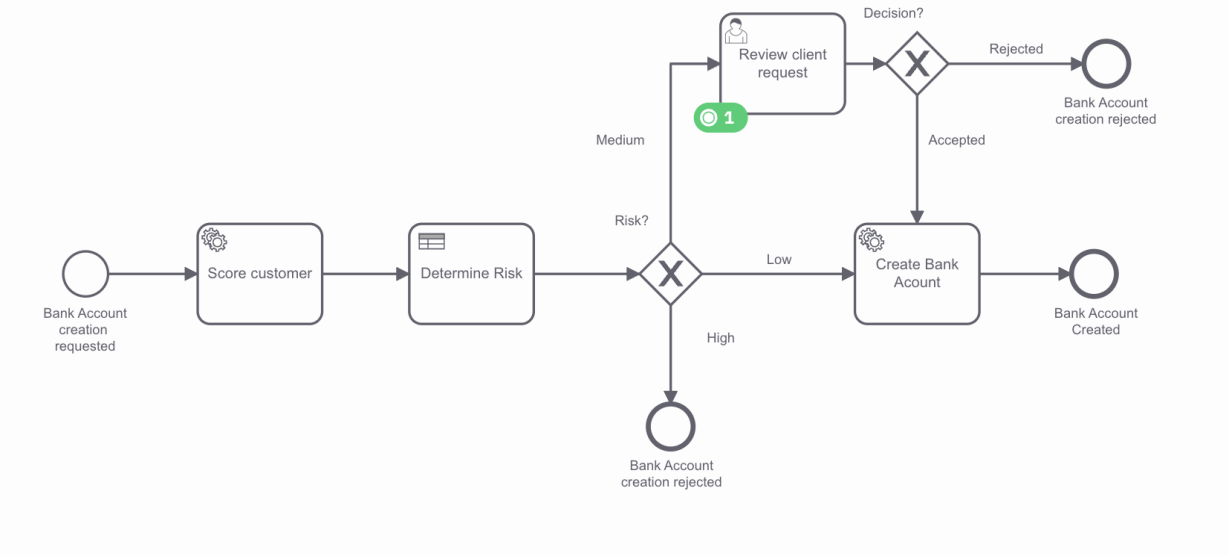
8. Deploy Determine Risk DMN

9. Update code in the Intellij



10. Run curl command : **curl -X PUT http://localhost:8080/customer**

11. You should see toke:



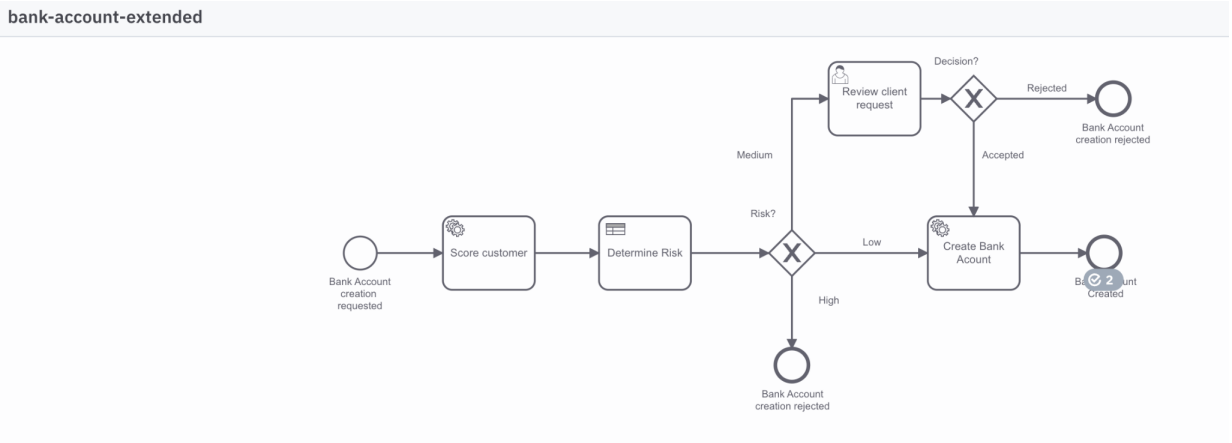
And logs in termina:

```
INFO 15690 --- [pool-6-thread-1] i.b.o.customer.process.ScoringAdapter    : Gained new customer!
```

12. Go to the task list and approve the **User Task**.

13. Account see created account in the logs and

```
INFO 15690 --- [pool-6-thread-1] i.b.o.c.p.CustomerOnboardingGlueCode     : Create new bank account via REST [{"key":4503599627586411,"type":"createNewBankAccount",
INFO 15690 --- [ctor-http-nio-3] .b.o.c.f.CreateBankAccountRestController : REST API called
```



14. Play around, create different events with different variables and check behaviour.