

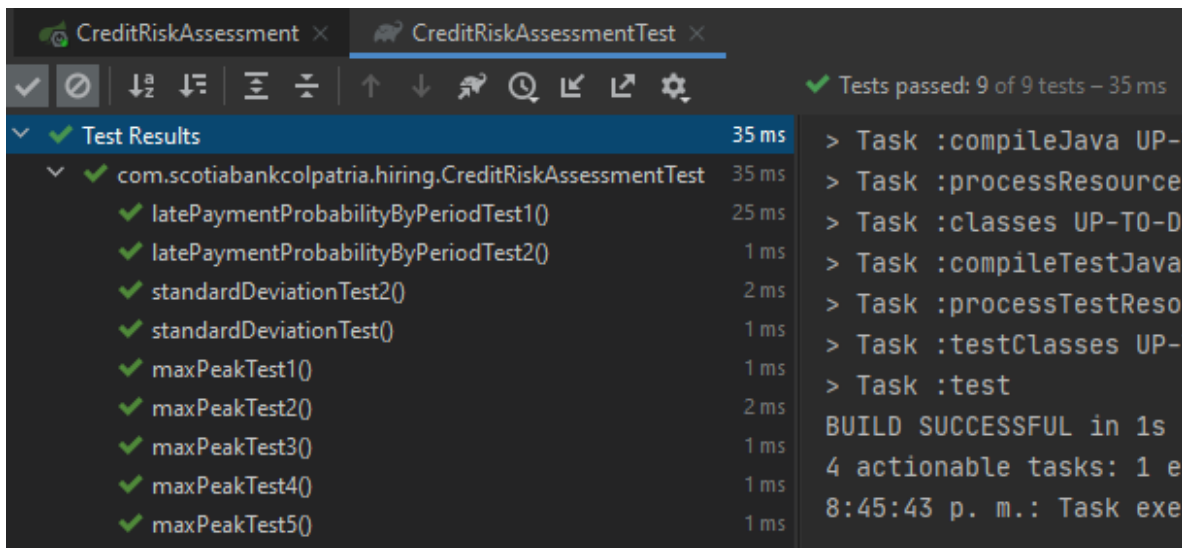
## Solución prueba técnica Evaluación de Riesgos

Realizada por: Brayan Fernando Cárdenas Piragauta.

En primera instancia, se implementaron los 3 métodos solicitados en la clase **CreditRiskAssessment**, con la lógica necesaria para cumplir las correspondientes funcionalidades:

```
standardDeviation,  
paymentDelayMaxPeakIndex,  
latePaymentProbabilityByPeriod.
```

Se ejecutaron las pruebas unitarias propuestas y pasaron todas como se puede observar en la Imagen:

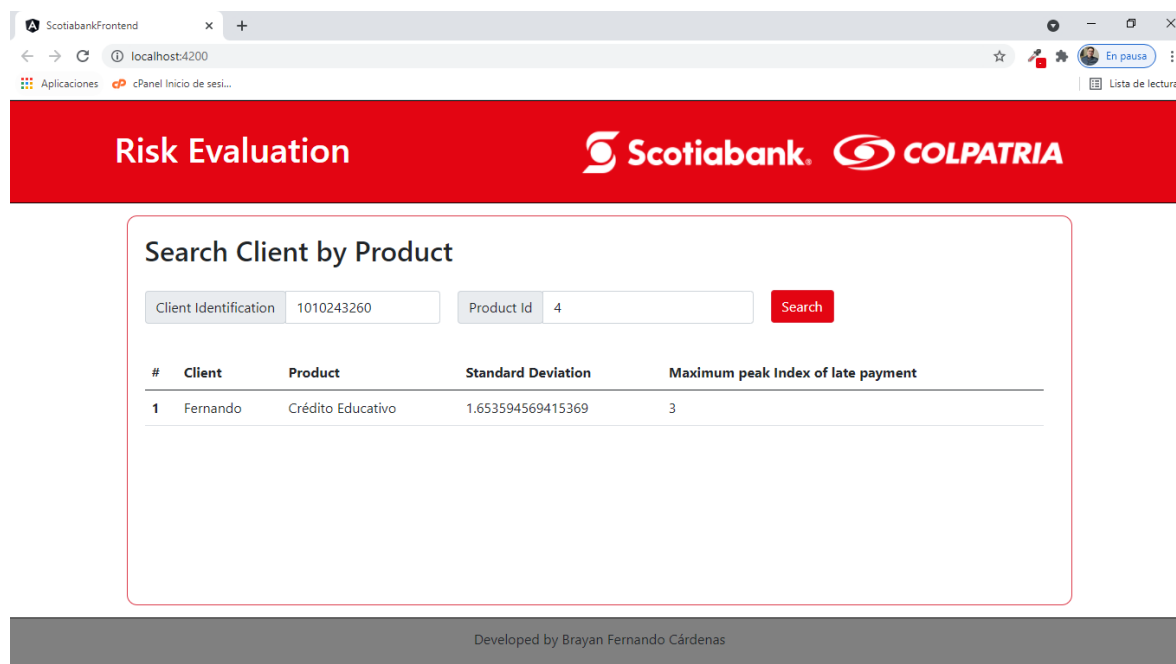


Luego de que pasaran los test, se propone realizar una aplicación web con las tecnologías deseadas para el cargo, donde se pudiese validar el funcionamiento de los métodos, por consiguiente, se diseña la siguiente aplicación:

## FRONTEND:

A continuación, se presenta el Frontend de la aplicación, esta Interfaz de usuario fue desarrollada usando Angular, Bootstrap, HTML y CSS.

Código fuente: /hiring-exercise-06-FRONTEND



La Interfaz de usuario permite consultar la Evaluación de Riesgo de un cliente, teniendo en cuenta su número de identificación (cédula de ciudadanía) y el producto a consultar, de tal forma, la aplicación retorna la desviación estándar y el Índice del pico máximo de pagos atrasados de un producto del cliente, mostrando el resultado de dos de los métodos implementados. Esto lo hace realizando peticiones al Backend de la aplicación, cuya lógica usa la información de pagos atrasados de una base de datos PostgreSQL y se muestra más adelante.

## Search Client by Product

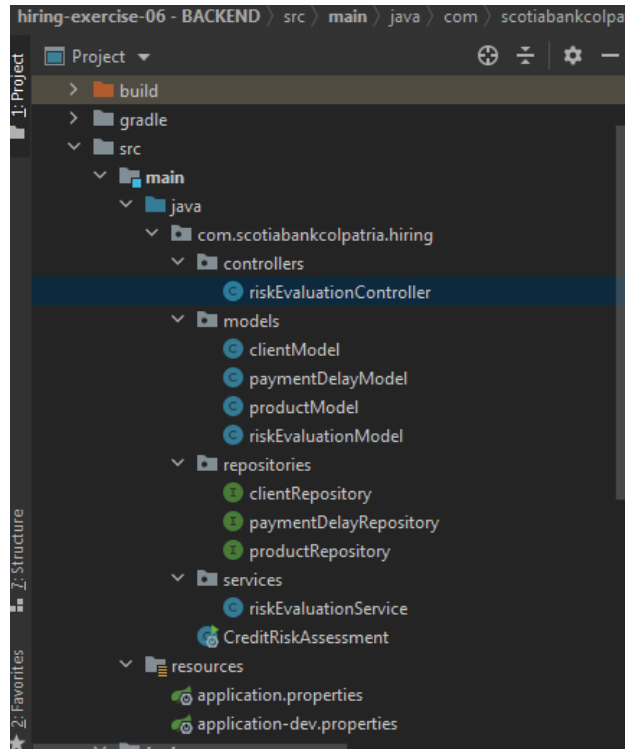
Client Identification	1010243260	Product Id	4	Search
-----------------------	------------	------------	---	--------

#	Client	Product	Standard Deviation	Maximum peak Index of late payment
1	Fernando	Crédito Educativo	1.653594569415369	3

## BACKEND:

La parte Backend de la aplicación se desarrolló usando Java 1.8, Spring Boot, Spring Data JPA para modelar capa de datos y sus respectivos repositorios, Spring Web para exponer los servicios de tipo Rest para ser consumidos por el Frontend, y Gradle como gestor de dependencias.

Código fuente: [/hiring-exercise-06-BACKEND](#)



Como se puede observar en la Imagen, el Backend de la aplicación fue desarrollado teniendo en cuenta una estructura de paquetes, que incluyen Modelos, Controladores, Servicios y Repositorios.

**riskEvaluationController** : Controlador de tipo REST que expone los servicios de la aplicación, recibe petición http.

**riskEvaluationService**: Servicio principal de la aplicación, en él se aloja la lógica de negocio de la misma, haciendo uso de los repositorios o capa de acceso a datos.

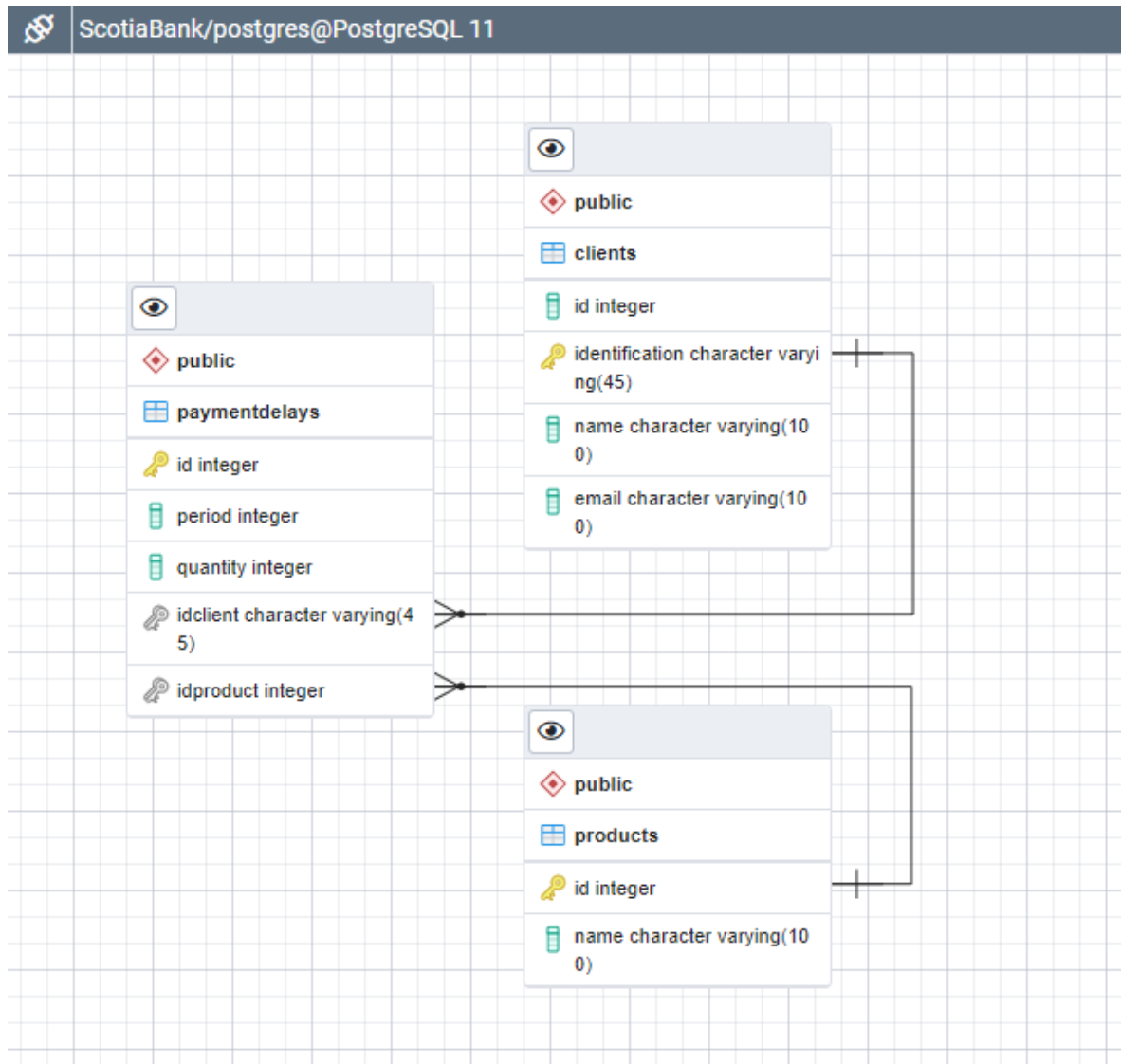
**Models**: Clases que representan entidades de tablas de BD. (client, product, paymentDelay)

**Repositories**: Interfaces que tienen como fin el mapeo de la capa de datos de cada una de las entidades descritas en los modelos. (client, product, paymentDelay).

## BASE DE DATOS:






Para el diseño de la capa de datos, se diseña una base de datos en PostgreSQL, se plantea el modelo entidad relación mostrado en la siguiente Imagen, donde es posible observar tres tablas, paymentdelays, clients, products con sus respectivos campos, llaves primarias y relaciones de llaves foráneas:

Modelo entidad relación de la base de datos diseñada:



La base de datos fue llenada con el script insert.sql y de esta forma en la tabla principal (payment delays) se pueden observar los pagos atrasados, relacionados con el periodo, cantidad de día atrasados, el cliente y el producto del banco:

## Data Output

	 id [PK] integer	 period integer	 quantity integer	 idclient character varying (45)	 idproduct integer
1	43	0	0	1010243260	1
2	44	1	1	1010243260	1
3	45	2	0	1010243260	1
4	46	3	0	1010243260	1
5	47	4	5	1010243260	1
6	48	5	1	1010243260	1
7	49	6	0	1010243260	1
8	50	7	0	1010243260	1
9	51	0	0	1010243260	2
10	52	1	3	1010243260	2
11	53	2	4	1010243260	2
12	54	3	3	1010243260	2
13	55	4	8	1010243260	2
14	56	5	2	1010243260	2
15	57	6	0	1010243260	2
16	58	7	2	1010243260	2
17	59	0	7	1010243260	3

## FUNCIONAMIENTO DE LA APP:

La aplicación de spring se ejecuta a través del puerto 8081 del computador local, y con un contextpath o url de acceso base al servicio:

<http://localhost:8081/scotiabank/api/CreditRiskApplication/getClientsRiskEvaluationByProduct>

<http://localhost:8081/scotiabank/api/CreditRiskApplication/getClientsRiskEvaluationByProduct?product=1&client=1010243260>

La base de datos debe estar creada y con las configuraciones existentes en el application.properties del ambiente de desarrollo.

con el servicio corriendo se procede a compilar el proyecto Angular del frontend, con el comando ng serve sobre la ruta del explorador de archivos, al hacerlo se despliega la aplicación en el puerto 4200 y queda lista para ser usada por el usuario.