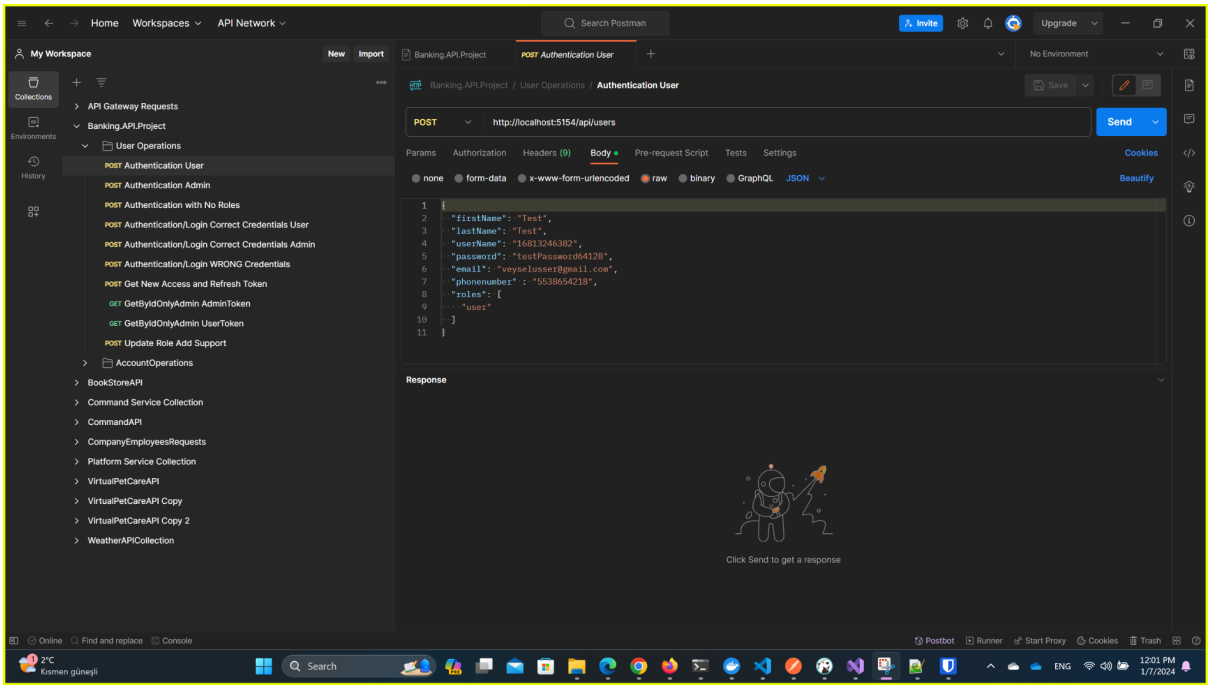
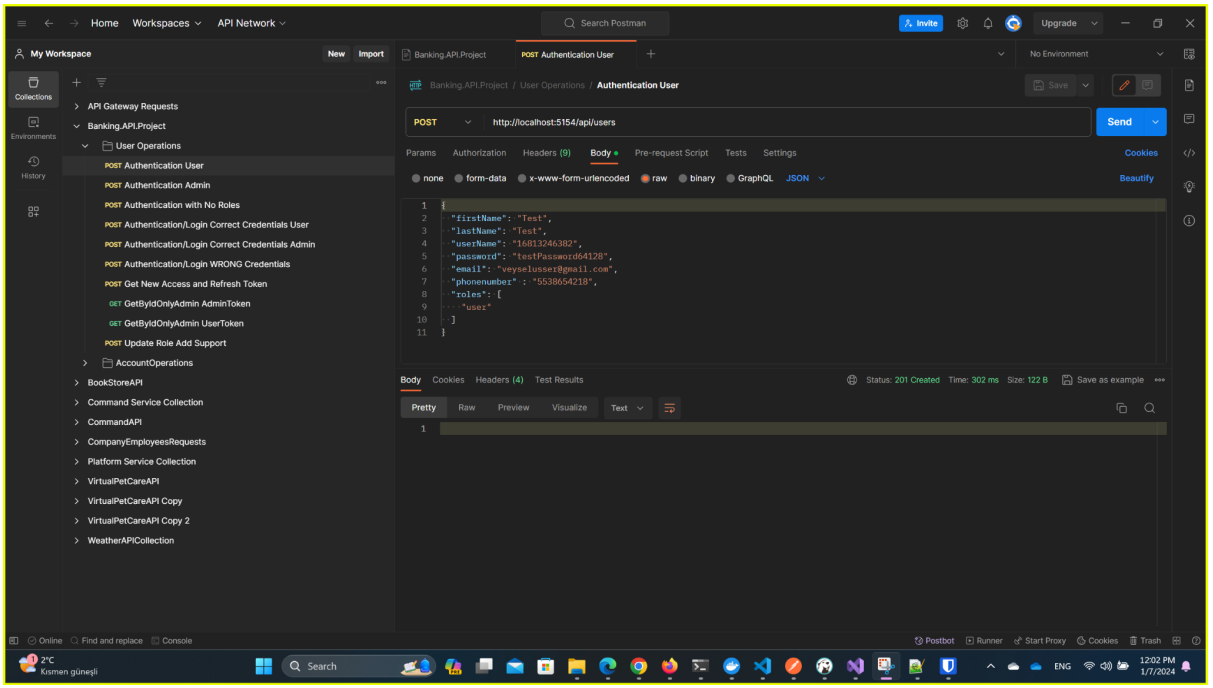
User Operations

Authentication:

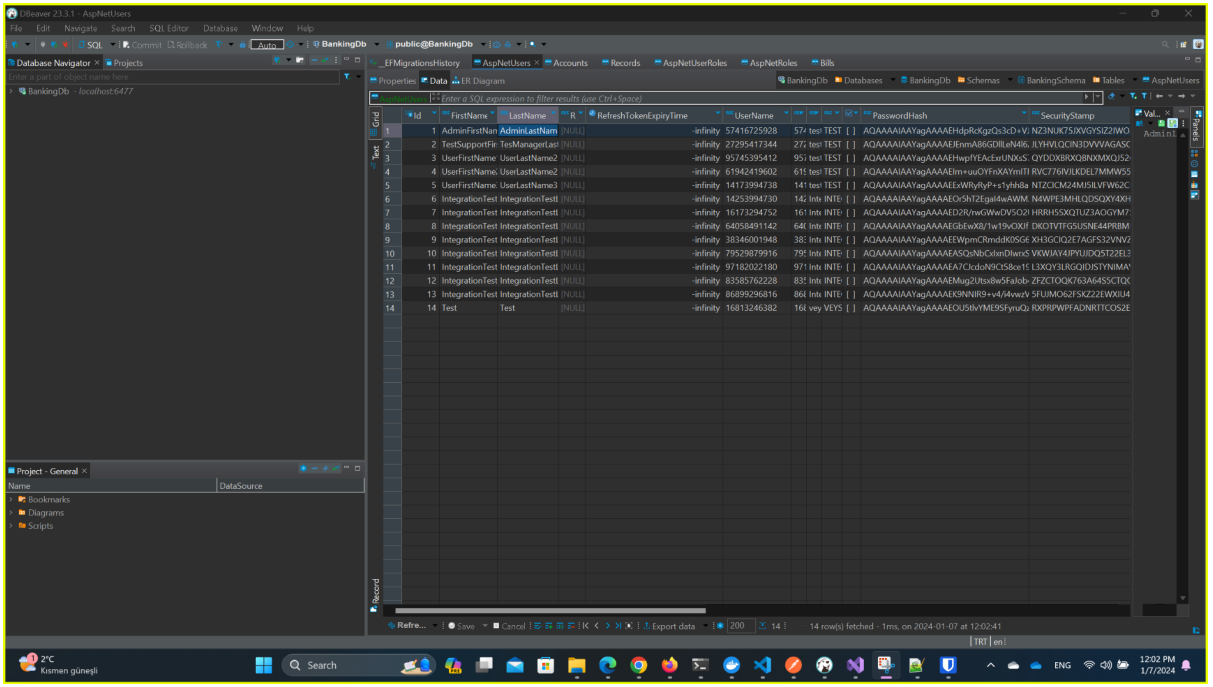
Users can sign up by providing valid parameters.



Username must be their TR id number.



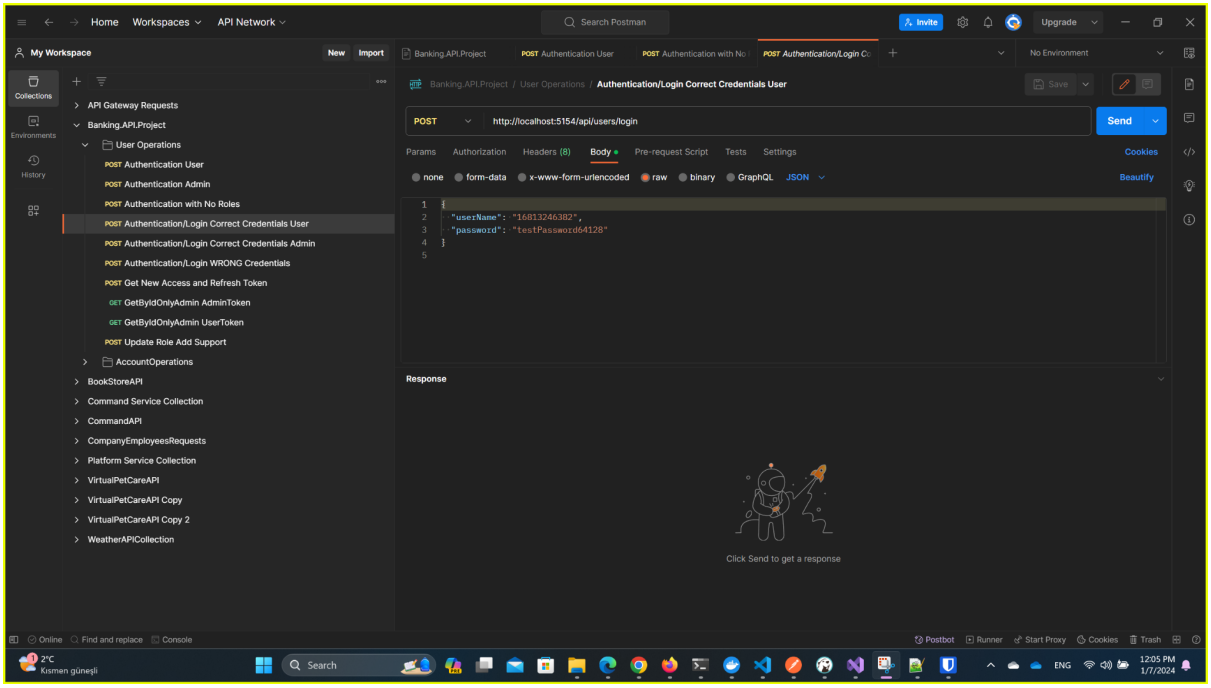
Returns 201 created if the user is created.



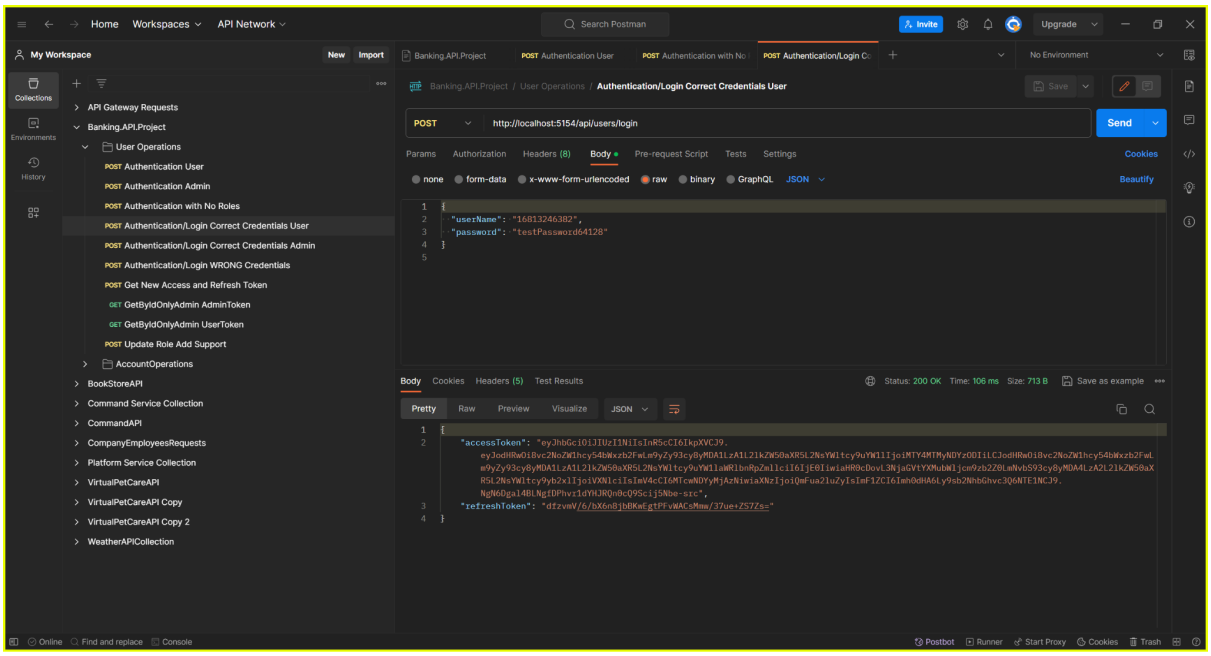
User password is stored in db after hashed.

Login:

For login, the user must provide correct credentials. Returns a bad request if either the username or password is wrong. Else, returns both access token and refresh token. Also, refresh token is stored in db. Anytime an access token expires, the user can get a new one.



A successful login operation returns both access token and refresh token.

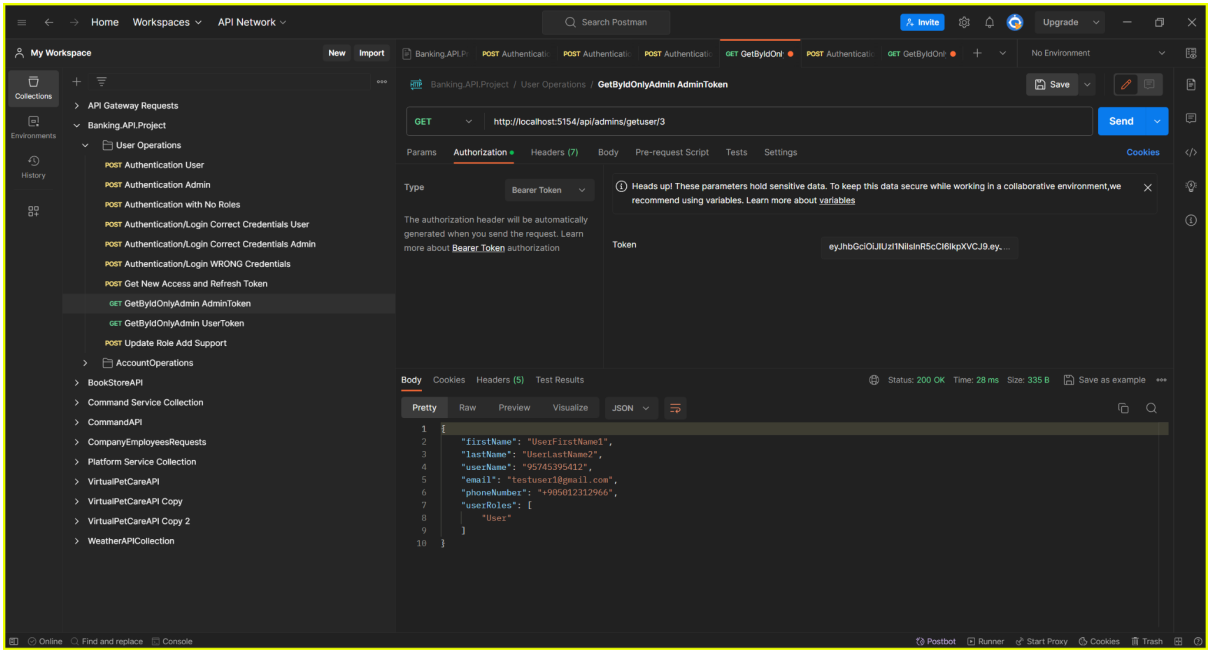


User Roles:

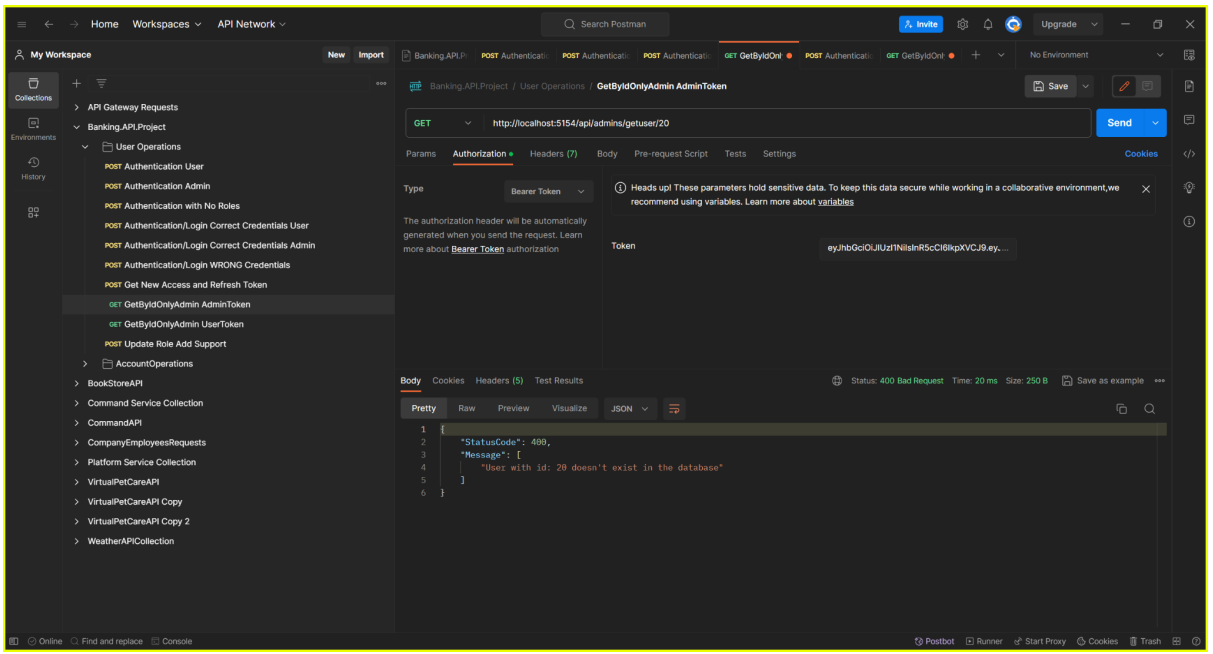
There are 3 roles for this api. Admin, Support and User.

Only admins can get users by their id.

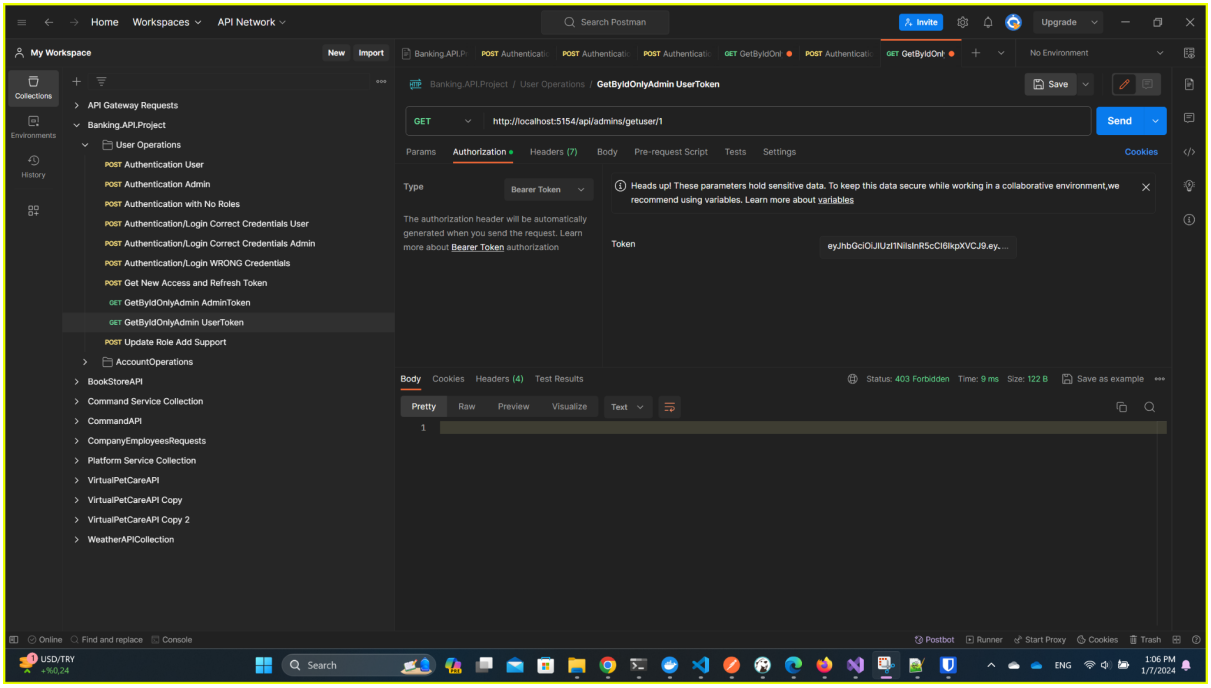
After supplying their access token, admins can get users by their id. If no user is found, they get a bad request.



A bad request as there is no user with id 20.



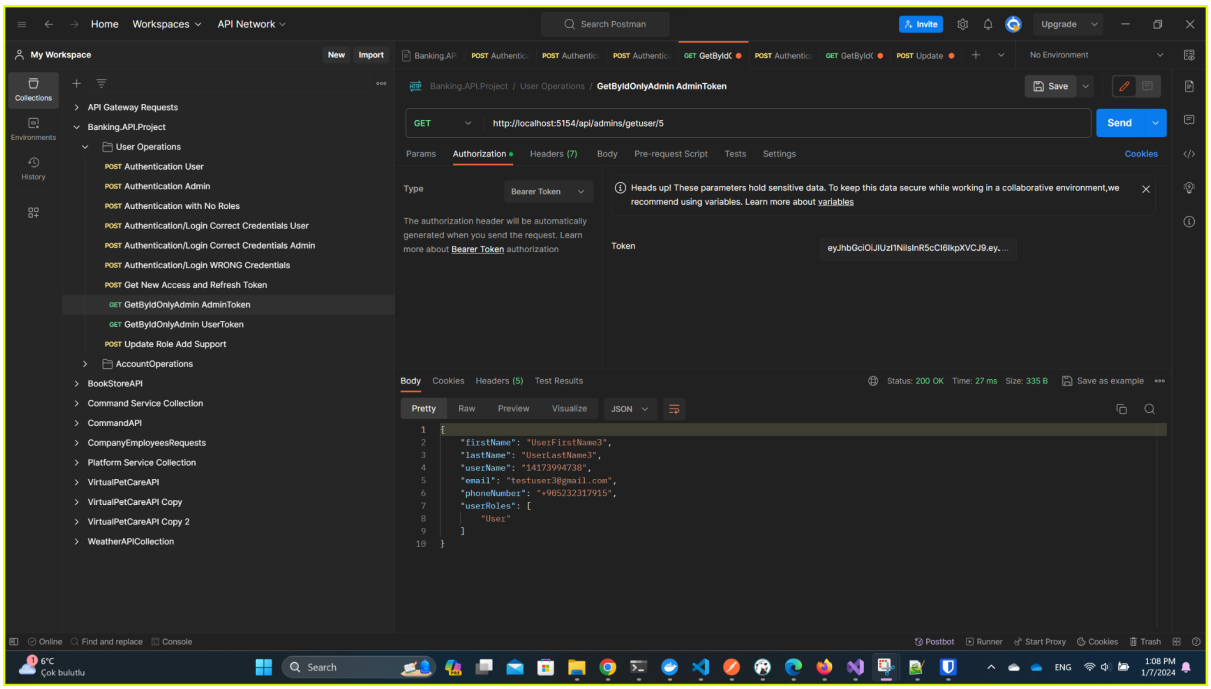
No roles other than “Admin” can use this endpoint. Users and Support roles will get 403 Forbidden.



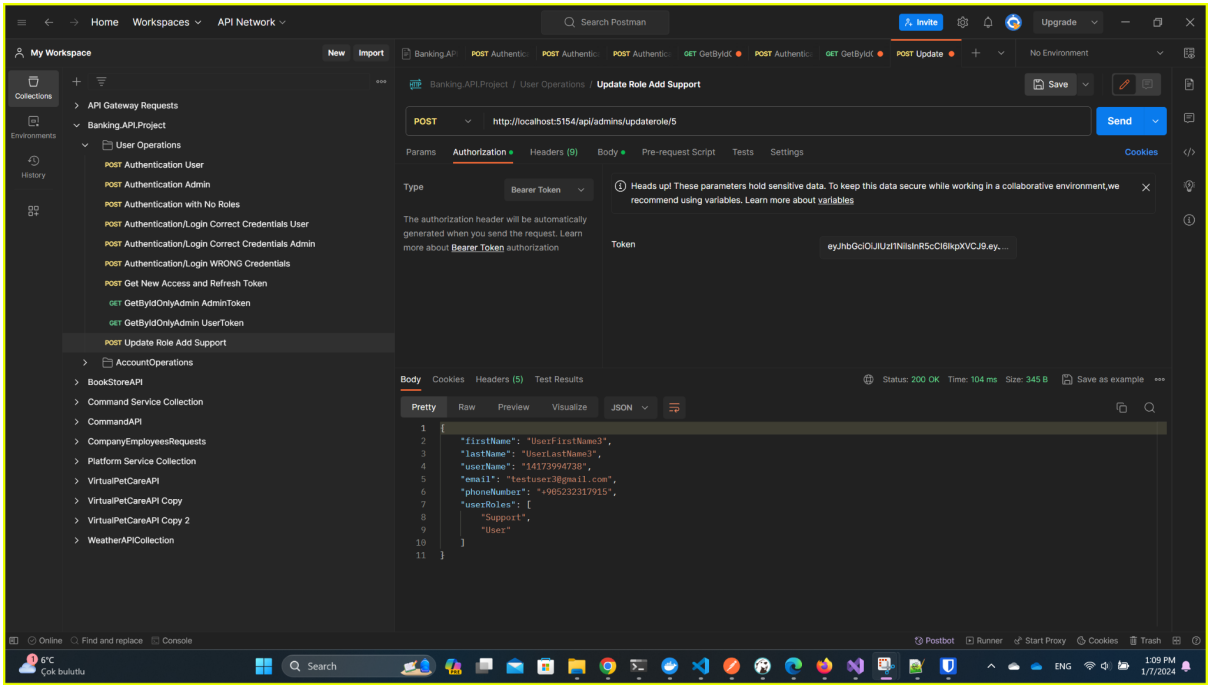
Updating Roles:

Only Admin can update roles.

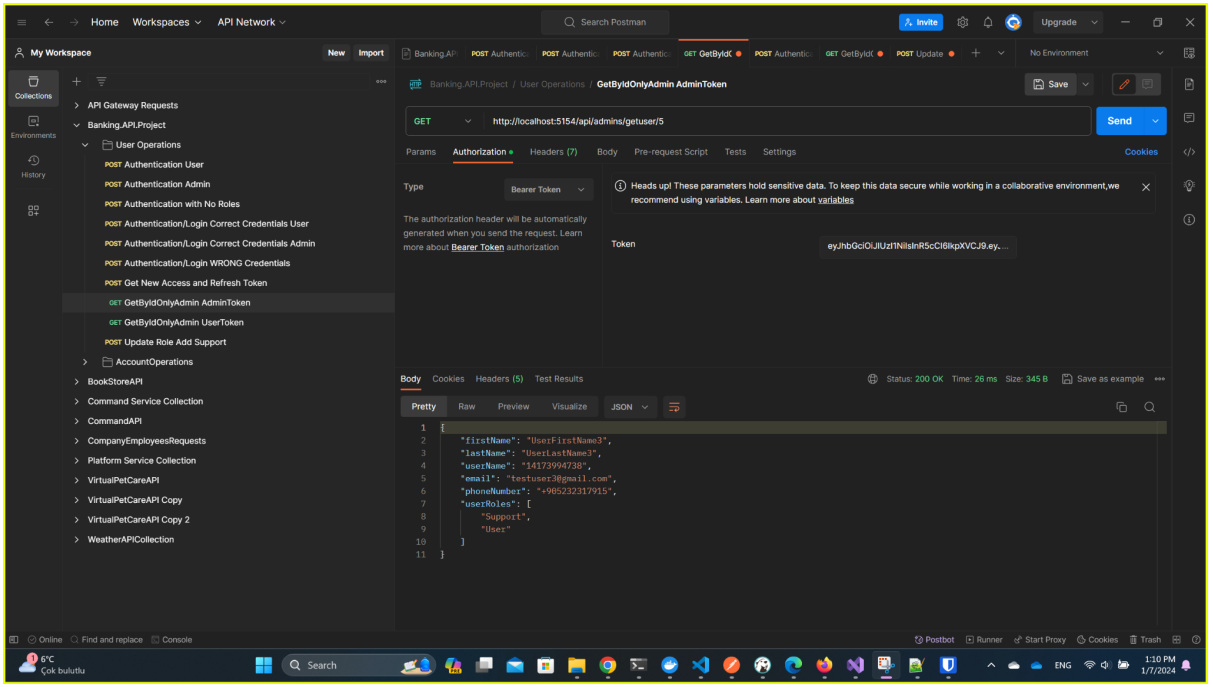
Before updating.



After updating:

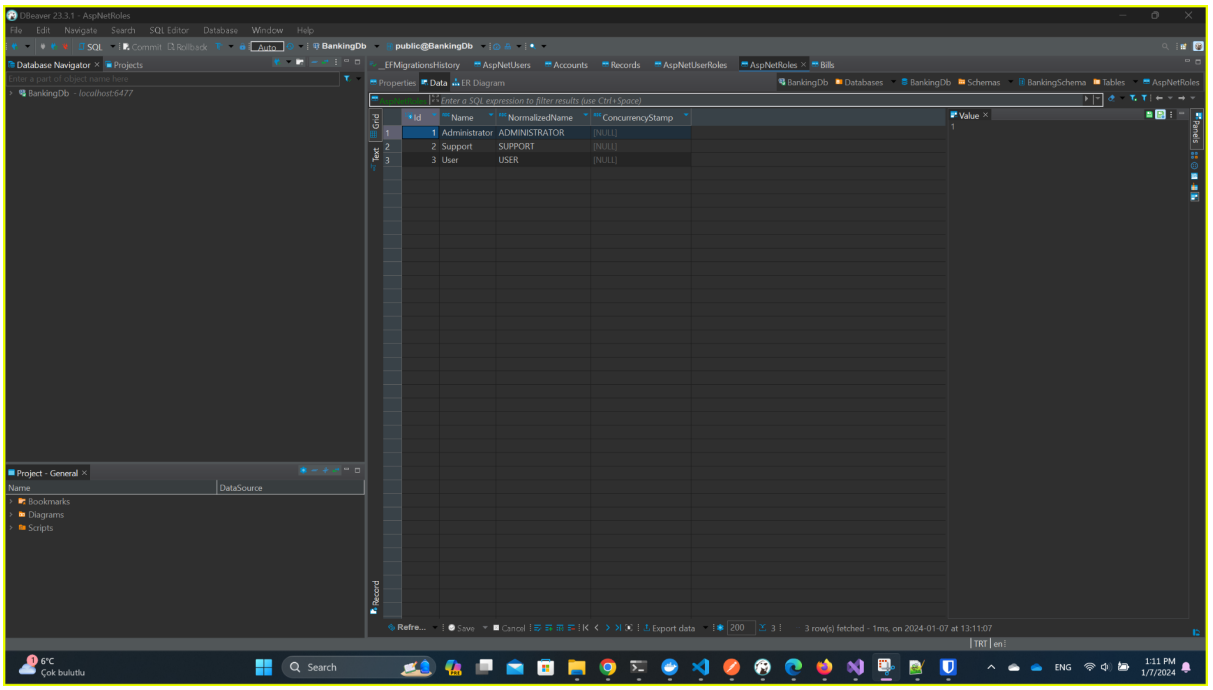


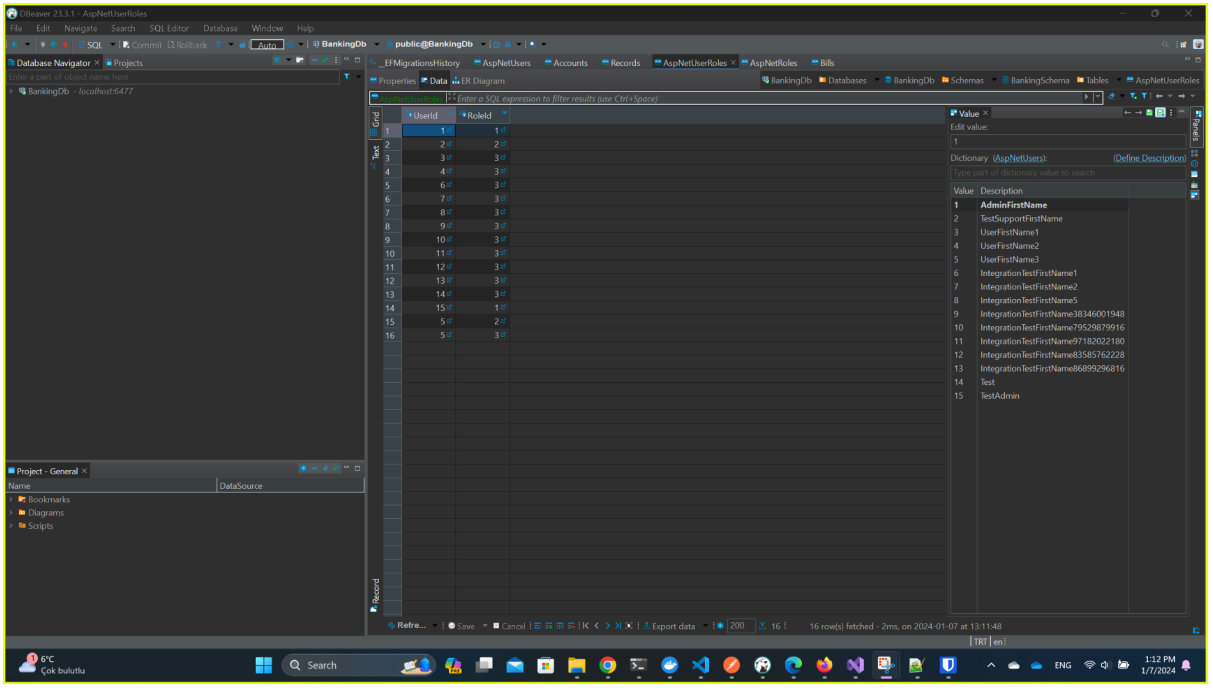
Verifying with get user by id:



Also in db:

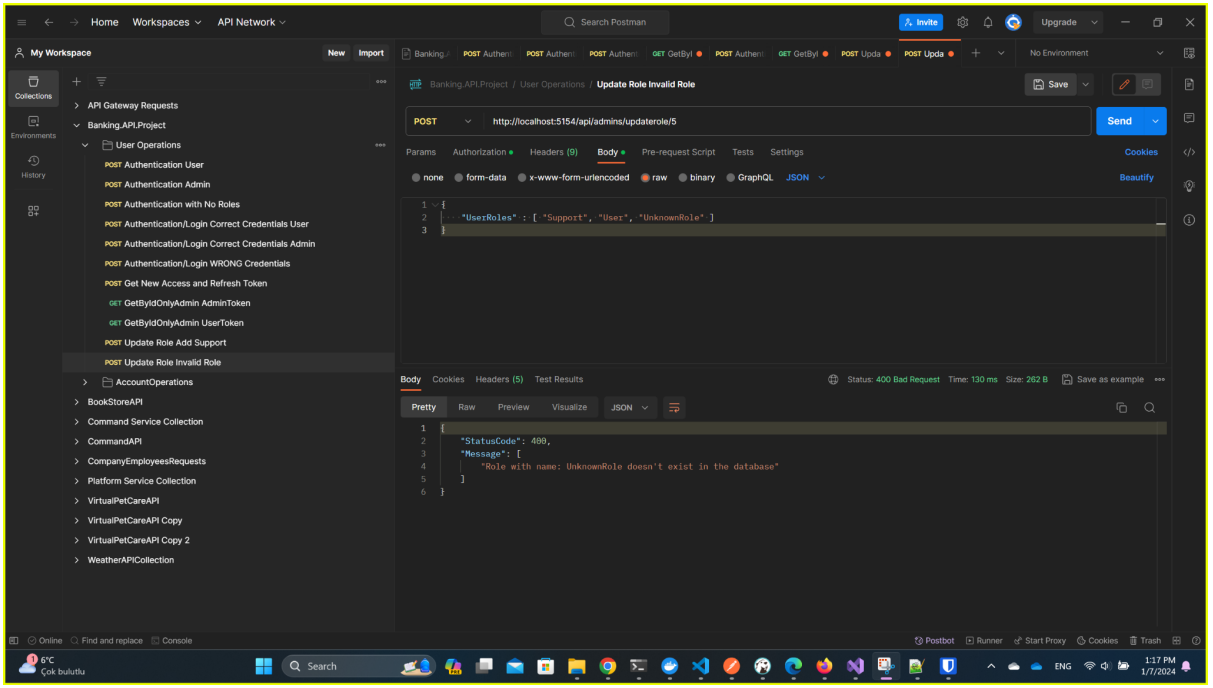
Support role id is 2 and User role id is 3.



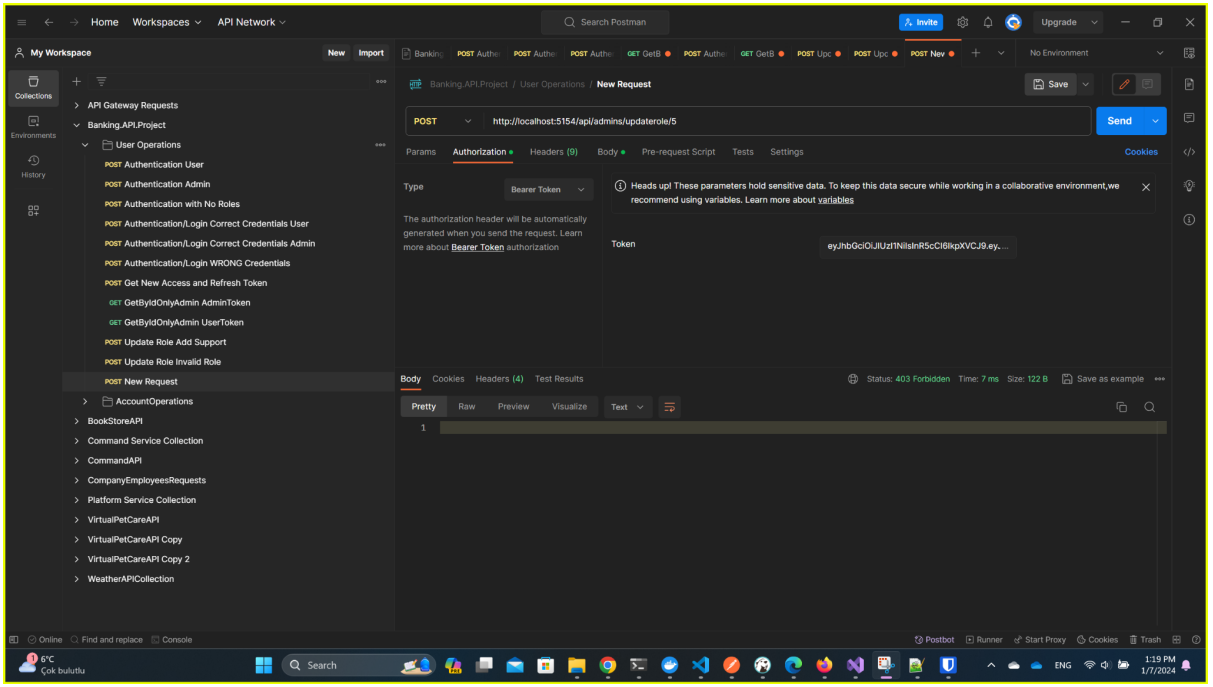


There are two rows with user id 5. Role id is 2 and 3. So it's correctly executed.

If the admin sends a role that does not exist the api will return a bad request.



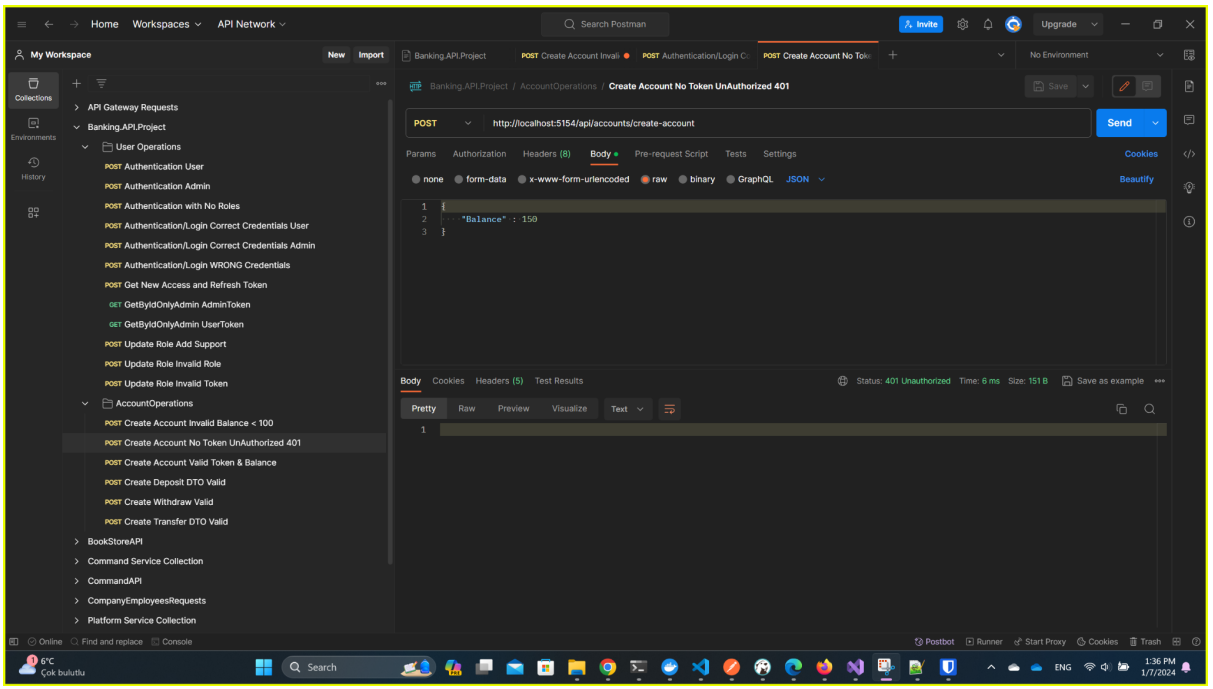
Other Roles will get 403 forbidden if they try to execute this endpoint.

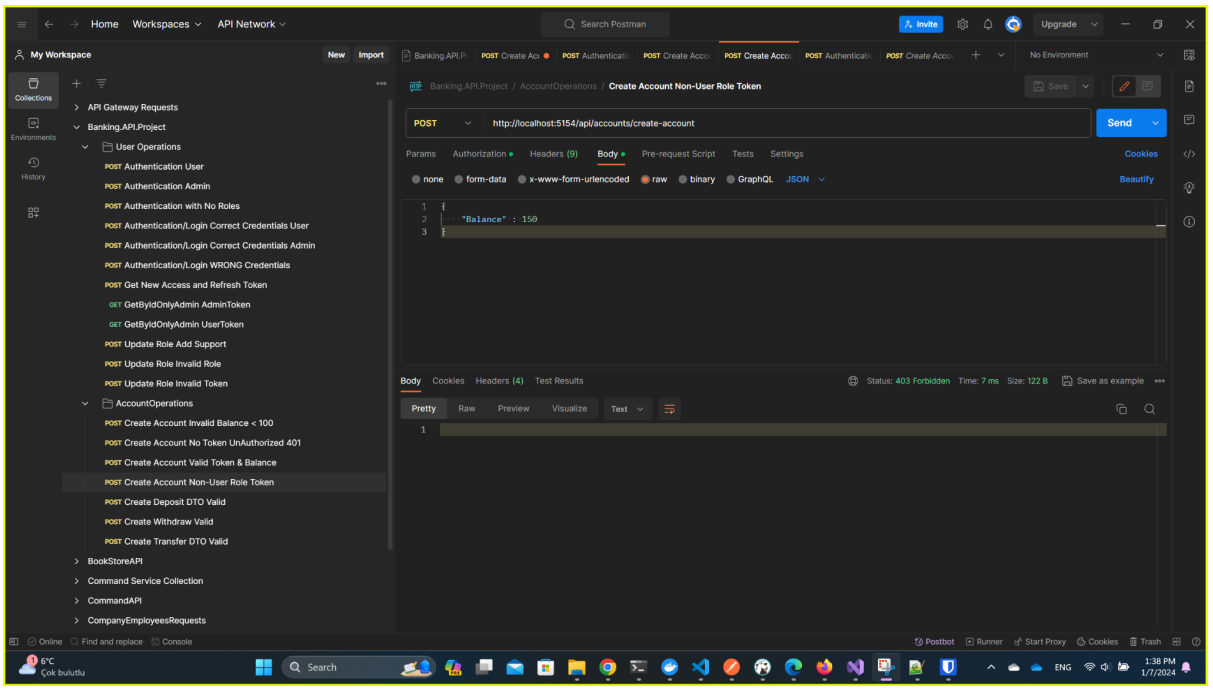


Account Operations

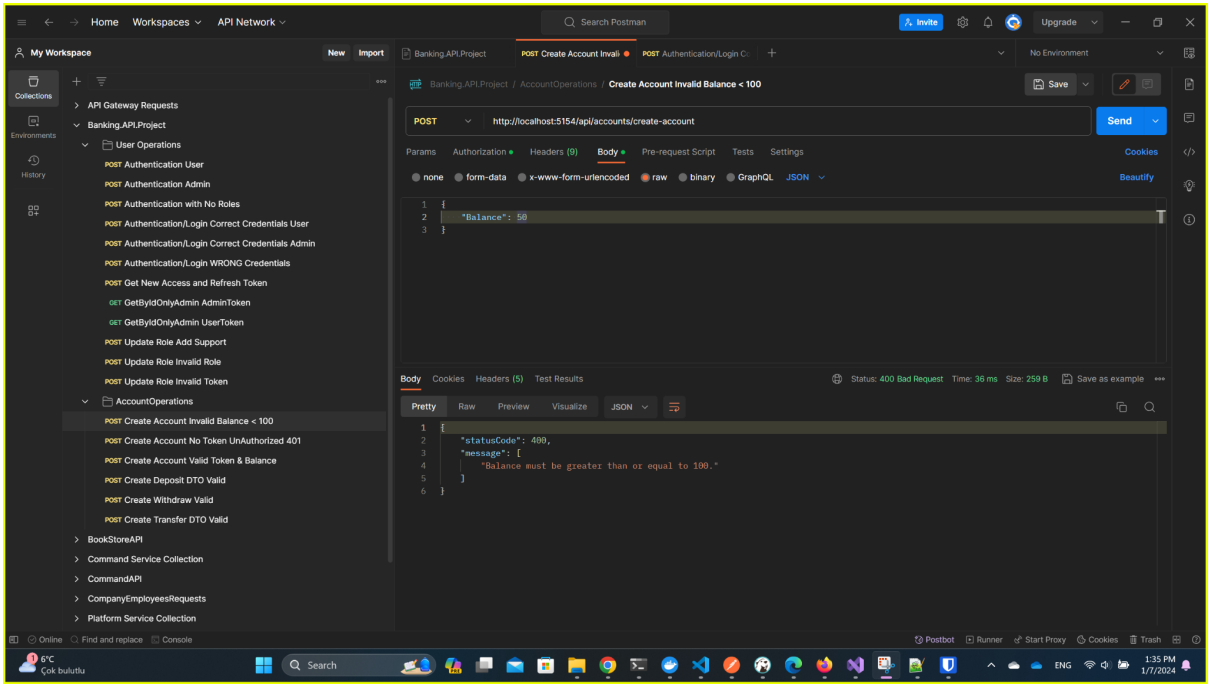
Create Account:

Only the “User” role can create an account. Others roles will get 403 forbidden. Requests that have no access token will get 401 unauthorized.

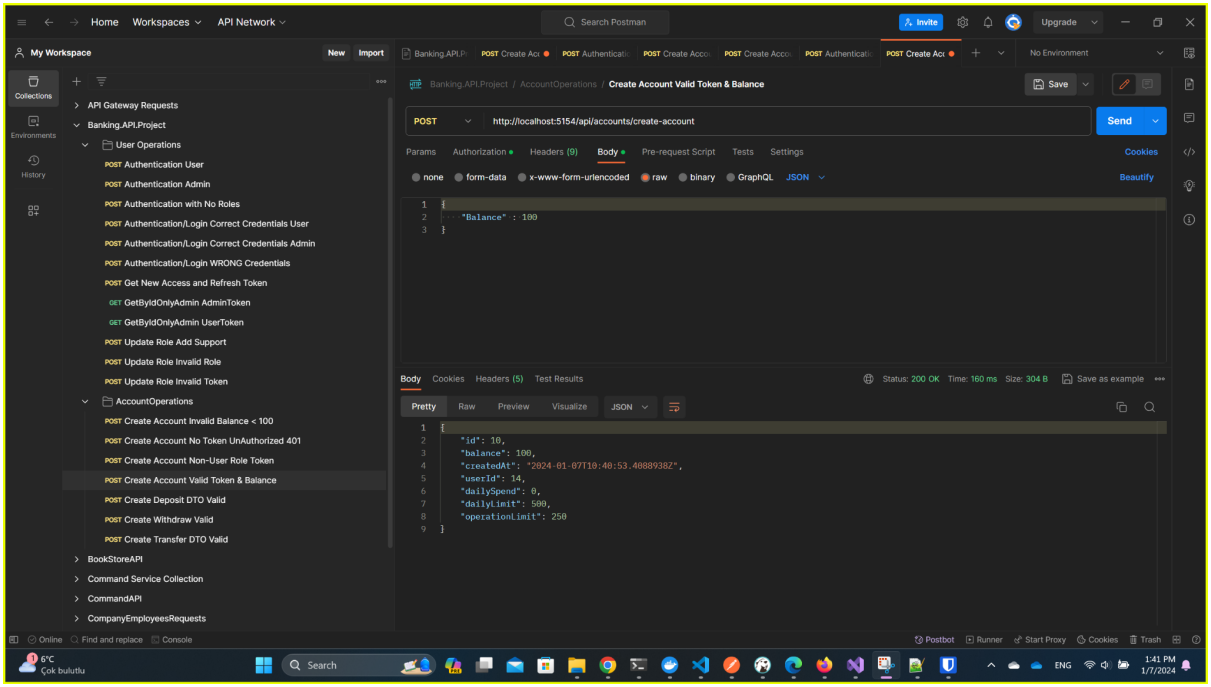




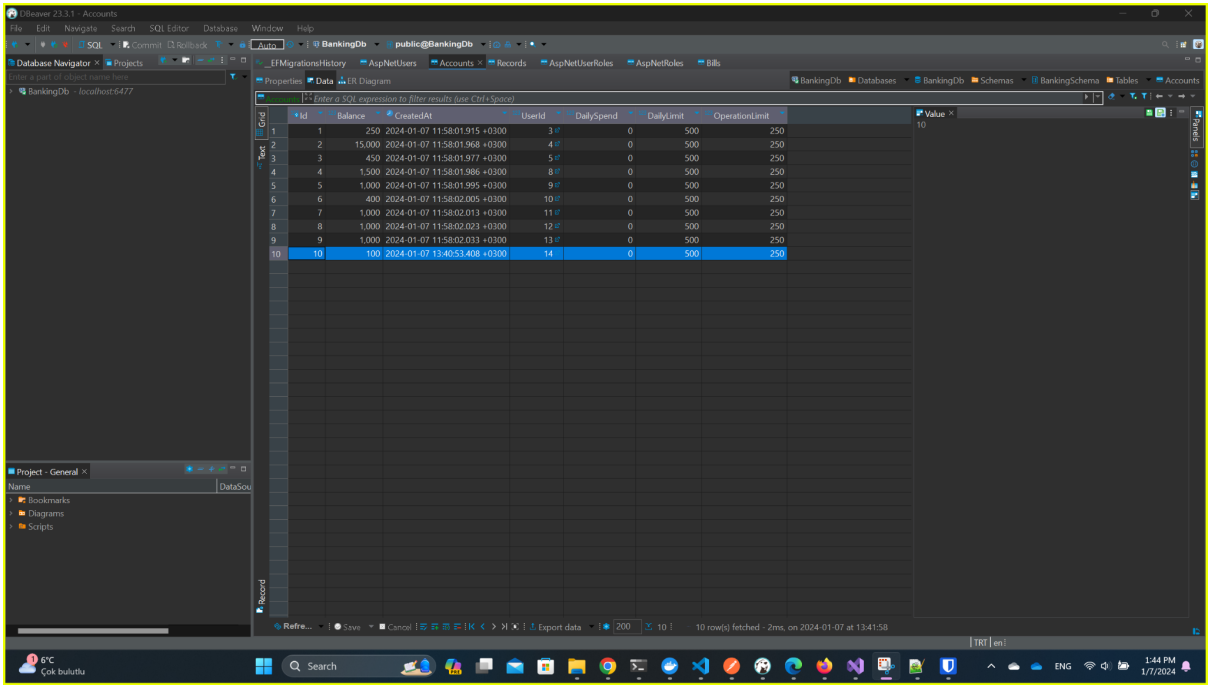
Users will then have to provide a valid account balance. Balance limit is 100. Below the limit will get 400 bad request.

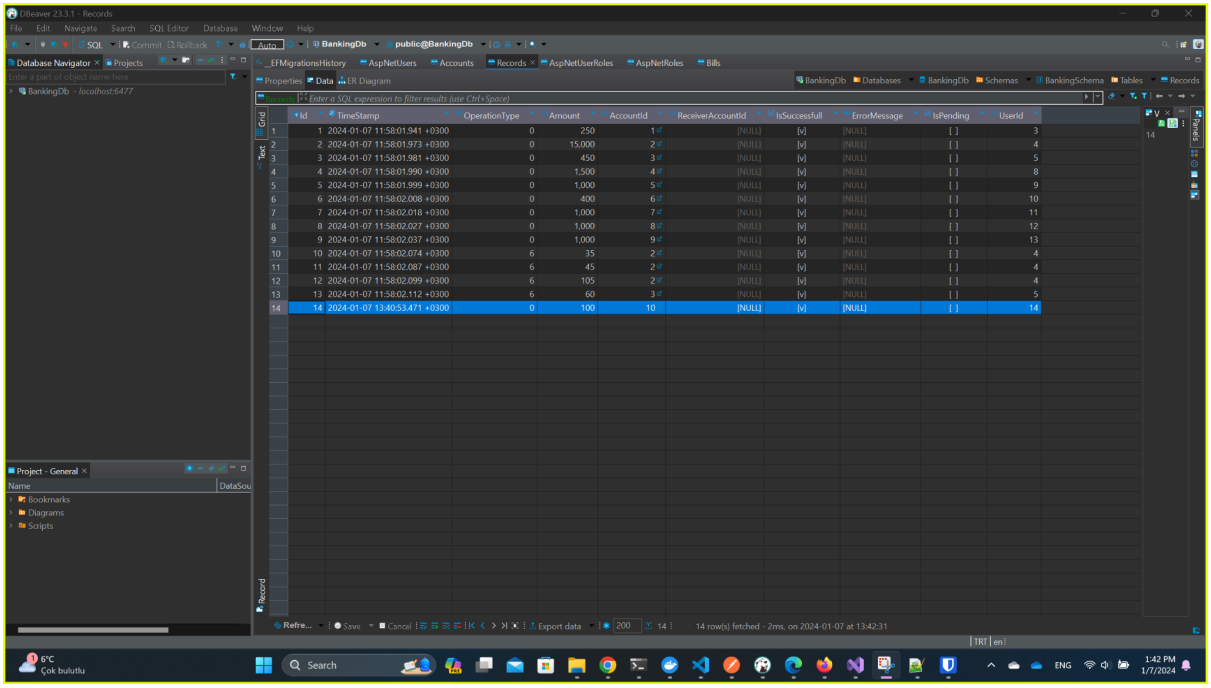


If all is ok, the created account DTO will be returned to the client.



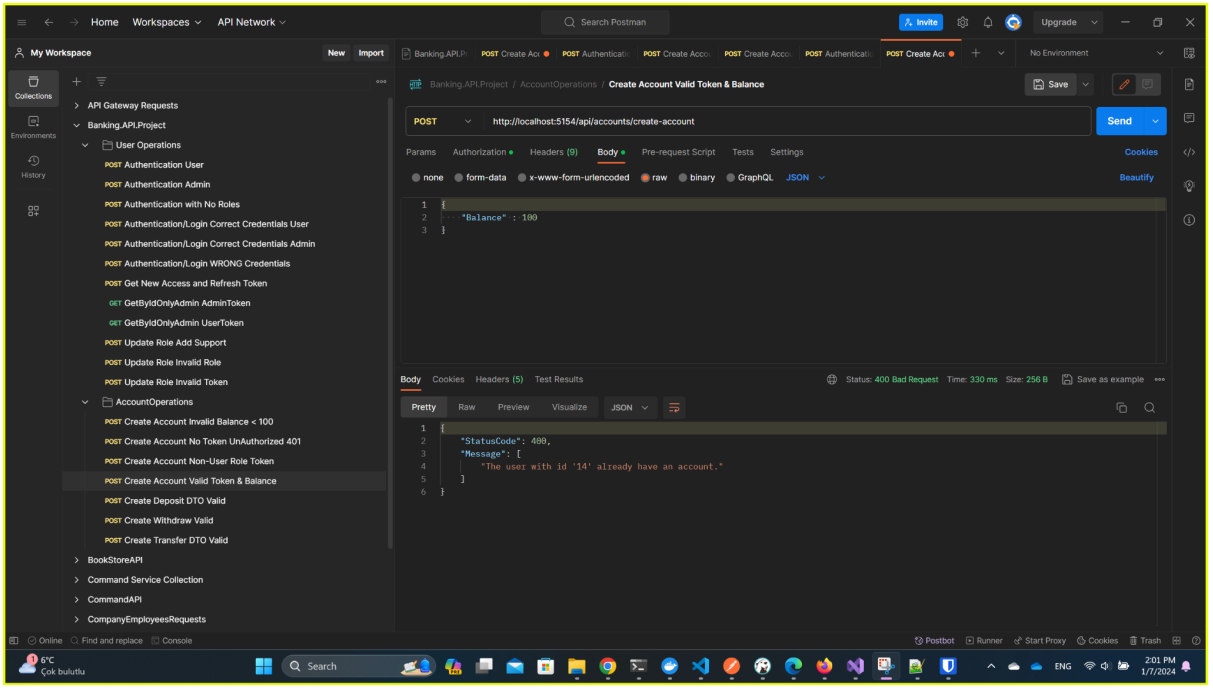
Verifying in the db.





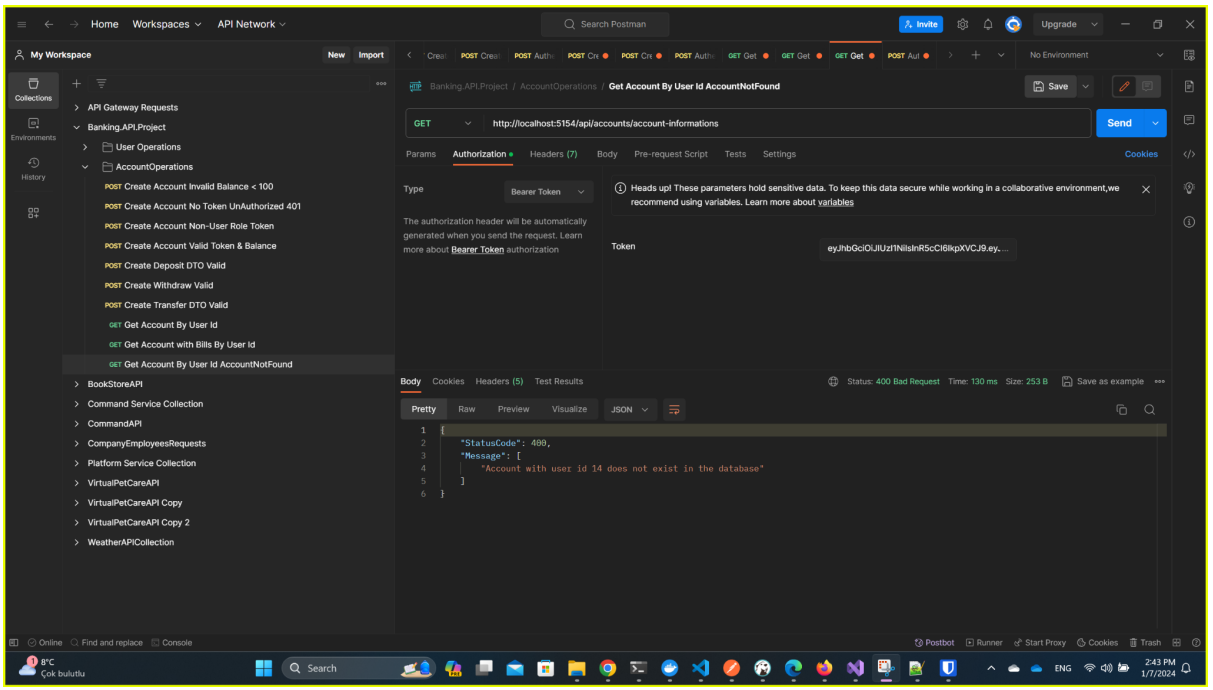
The record has been created too. OperationType 0 means CreateAccount. Its an enum value.

For this project, a user can only have an account.

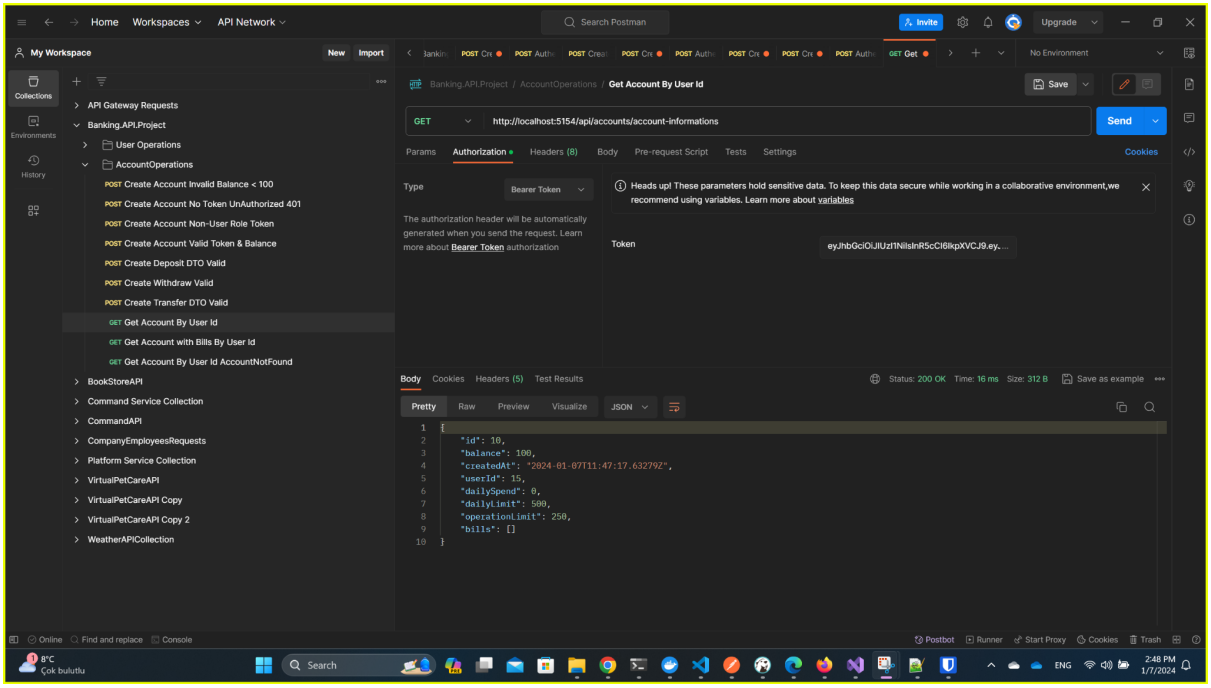


Get Account By User Id:

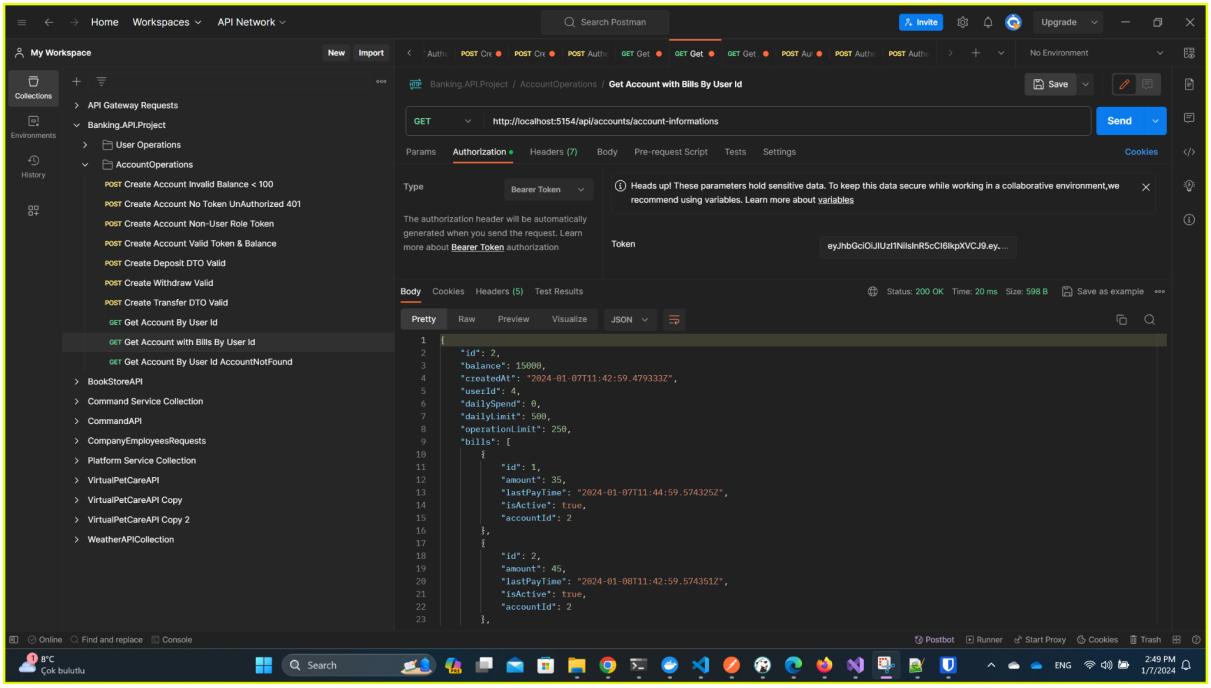
User has to have an account created before. Else, it will return 404 bad request.



It will return account informations.

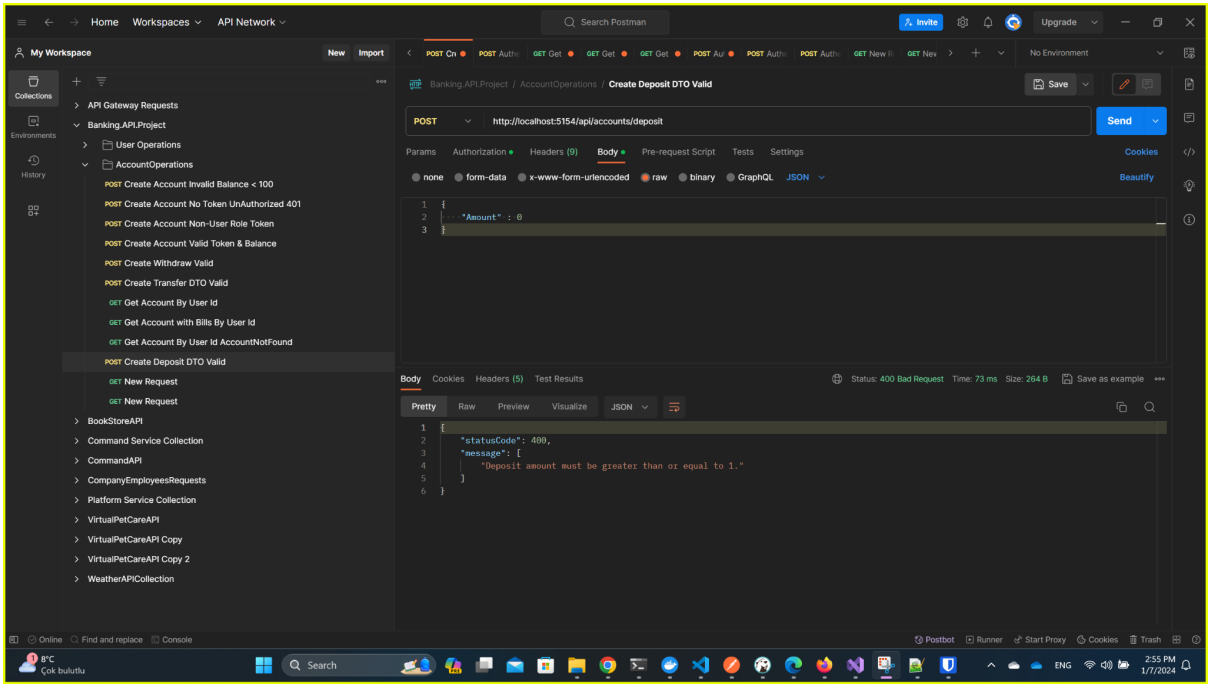


Also if the user has bills, they will return too.

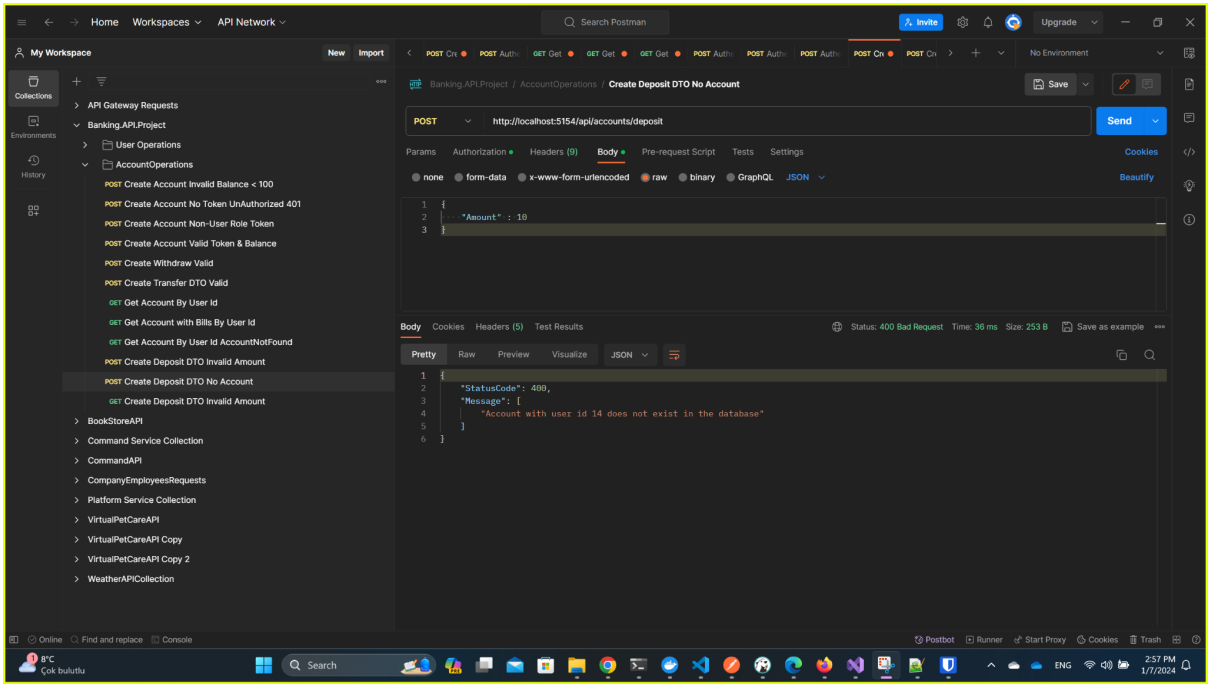


Deposit:

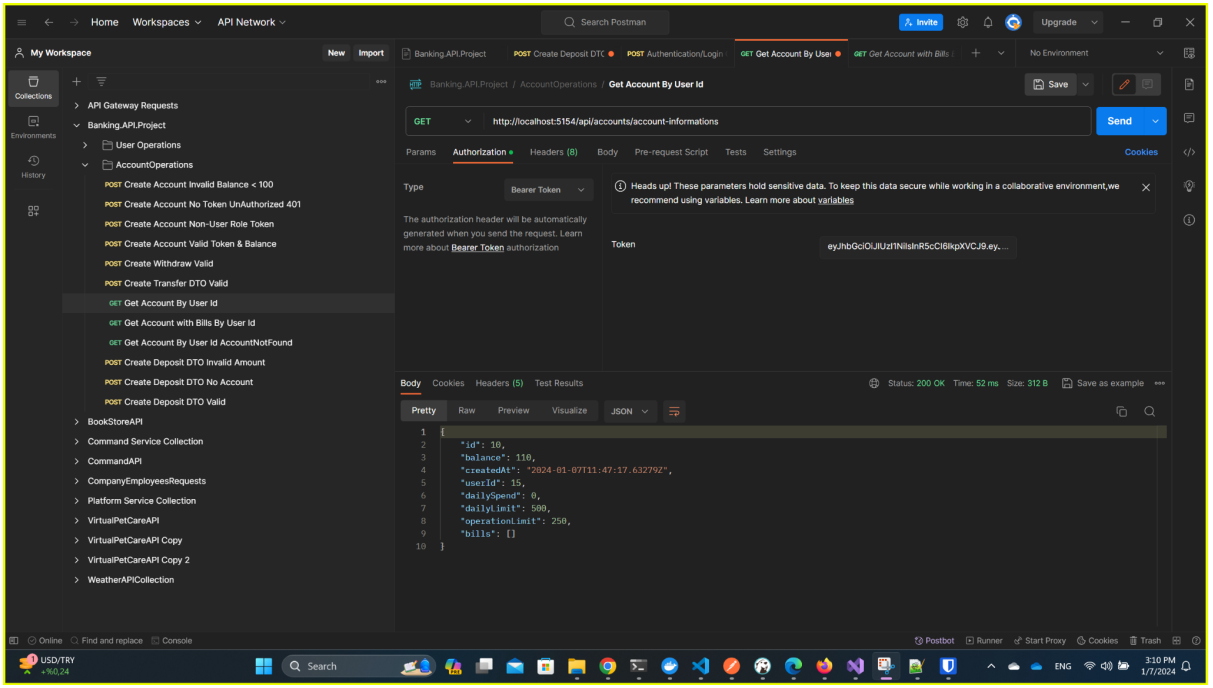
Deposit amount has to be greater than 1.



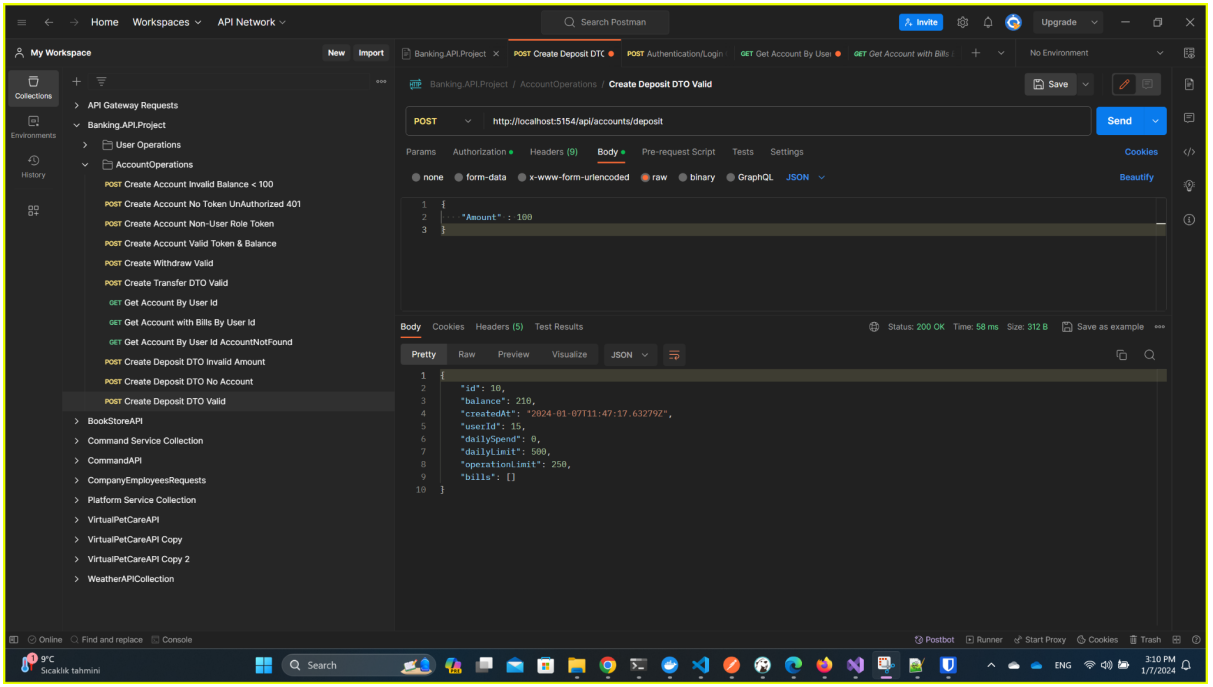
If a user has not created an account before, it will return bad request.



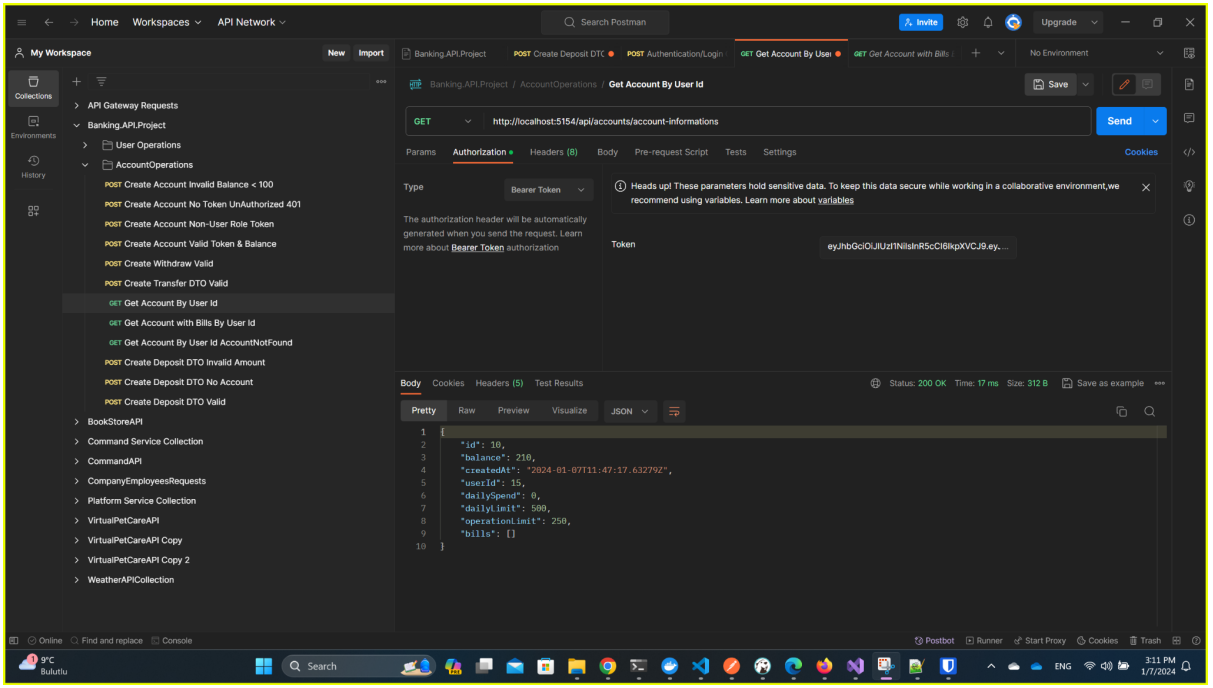
Before a valid deposit request:



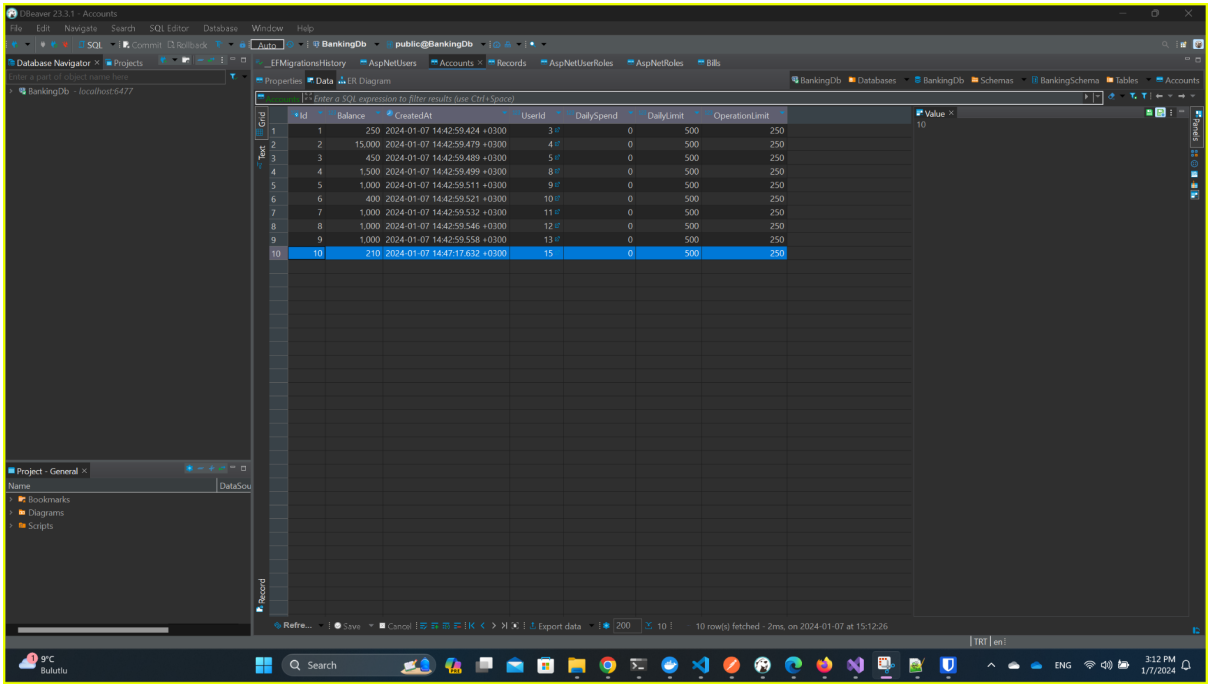
Response of deposit request:



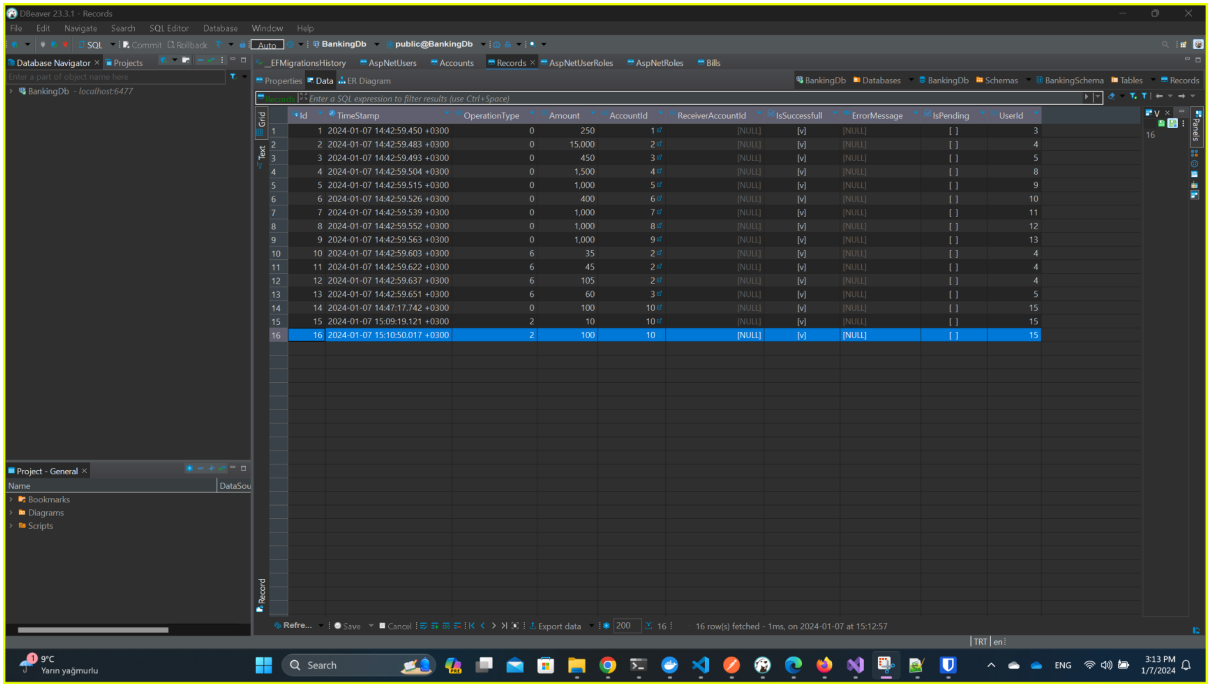
After deposit when getting the same account:



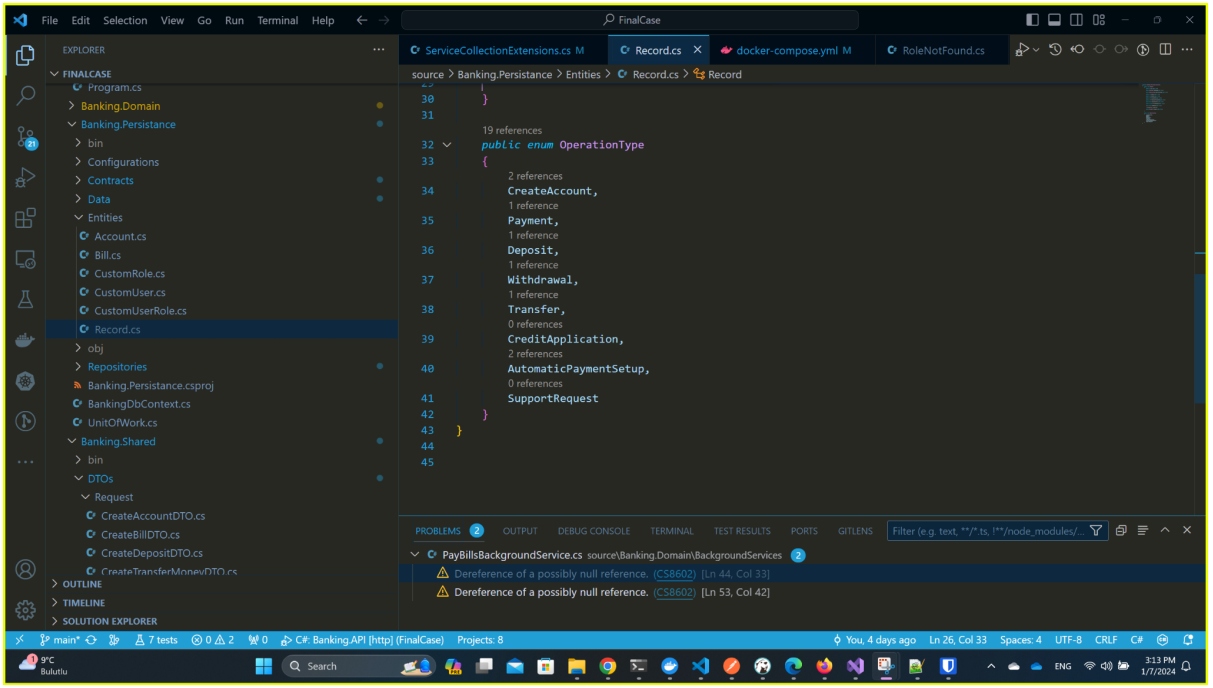
And in the db:



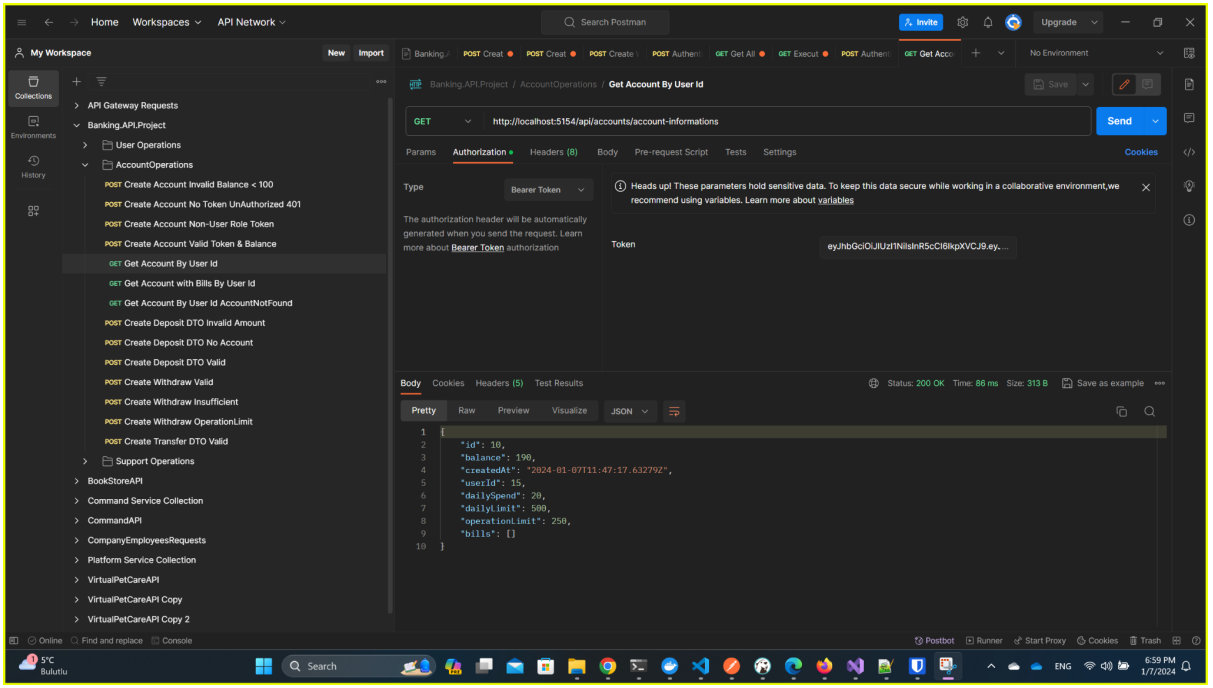
In records table:



Operation Type:

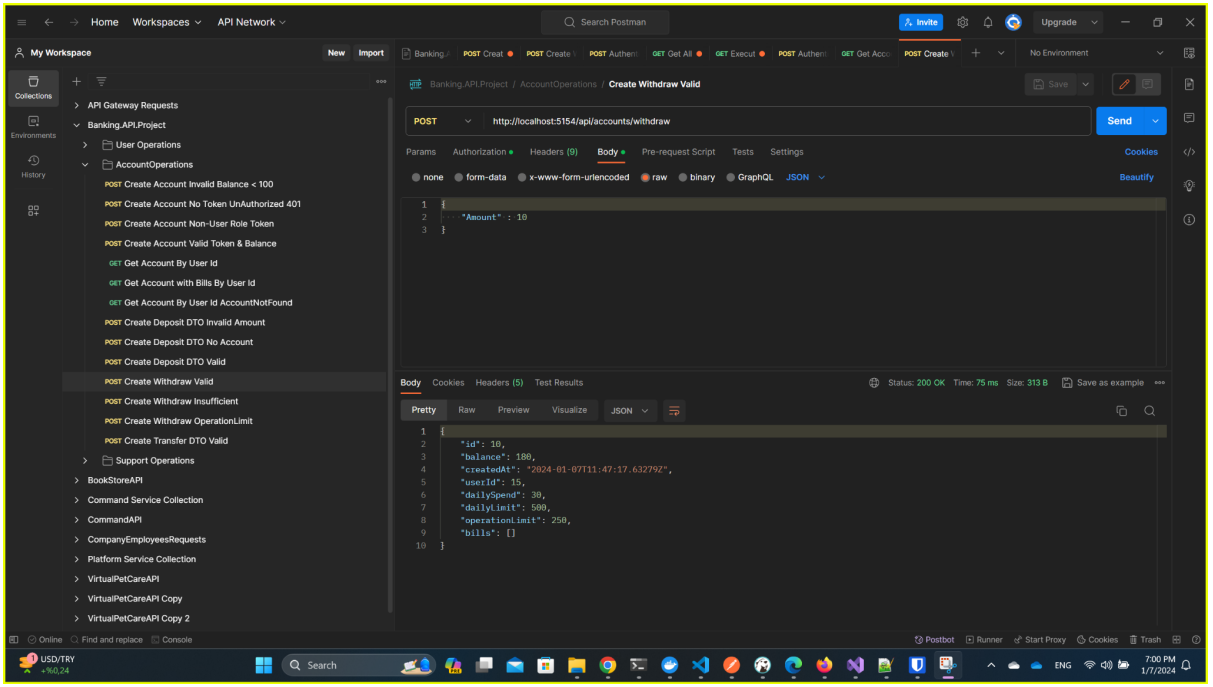


Withdraw:

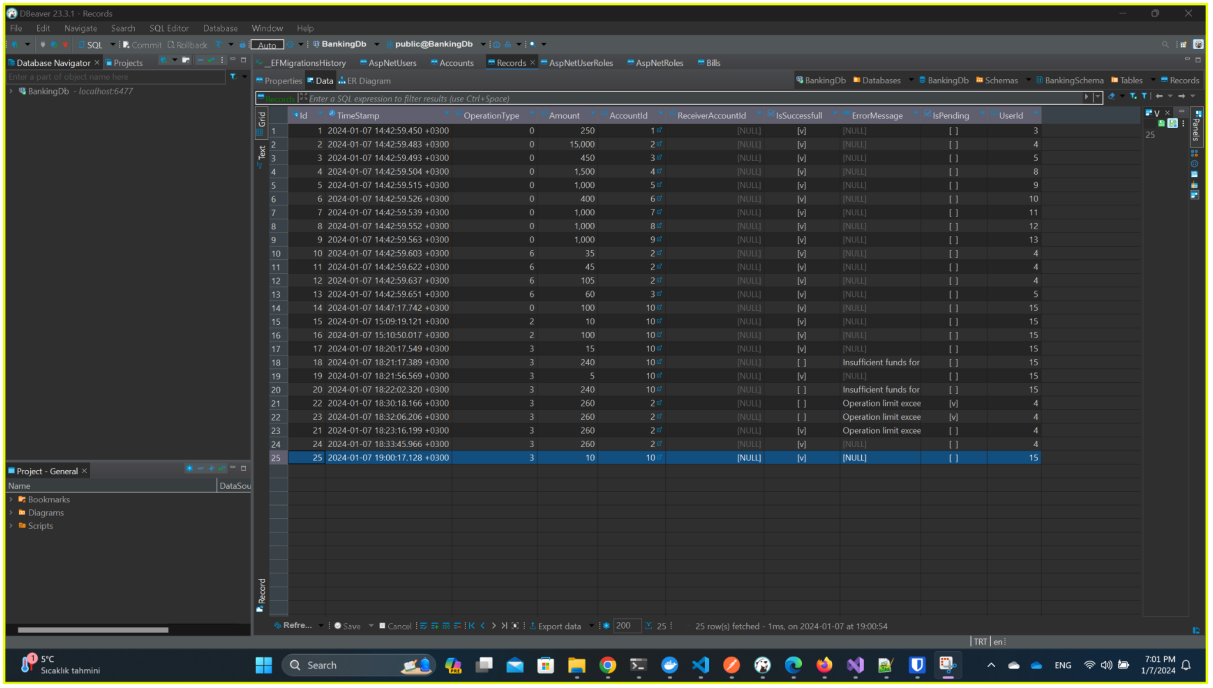


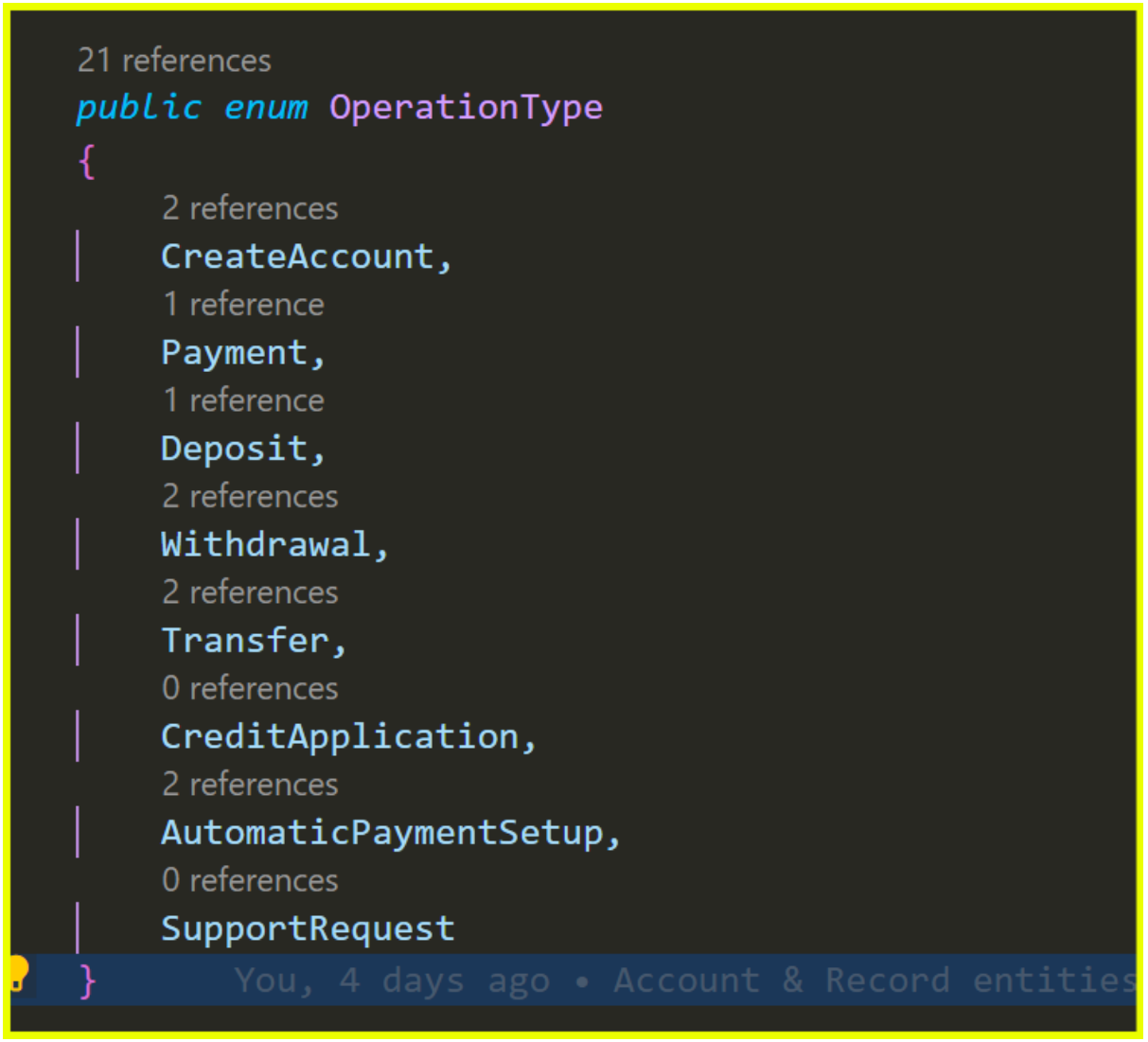
Before withdrawing the amount is 190.

After withdrawing 10 it is now 180.

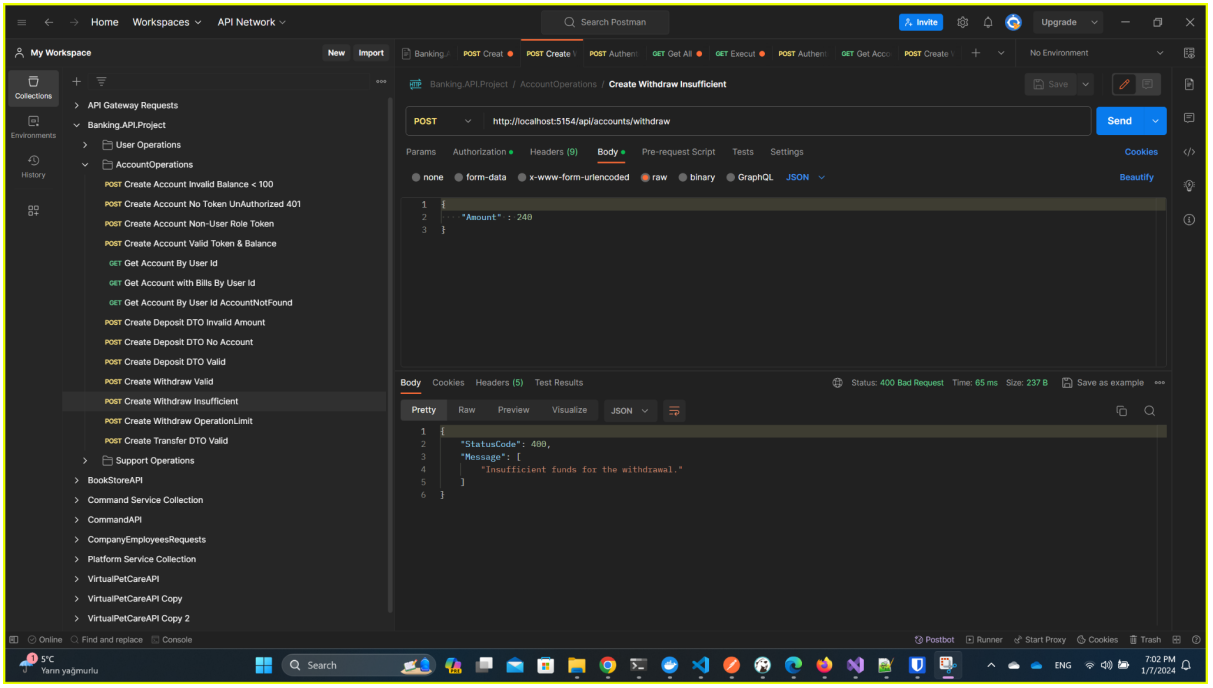


The record is below. OperationType 3 means withdraw.

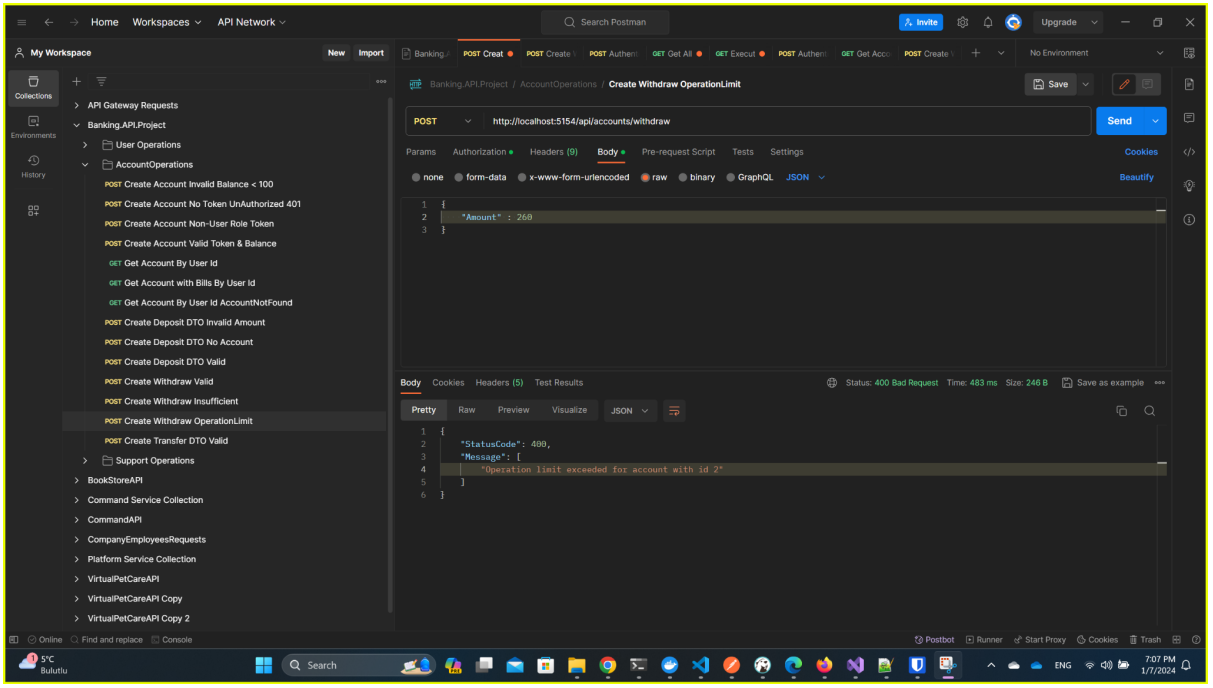



21 references
public enum OperationType
{
2 references
CreateAccount,
1 reference
Payment,
1 reference
Deposit,
2 references
Withdrawal,
2 references
Transfer,
0 references
CreditApplication,
2 references
AutomaticPaymentSetup,
0 references
SupportRequest
}
You, 4 days ago • Account & Record entities

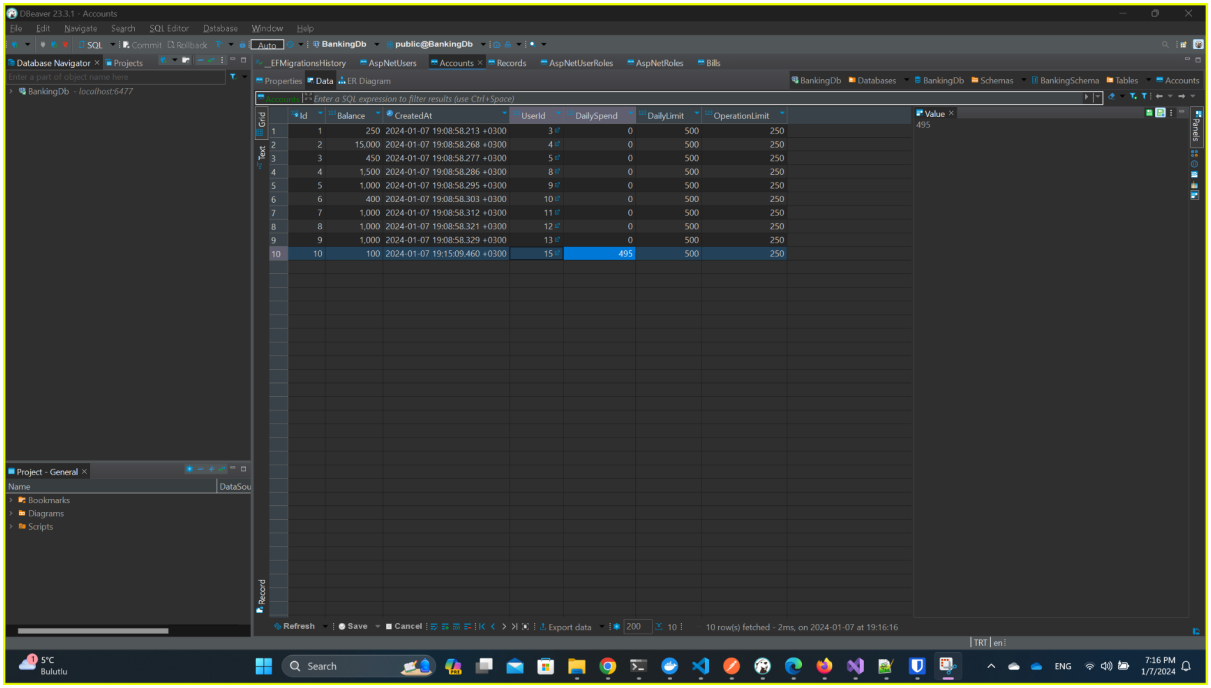
If balance is less than amount: returns a bad request.

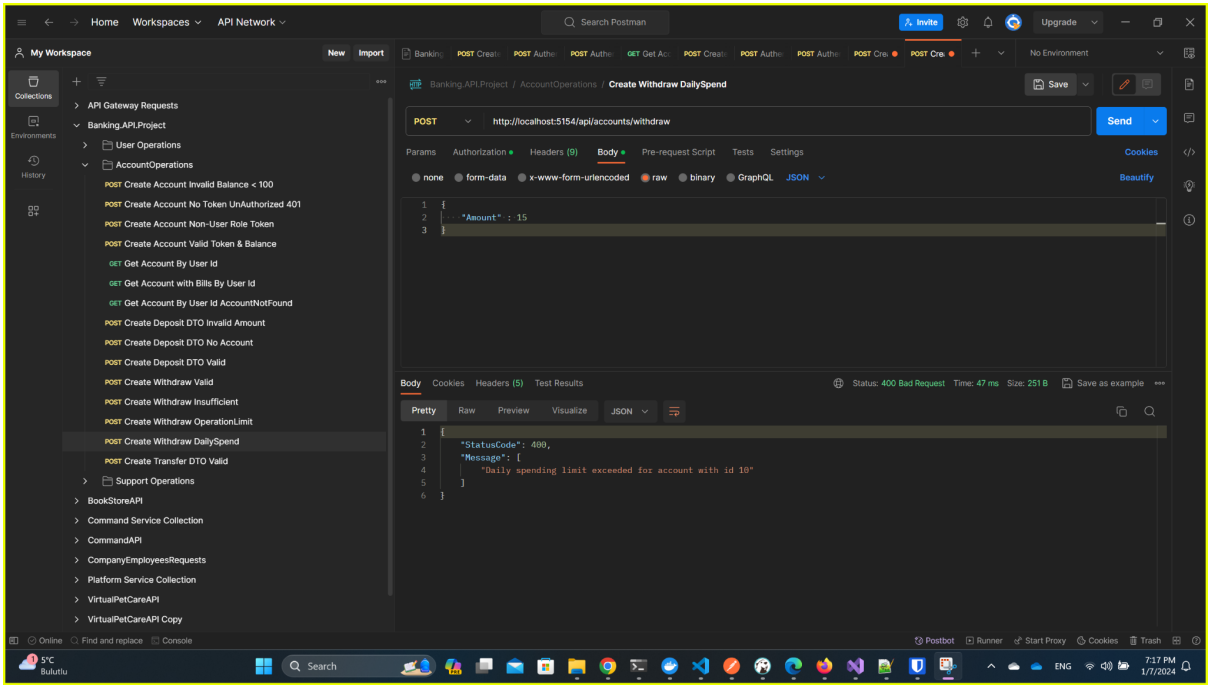


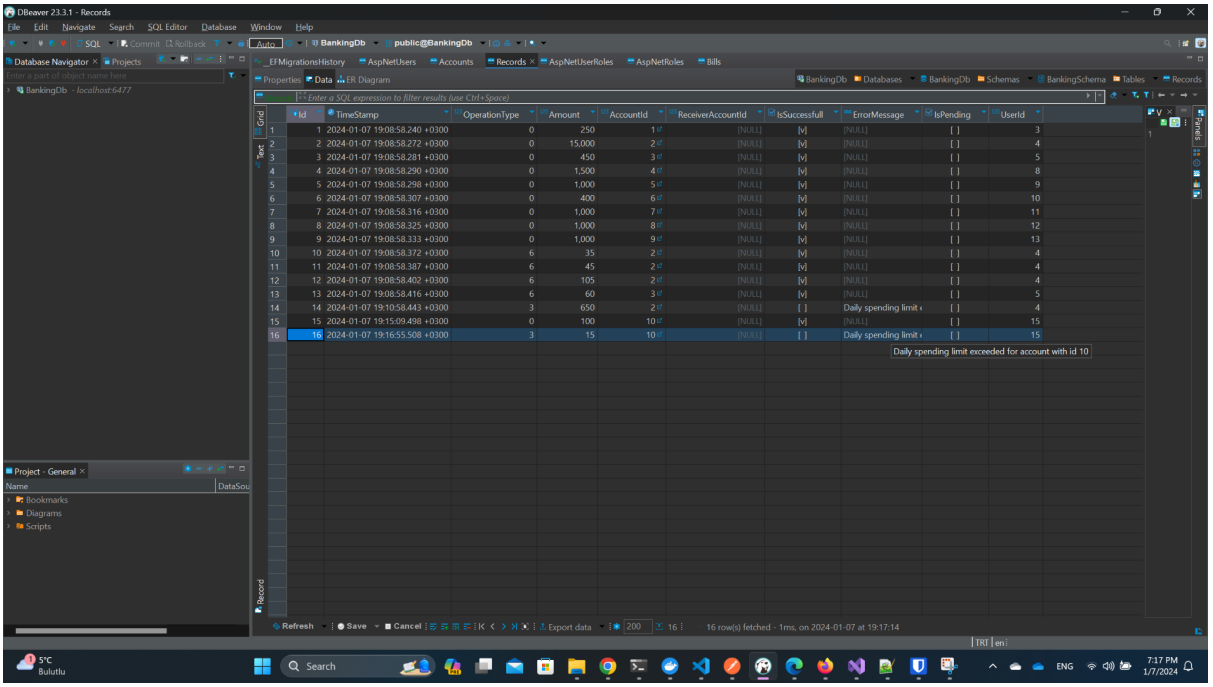
If the amount exceeds OperationLimit it returns a bad request.



Editing DailySpend value for account id 10. Now sending the same request.



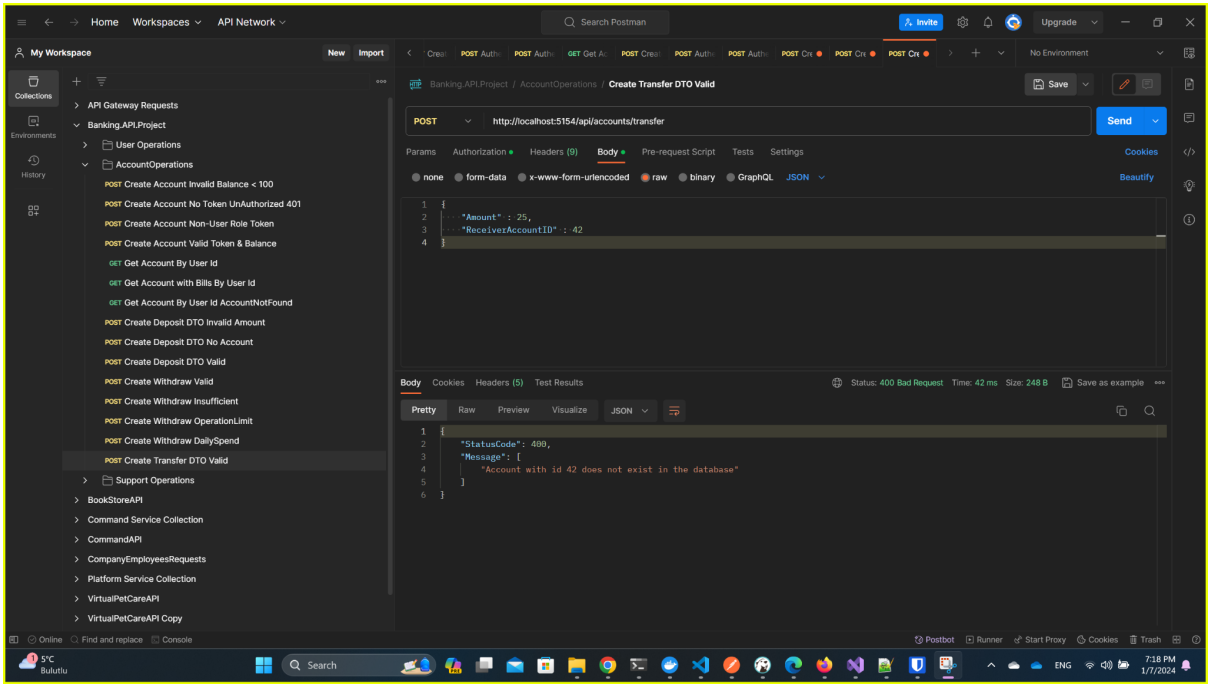


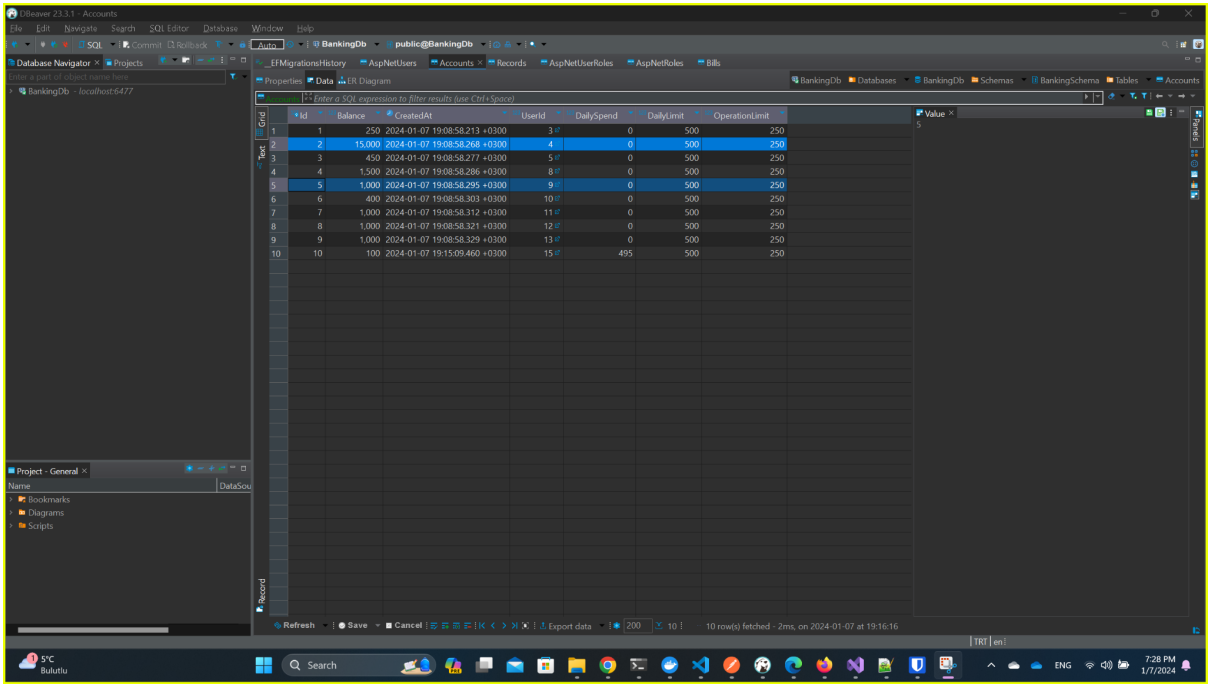


Record has been created too with an error message.

Transfer:

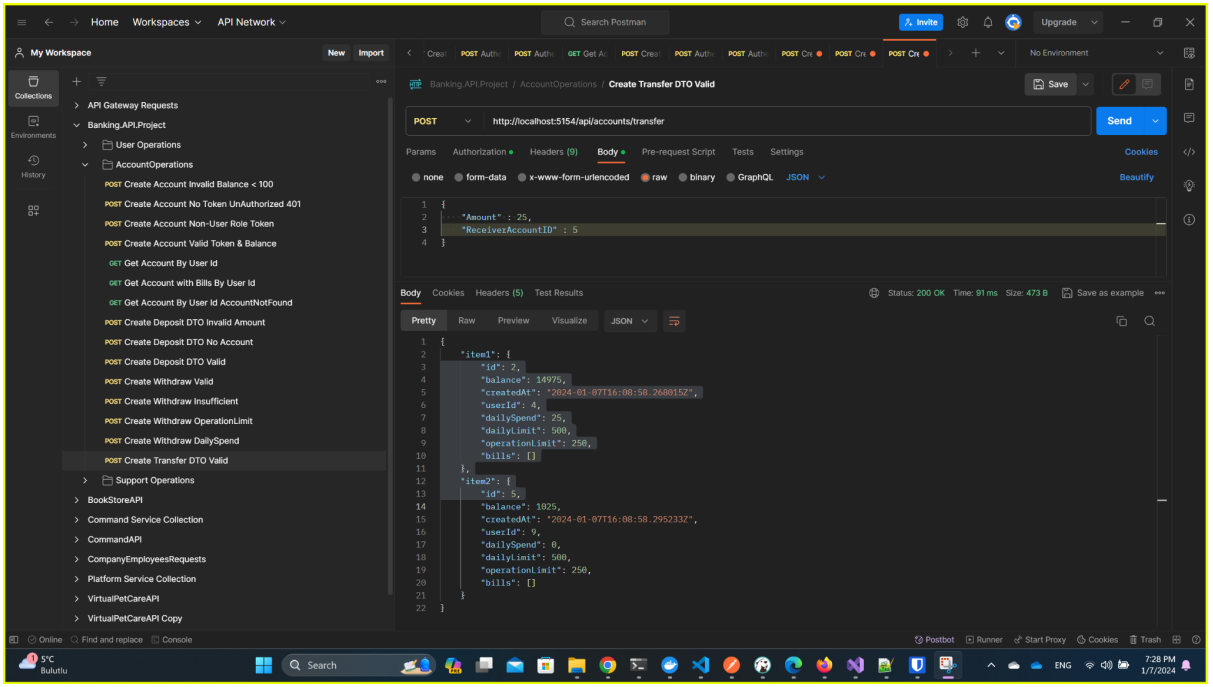
If the receiver account id does not exist.



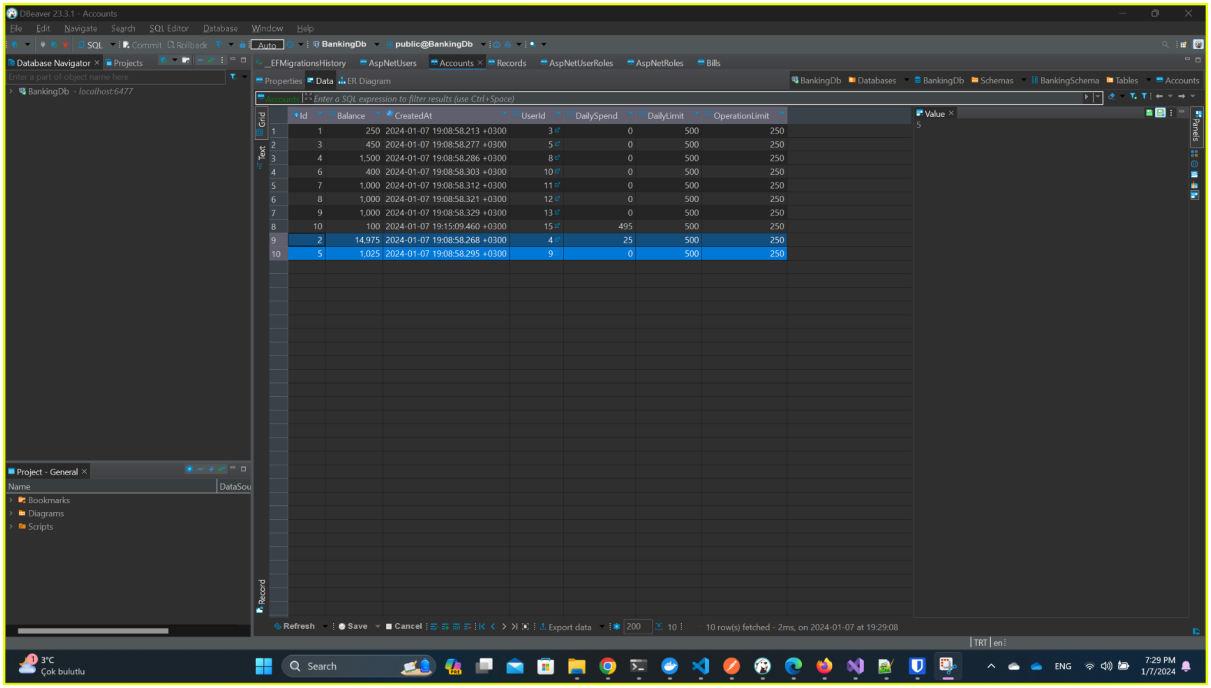


Before sending a valid transfer from account 2 to account 5.

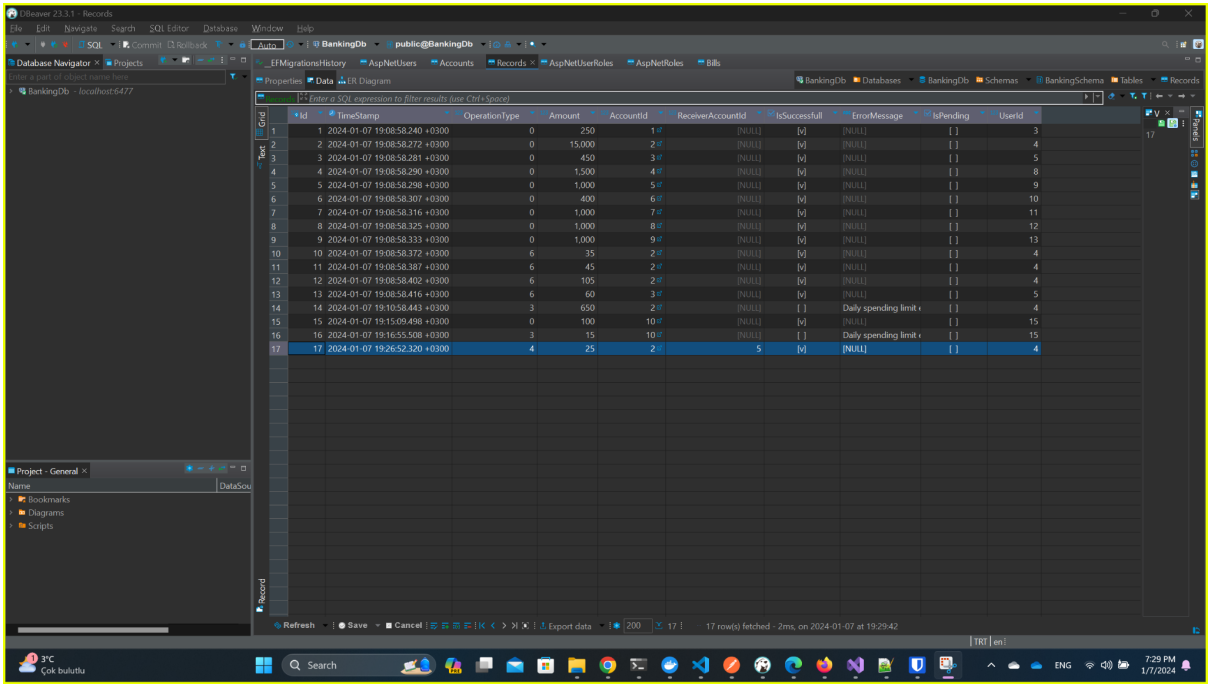
After sending a valid transfer:



Updated accounts:



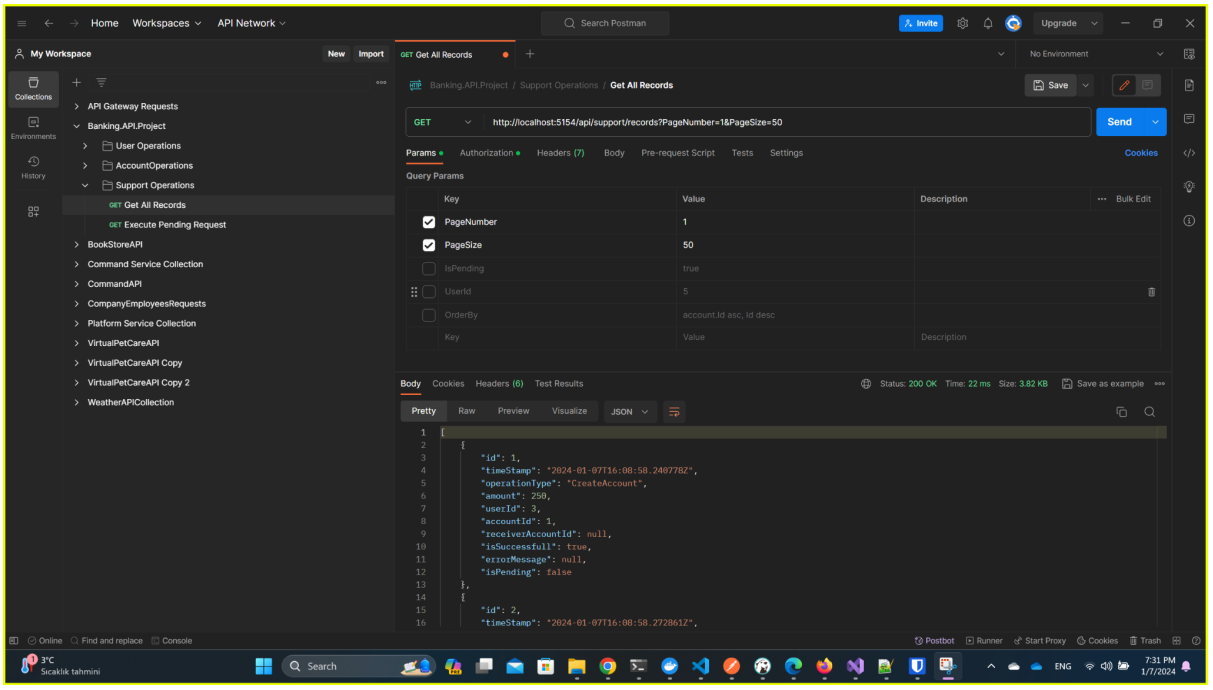
Created record:

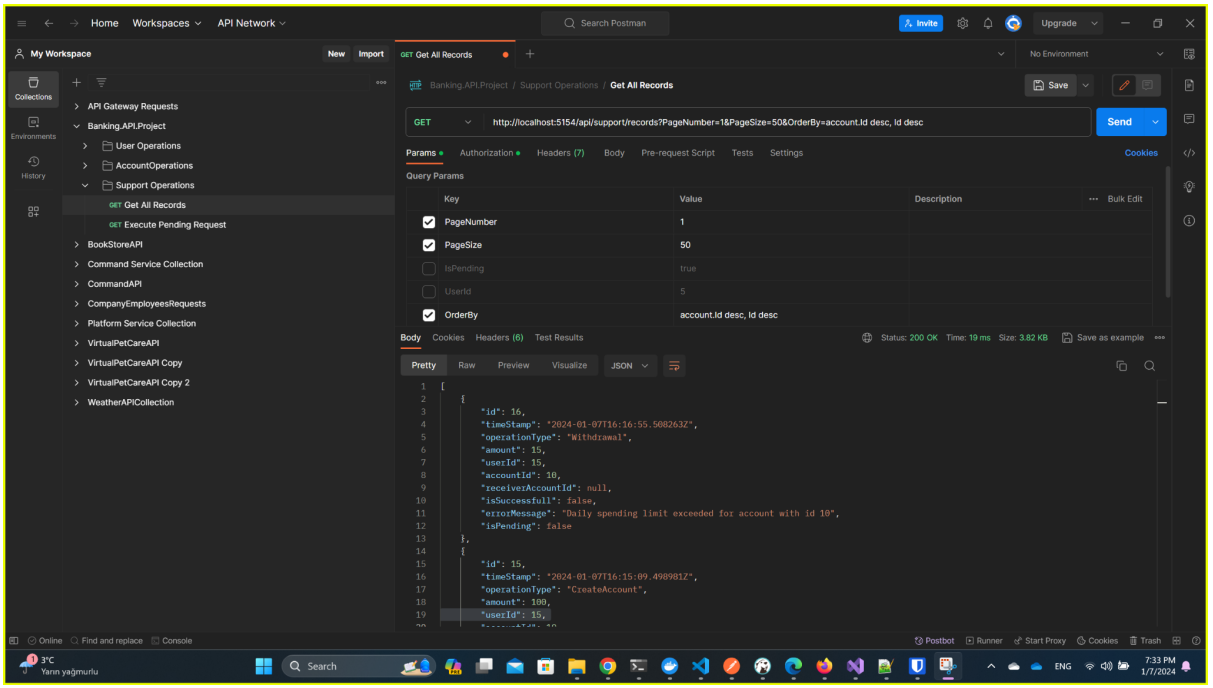


Support Role Operations

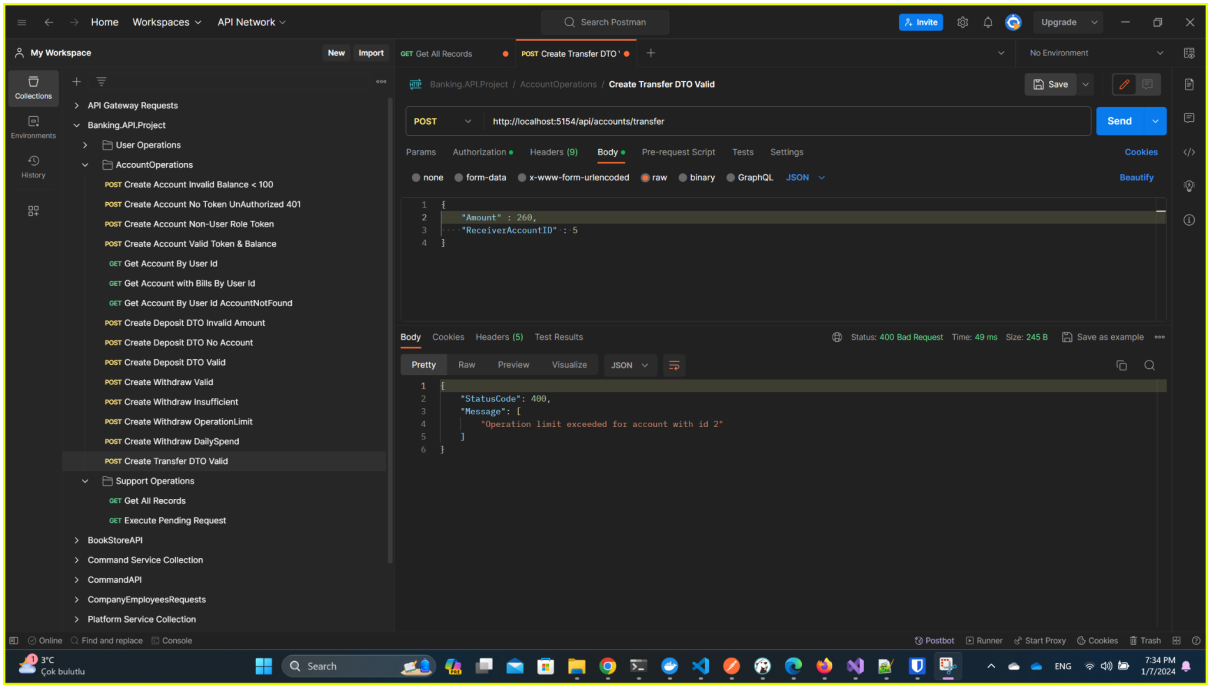
Get All Record By Querying:

Supports can query records. They can get records that are pending, or not pending. They can sort records, or not sort.

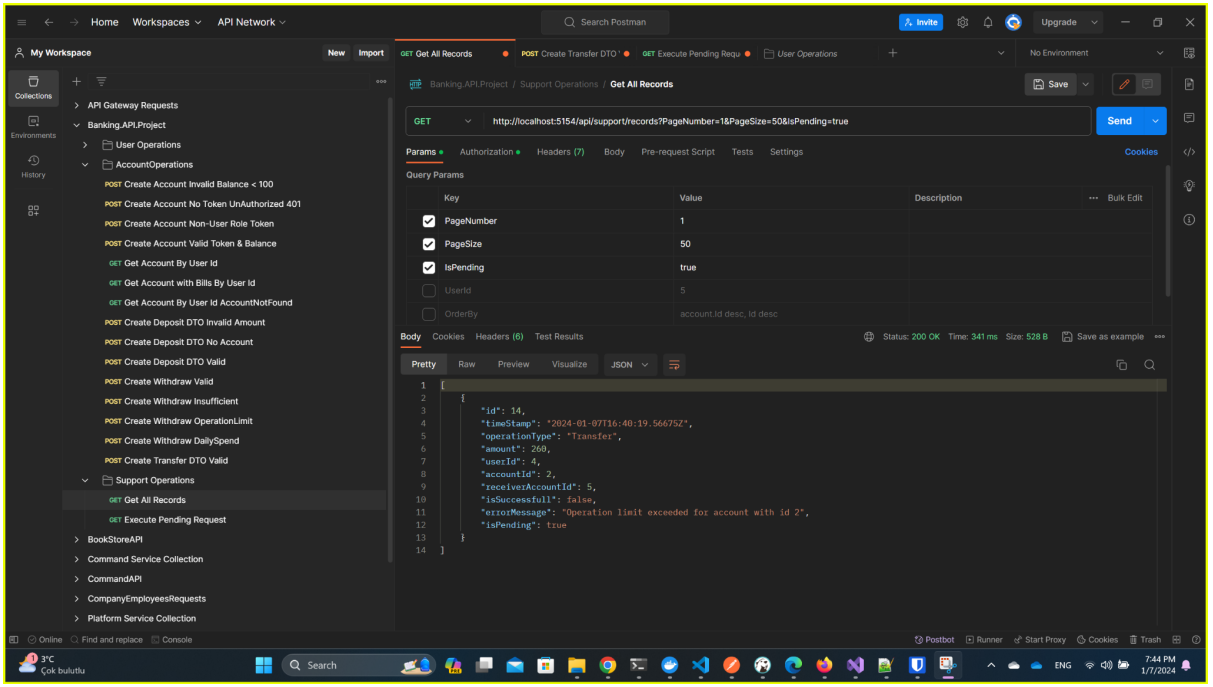




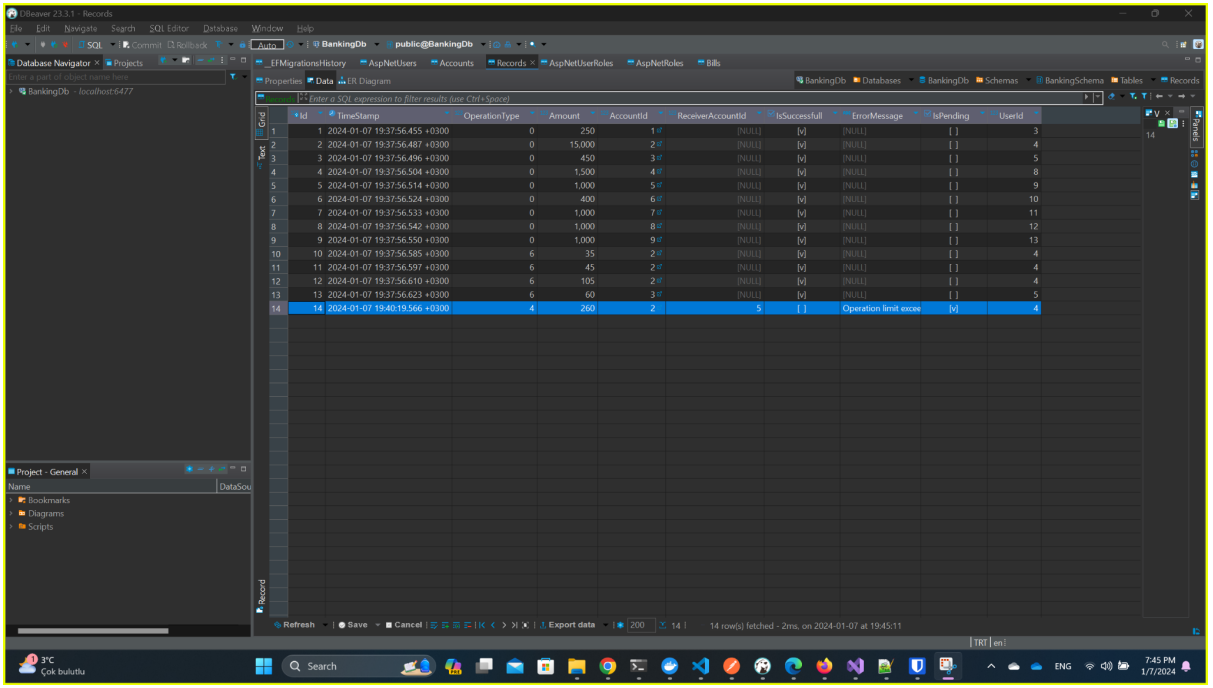
Now sending a transfer request that will exceed the operation limit that is 200.



Support can get this record via querying.

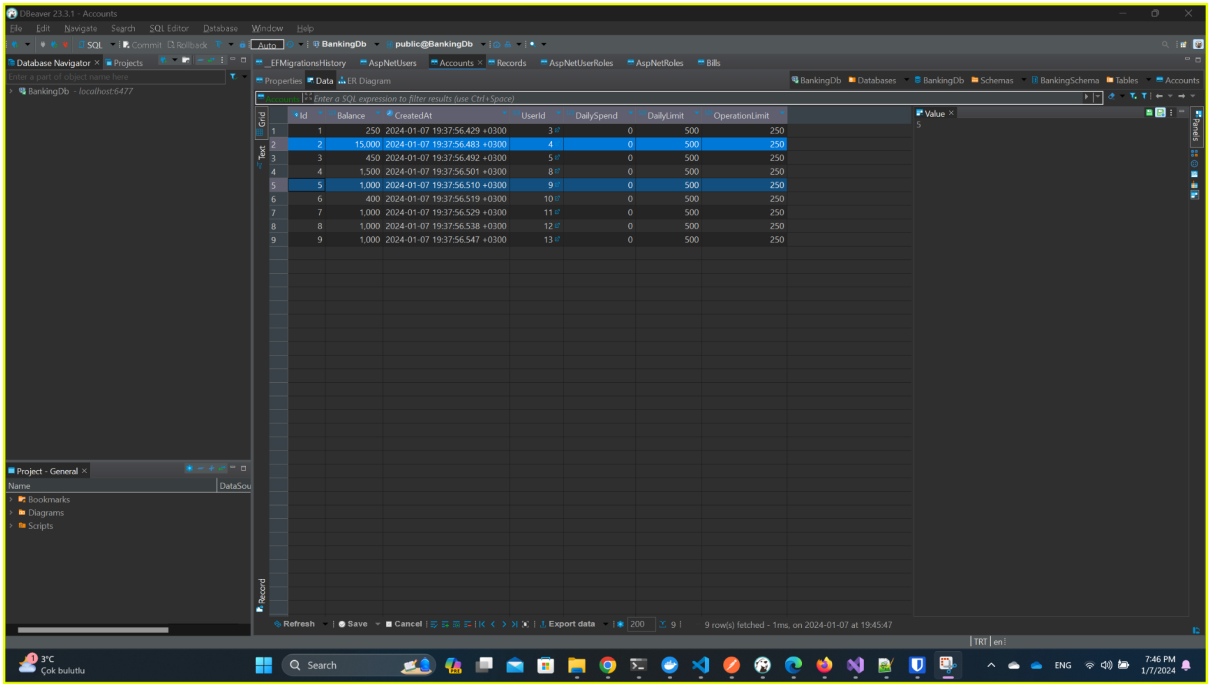


In records table:

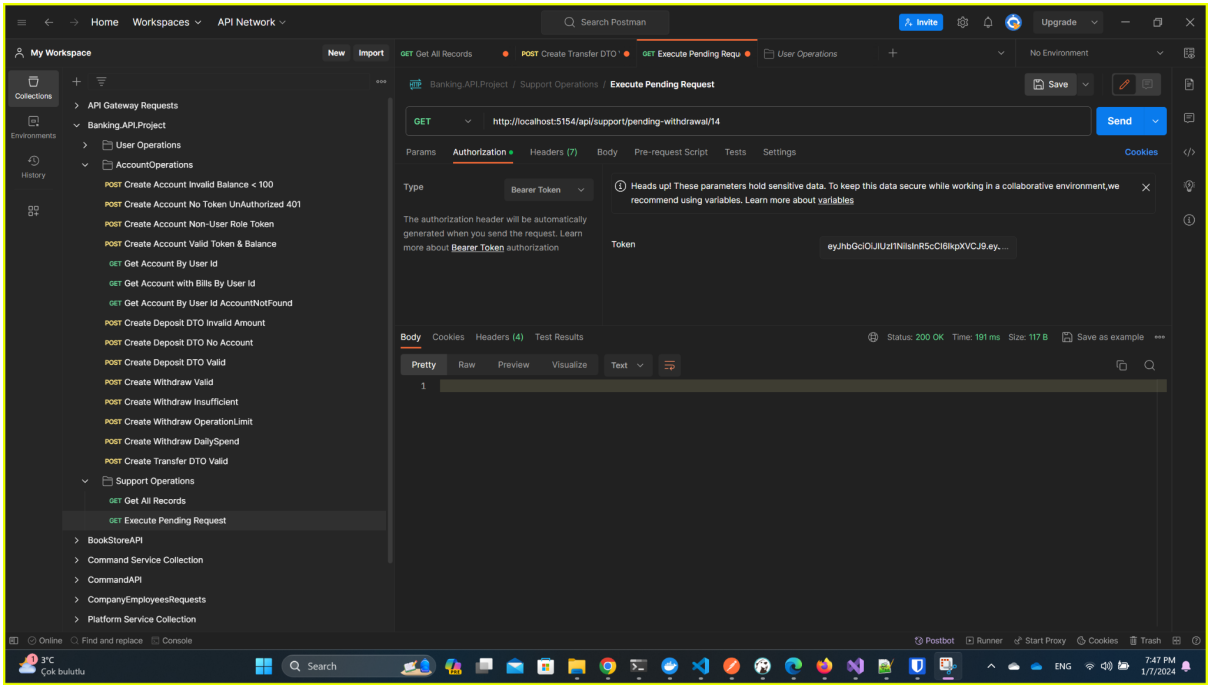


Record with id has been created in the last request. And IsPending = true. Support can execute this transfer that is blocked by the system.

Accounts right before executing transfer by support



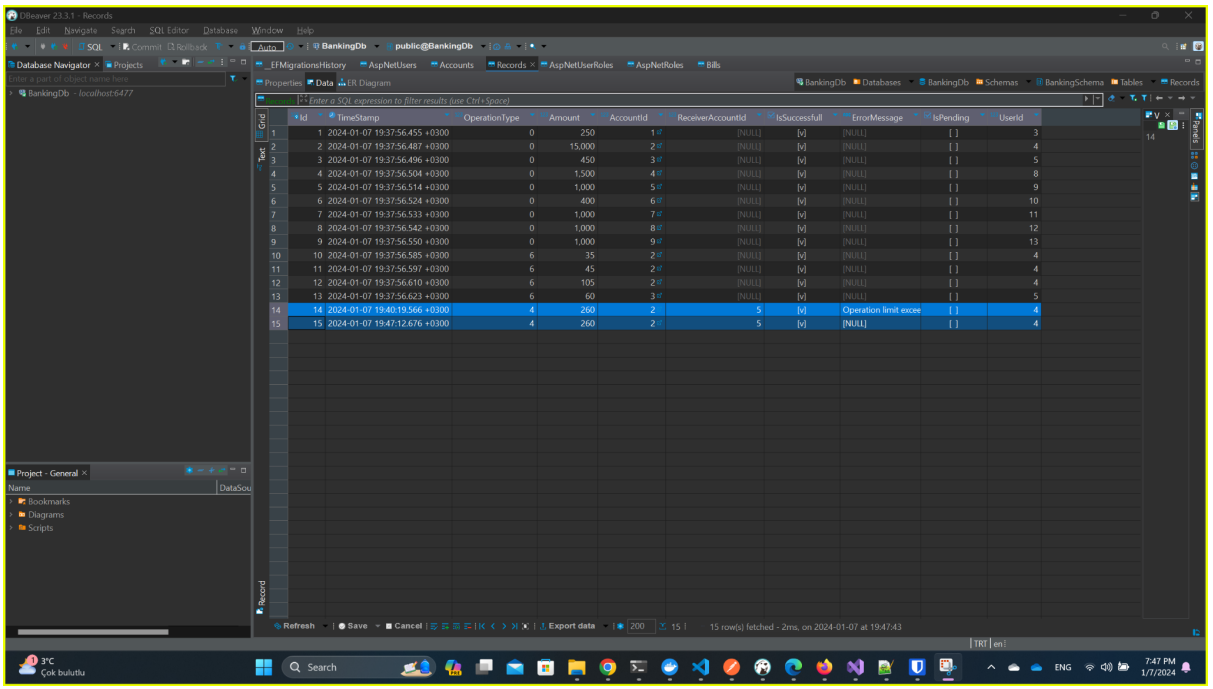
Now executing pending record.



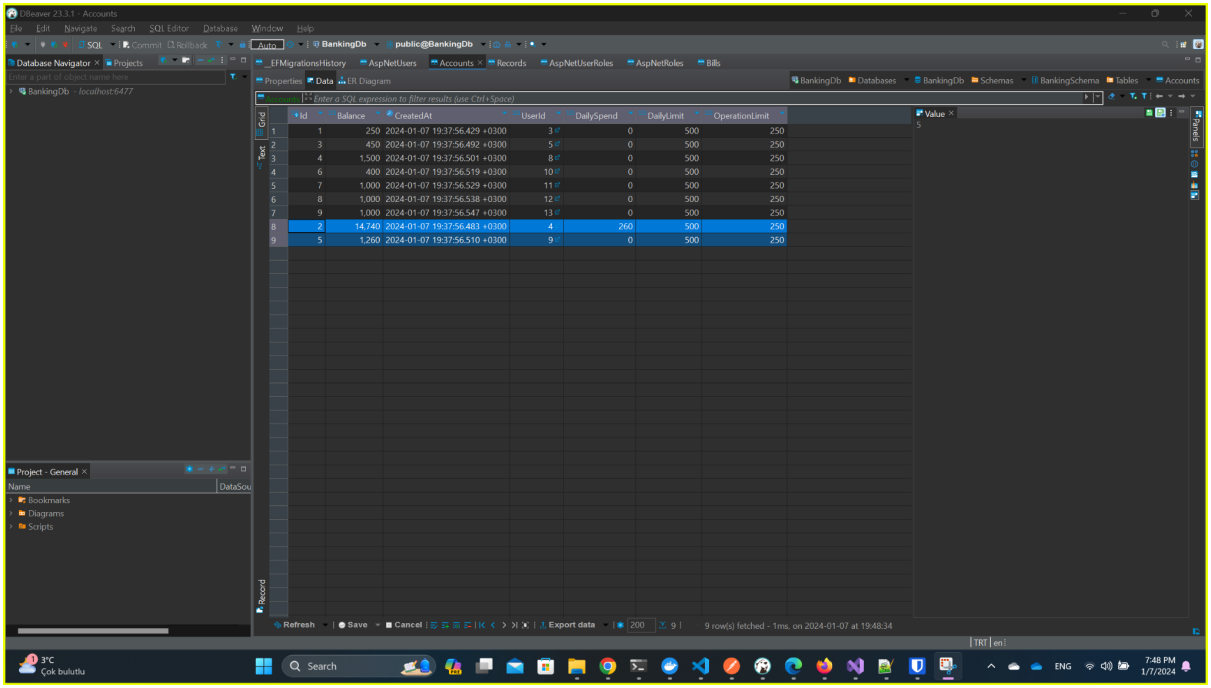
It returned 200.

And in db:

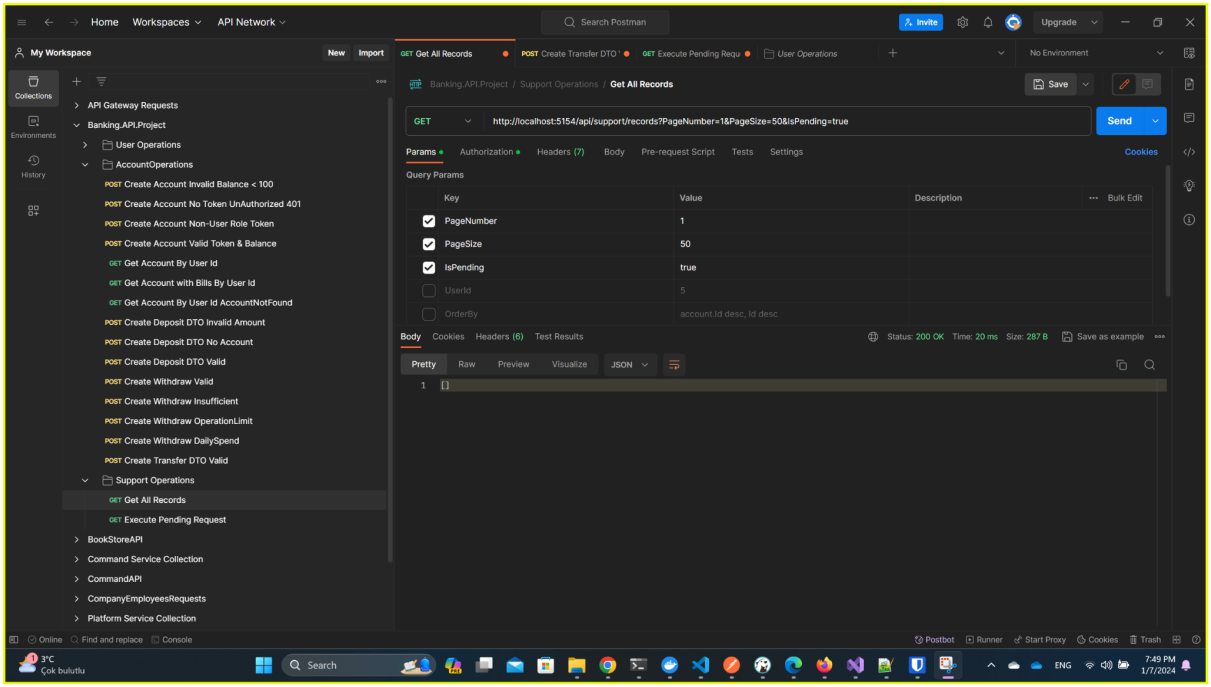
A new record has been created for the transfer. Old record that is pending is now not pending.



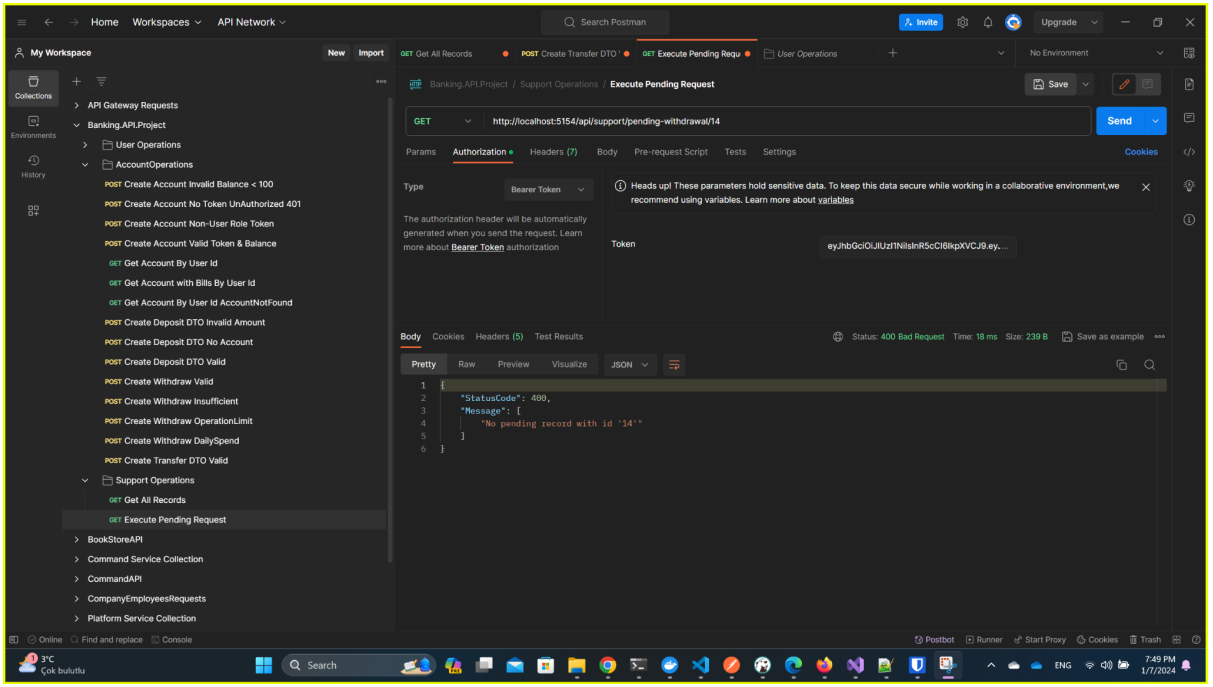
Also Accounts:



And support tries to get pending records:



And if try to execute the same record one more time:

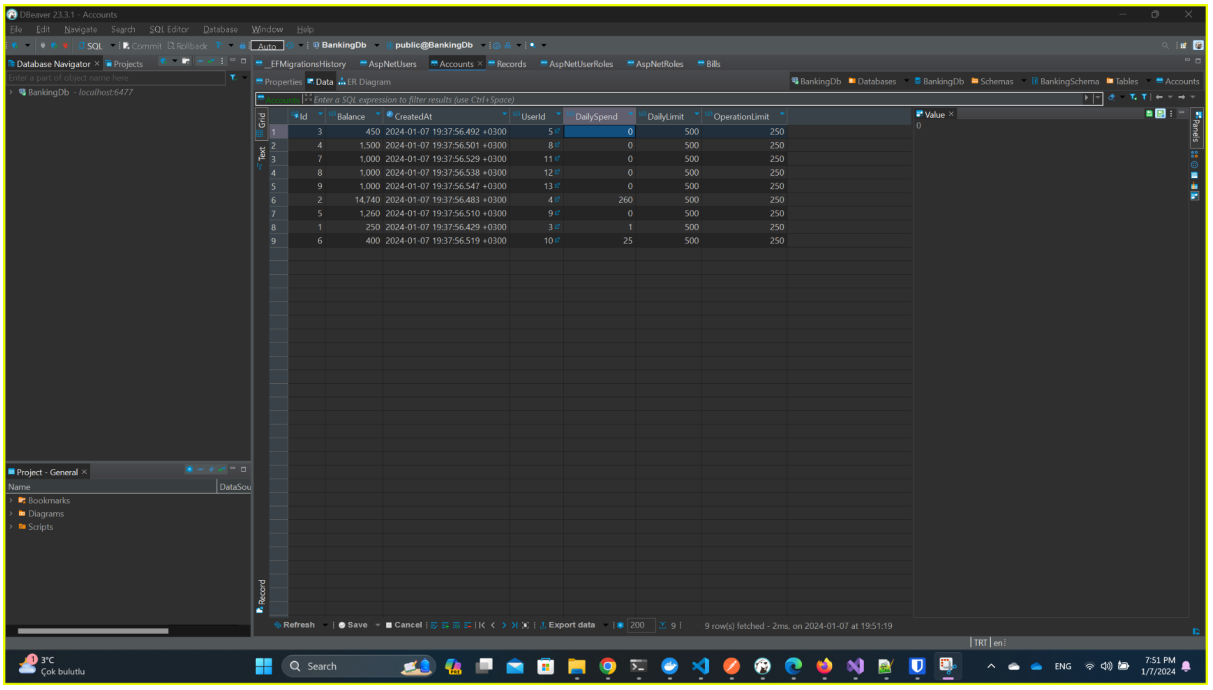


Gets 400 bad request.

Background Services

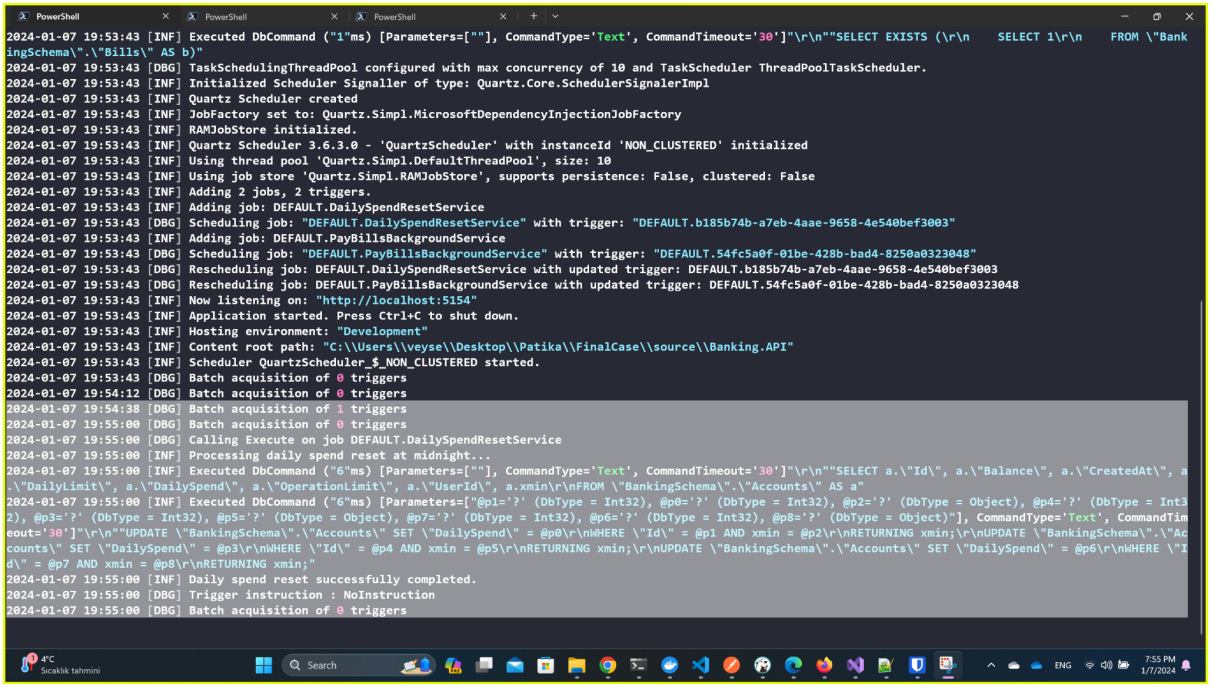
Reset Daily Spend:

Accounts daily spends before background service.

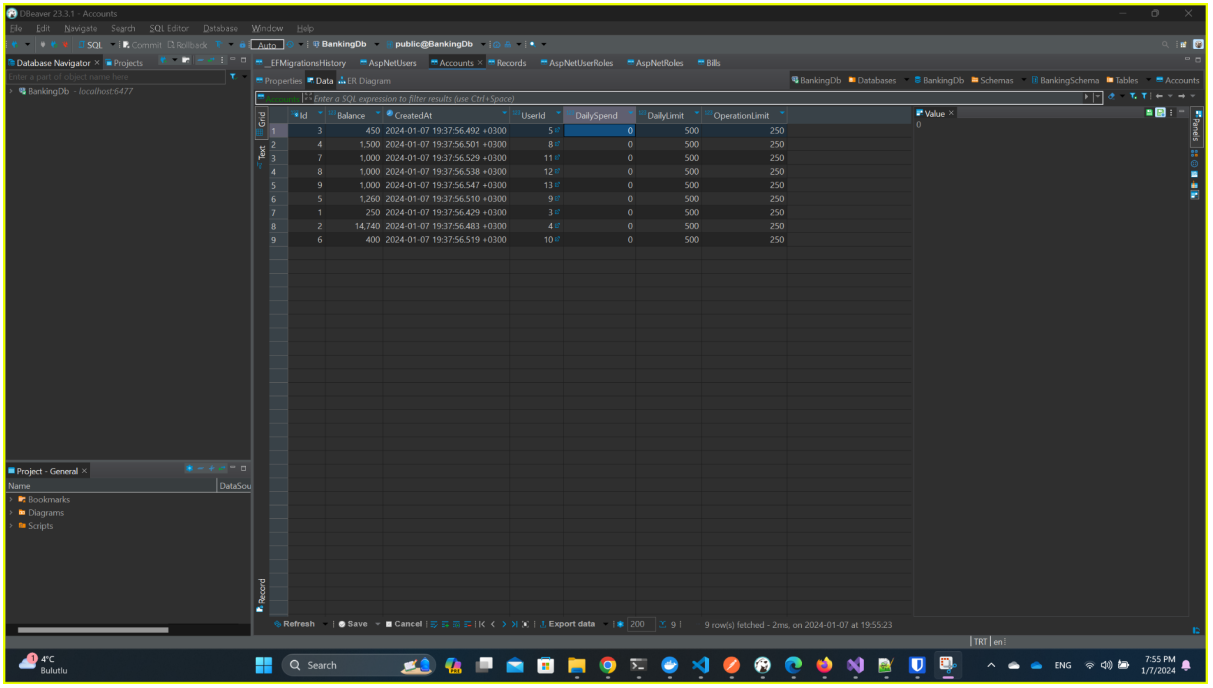


At 19.55, daily spend background service will be executed.

```
1 reference
public static IServiceCollection ConfigureQuartzJobs(this IServiceCollection services)
{
    // Add Quartz services
    services.AddQuartz(q =>
    {
        q.UseMicrosoftDependencyInjectionJobFactory();

        var dailySpendJobKey = JobKey.Create(nameof(DailySpendResetService));
        var payBillsJobKey = JobKey.Create(nameof(PayBillsBackgroundService));

        q.AddJob<DailySpendResetService>(dailySpendJobKey)
            .AddTrigger(trigger =>
                        trigger.ForJob(dailySpendJobKey)
                        .WithSchedule(CronScheduleBuilder.DailyAtHourAndMinute(19, 55)));

         q.AddJob<PayBillsBackgroundService>(payBillsJobKey)
            .AddTrigger(trigger =>
                        trigger.ForJob(payBillsJobKey)
                        .WithSchedule(CronScheduleBuilder.DailyAtHourAndMinute(18, 57)));
    });

    services.AddQuartzHostedService(options =>
    {
        options.WaitForJobsToComplete = true;
    });


    return services;

}       You, 3 days ago • added background service for resetting daily spen…
```

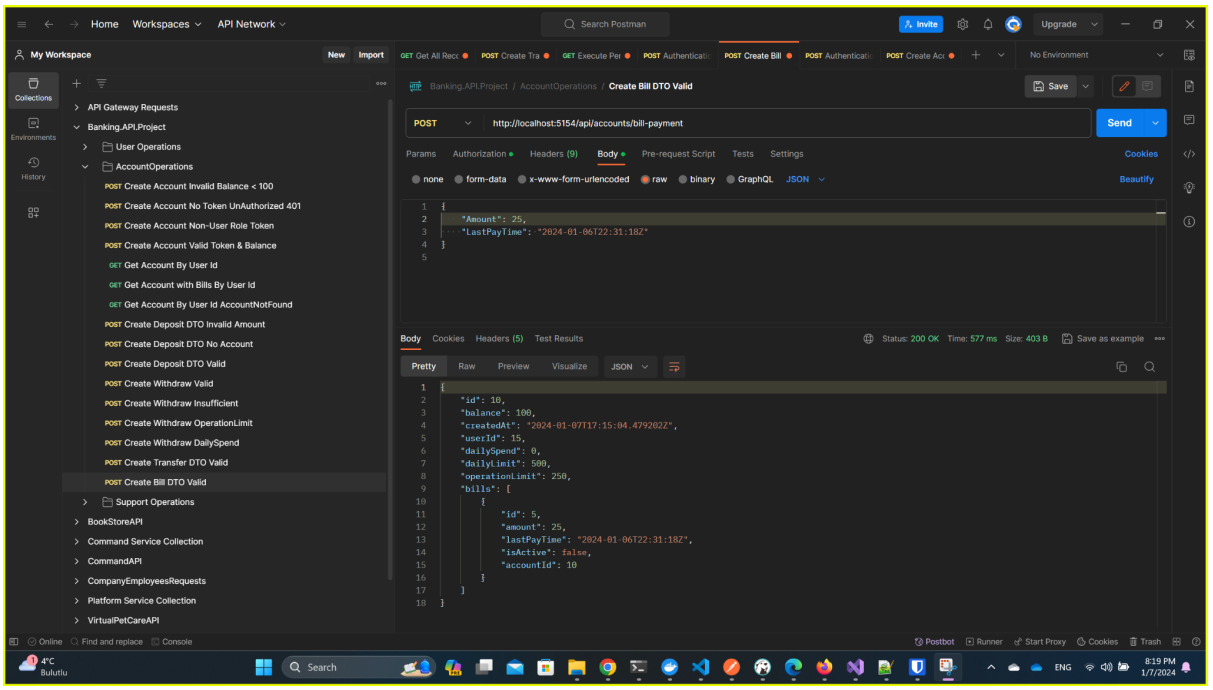


Accounts after execution:

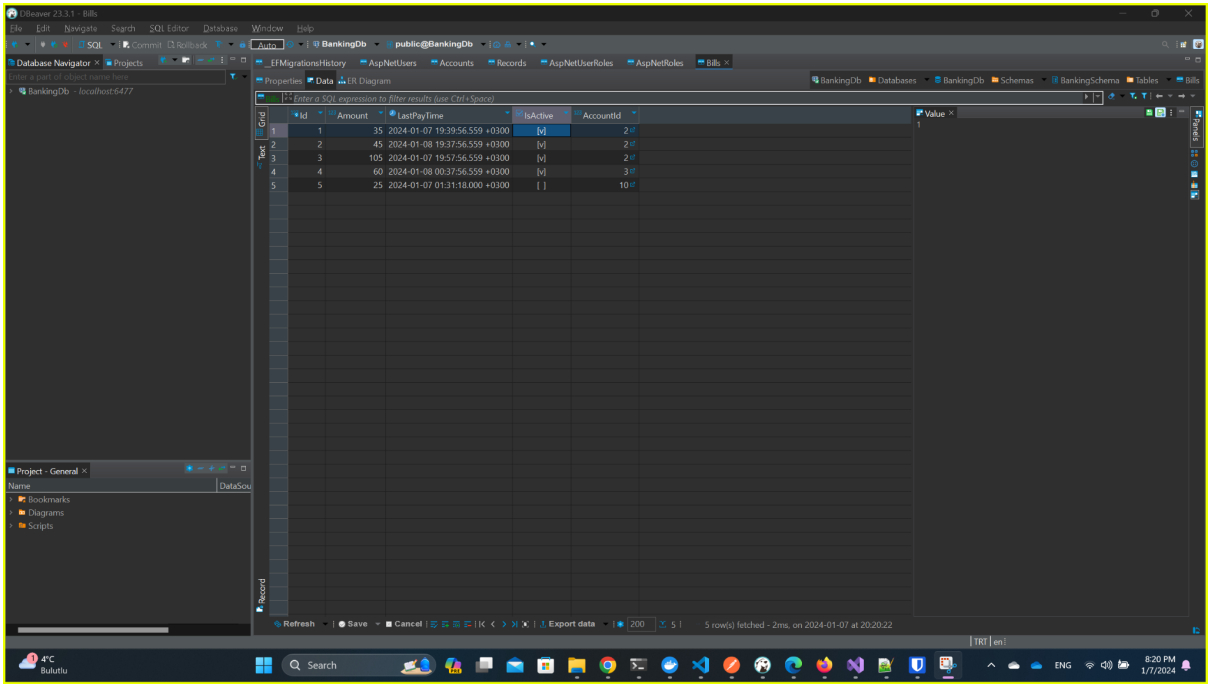


Pay Bills BackgroundService:

Users can add bill.



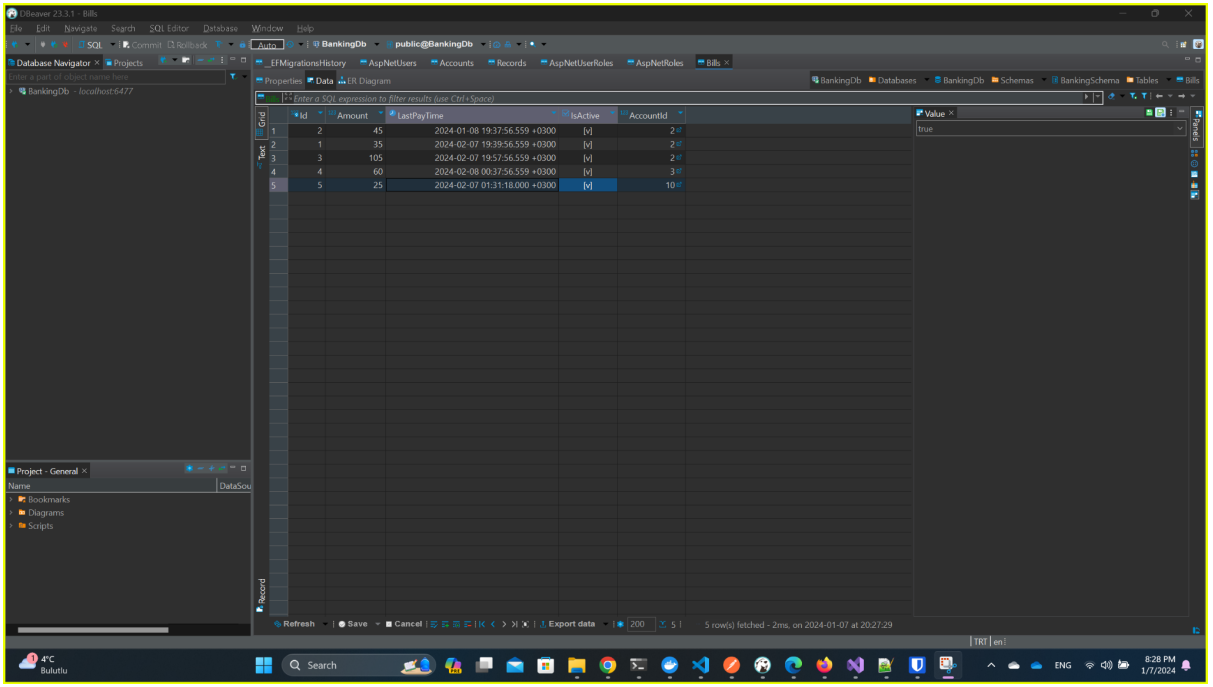
In Bills table:



Firing the background service at 20:27

```
1 reference
public static IServiceCollection ConfigureQuartzJobs(this IServiceCollection services)
{
    // Add Quartz services
    services.AddQuartz(q =>
    {
        q.UseMicrosoftDependencyInjectionJobFactory();

        var dailySpendJobKey = JobKey.Create(nameof(DailySpendResetService));
        var payBillsJobKey = JobKey.Create(nameof(PayBillsBackgroundService));

        q.AddJob<DailySpendResetService>(dailySpendJobKey)
            .AddTrigger(trigger =>
                        trigger.ForJob(dailySpendJobKey)
                        .WithSchedule(CronScheduleBuilder.DailyAtHourAndMinute(19, 55)));

         q.AddJob<PayBillsBackgroundService>(payBillsJobKey)
            .AddTrigger(trigger =>
                        trigger.ForJob(payBillsJobKey)
                        .WithSchedule(CronScheduleBuilder.DailyAtHourAndMinute(20, 27)));
    });
You, 5 seconds ago • Uncommitted changes
    services.AddQuartzHostedService(options =>
    {
        options.WaitForJobsToComplete = true;
    });
```

```
PowerShell   PowerShell   PowerShell
M \"BankingSchema\".\"Bills\" AS b\r\n    WHERE a.\"Id\" = b.\"AccountId\") IS NULL)\r\nORDER BY a.\"Id\""
2024-01-07 20:27:00 [DBG] Account ID: 2, Balance: 14740, DailySpend: 0
2024-01-07 20:27:00 [DBG] Account new balance: 14705
2024-01-07 20:27:00 [DBG] Bill new date: 02/07/2024 16:39:56
2024-01-07 20:27:00 [DBG]   Bill ID: 1, Amount: 35, LastPayTime: 2/7/2024 4:39:56 PM
2024-01-07 20:27:00 [DBG] Account new balance: 14600
2024-01-07 20:27:00 [DBG] Bill new date: 02/07/2024 16:57:56
2024-01-07 20:27:00 [DBG]   Bill ID: 3, Amount: 105, LastPayTime: 2/7/2024 4:57:56 PM
2024-01-07 20:27:00 [DBG] Account ID: 3, Balance: 450, DailySpend: 0
2024-01-07 20:27:00 [DBG] Account new balance: 390
2024-01-07 20:27:00 [DBG] Bill new date: 02/07/2024 21:37:56
2024-01-07 20:27:00 [DBG]   Bill ID: 4, Amount: 60, LastPayTime: 2/7/2024 9:37:56 PM
2024-01-07 20:27:00 [DBG] Account ID: 10, Balance: 100, DailySpend: 0
2024-01-07 20:27:00 [DBG] Account new balance: 75
2024-01-07 20:27:00 [DBG] Bill new date: 02/06/2024 22:31:18
2024-01-07 20:27:00 [DBG]   Bill ID: 5, Amount: 25, LastPayTime: 2/6/2024 10:31:18 PM
2024-01-07 20:27:00 [INF] Executed DbCommand ("6"ms) [Parameters=["@p1='?' (DbType = Int32), @p0='?' (DbType = Int32), @p2='?' (DbType = Object), @p4='?' (DbType = Int32), @p3='?' (DbType = Int32), @p5='?' (DbType = Object), @p7='?' (DbType = Int32), @p6='?' (DbType = Int32), @p8='?' (DbType = Object), @p10='?' (DbType = Int32), @p9='?' (DbType = DateTime), @p12='?' (DbType = Int32), @p11='?' (DbType = DateTime), @p14='?' (DbType = Int32), @p13='?' (DbType = DateTime), @p16='?' (DbType = Int32), @p15='?' (DbType = DateTime), @p17='?' (DbType = Int32), @p18='?' (DbType = Int32), @p19='?', @p20='?' (DbType = Boolean), @p21='?' (DbType = Boolean), @p22='?' (DbType = Int32), @p23='?' (DbType = Int32), @p24='?' (DbType = DateTime), @p25='?' (DbType = Int32), @p26='?' (DbType = Int32), @p27='?' (DbType = Int32), @p28='?', @p29='?' (DbType = Boolean), @p30='?' (DbType = Boolean), @p31='?' (DbType = Int32), @p32='?' (DbType = Int32), @p33='?' (DbType = DateTime), @p34='?' (DbType = Int32), @p35='?' (DbType = Int32), @p36='?' (DbType = Int32), @p37='?', @p38='?' (DbType = Boolean), @p39='?' (DbType = Boolean), @p40='?' (DbType = Int32), @p41='?' (DbType = Int32), @p42='?' (DbType = DateTime), @p43='?' (DbType = Int32), @p44='?' (DbType = Int32), @p45='?' (DbType = Int32), @p46='?', @p47='?' (DbType = Boolean), @p48='?' (DbType = Boolean), @p49='?' (DbType = Int32), @p50='?' (DbType = Int32), @p51='?' (DbType = DateTime), @p52='?' (DbType = Int32)"], CommandType='Text', CommandTimeout='30']"\r\n""
UPDATE \"BankingSchema\".\"Accounts\" SET \"Balance\" = @p0\r\nWHERE \"Id\" = @p1 AND xmin = @p2\r\nRETURNING xmin;\r\nUPDATE \"BankingSchema\".\"Accounts\" SET \"Balance\" = @p3\r\nWHERE \"Id\" = @p4 AND xmin = @p5\r\nRETURNING xmin;\r\nUPDATE \"BankingSchema\".\"Accounts\" SET \"Balance\" = @p6\r\nWHERE \"Id\" = @p7 AND xmin = @p8\r\nRETURNING xmin;\r\nUPDATE \"BankingSchema\".\"Bills\" SET \"LastPayTime\" = @p9\r\nWHERE \"Id\" = @p10;\r\nUPDATE \"BankingSchema\".\"Bills\" SET \"LastPayTime\" = @p11\r\nWHERE \"Id\" = @p12;\r\nUPDATE \"BankingSchema\".\"Bills\" SET \"LastPayTime\" = @p13\r\nWHERE \"Id\" = @p14;\r\nUPDATE \"BankingSchema\".\"Bills\" SET \"LastPayTime\" = @p15\r\nWHERE \"Id\" = @p16;\r\nINSERT INTO \"BankingSchema\".\"Records\" (\"AccountId\", \"Amount\", \"ErrorMessage\", \"IsPending\", \"IsSuccessfull\", \"OperationType\", \"ReceiverAccountId\", \"TimeStamp\", \"UserId\")\r\nVALUES (@p17, @p18, @p19, @p20, @p21, @p22, @p23, @p24, @p25)\r\nRETURNING \"Id\";\r\nINSERT INTO \"BankingSchema\".\"Records\" (\"AccountId\", \"Amount\", \"ErrorMessage\", \"IsPending\", \"IsSuccessfull\", \"OperationType\", \"ReceiverAccountId\", \"TimeStamp\", \"UserId\")\r\nVALUES (@p26, @p27, @p28, @p29, @p30, @p31, @p32, @p33, @p34)\r\nRETURNING \"Id\";\r\nINSERT INTO \"BankingSchema\".\"Records\" (\"AccountId\", \"Amount\", \"ErrorMessage\", \"IsPending\", \"IsSuccessfull\", \"OperationType\", \"ReceiverAccountId\", \"TimeStamp\", \"UserId\")\r\nVALUES (@p35, @p36, @p37, @p38, @p39, @p40, @p41, @p42, @p43)\r\nRETURNING \"Id\";\r\nINSERT INTO \"BankingSchema\".\"Records\" (\"AccountId\", \"Amount\", \"ErrorMessage\", \"IsPending\", \"IsSuccessfull\", \"OperationType\", \"ReceiverAccountId\", \"TimeStamp\", \"UserId\")\r\nVALUES (@p44, @p45, @p46, @p47, @p48, @p49, @p50, @p51, @p52)\r\nRETURNING \"Id\";"
2024-01-07 20:27:00 [DBG] Trigger instruction : NoInstruction
2024-01-07 20:27:00 [DBG] Batch acquisition of 0 triggers
4°C Bulutlu   Search   ENG   8:27 PM 1/7/2024
```



Now LastPayTime has been added one month. Only dates with less than today's has been paid.