## How to configure and install the various components created in the steps

Clone this repository: https://github.com/bean1352/DerivcoAssessment

### Step 1

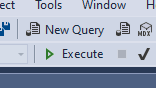
1. Open up SQL Server Management Studio.
2. Connect to the server of your choice.
3. Navigate to .\DerivcoTest\Step1
4. Open the file “SQL\_DatabaseInstallation.sql” with SQL Server Management Studio and click the “Execute” button located in the toolbar.
   1. 

Figure : SQL Execute Button

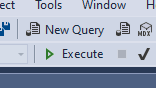
1. You should get a “Command completed successfully” message.
2. Now you have created and populated the tables for the database we will be using to conduct our queries for the test.
3. Open the file “CreateStoredProcedure\_Query.sql” with SQL Server Management Studio and click the “Execute” button located in the toolbar.
   1. 

Figure : SQL Execute Button

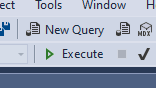
1. You should get a “Command completed successfully” message.
2. Now you have created the stored procedure that we will be using to execute our commands.
3. Open the file “UnitTest\_Queries.sql” with SQL Server Management Studio and click the “Execute” button located in the toolbar.
   1. 

Figure : SQL Execute Button

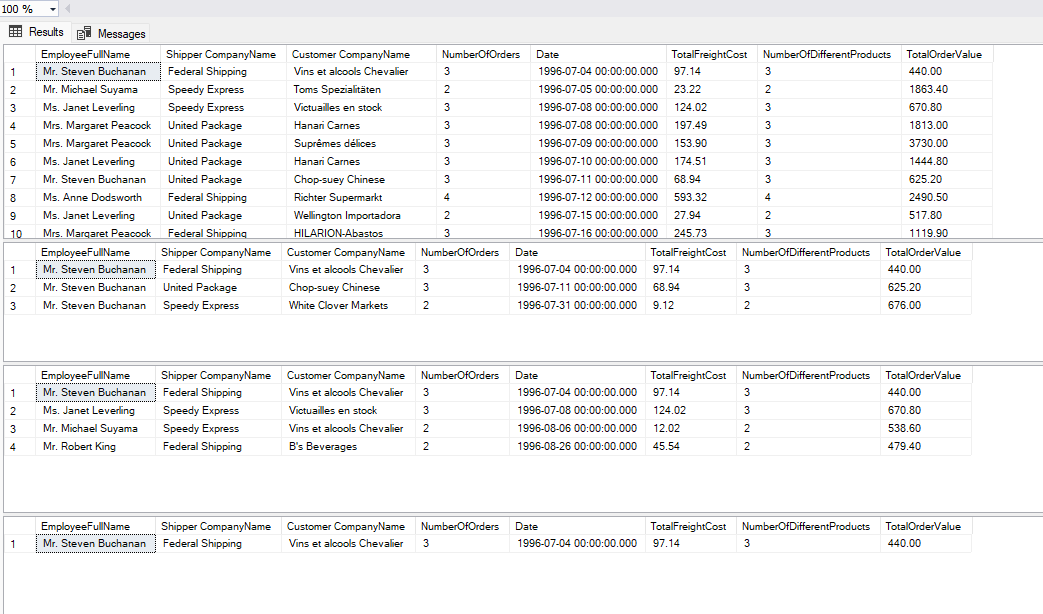
1. You should see four tables in the results tab
   1. 

Figure : SQL Stored Procedure Results

1. These four tables are the results of the four different commands we executed on the stored procedure that we just made on the database we just created and populated.

### Step 2

1. This solution uses .Net 6, SQLite and dapper
2. This application follows the SOLID Principle (“Solid Principles in C# - a Complete Guidance”):

* Single responsibility: Every class has only one responsibility, whether being a business logic class, controller class or model class.
* Open/closed: the classes have been created in such a way that they are open for extension but closed for modification. The classes are scalable and don’t need to be modified unless new functionality needs to be added.
* Liskov substitution: subclasses incorporate all classes of their base class
* Interface segregation: Code is very modular as it is broken down into small methods
* Dependency inversion: Abstraction between high level and low level classes

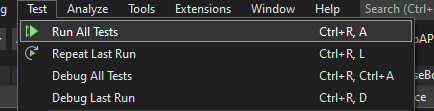
1. Requests are not thread blocking.
2. JSON Return type.
3. Integrated Unit Tests
4. Global exception handling
5. Dependency Injection Design Pattern
6. How to configure and run the solution
   1. Navigate to .\DerivcoTest\Step2 and open the DerivcoWebAPI folder with Visual Studio.
   2. Wait for the solution to build.
   3. To run all tests, click the “Test” drop down in the toolbar and select “Run All Tests”
      1. 

Figure : Web API Run Tests Button

* 1. All tests should run successfully.
  2. Now you can start the application by clicking the “Play Button”
     1. 

Figure : Web API Play Button

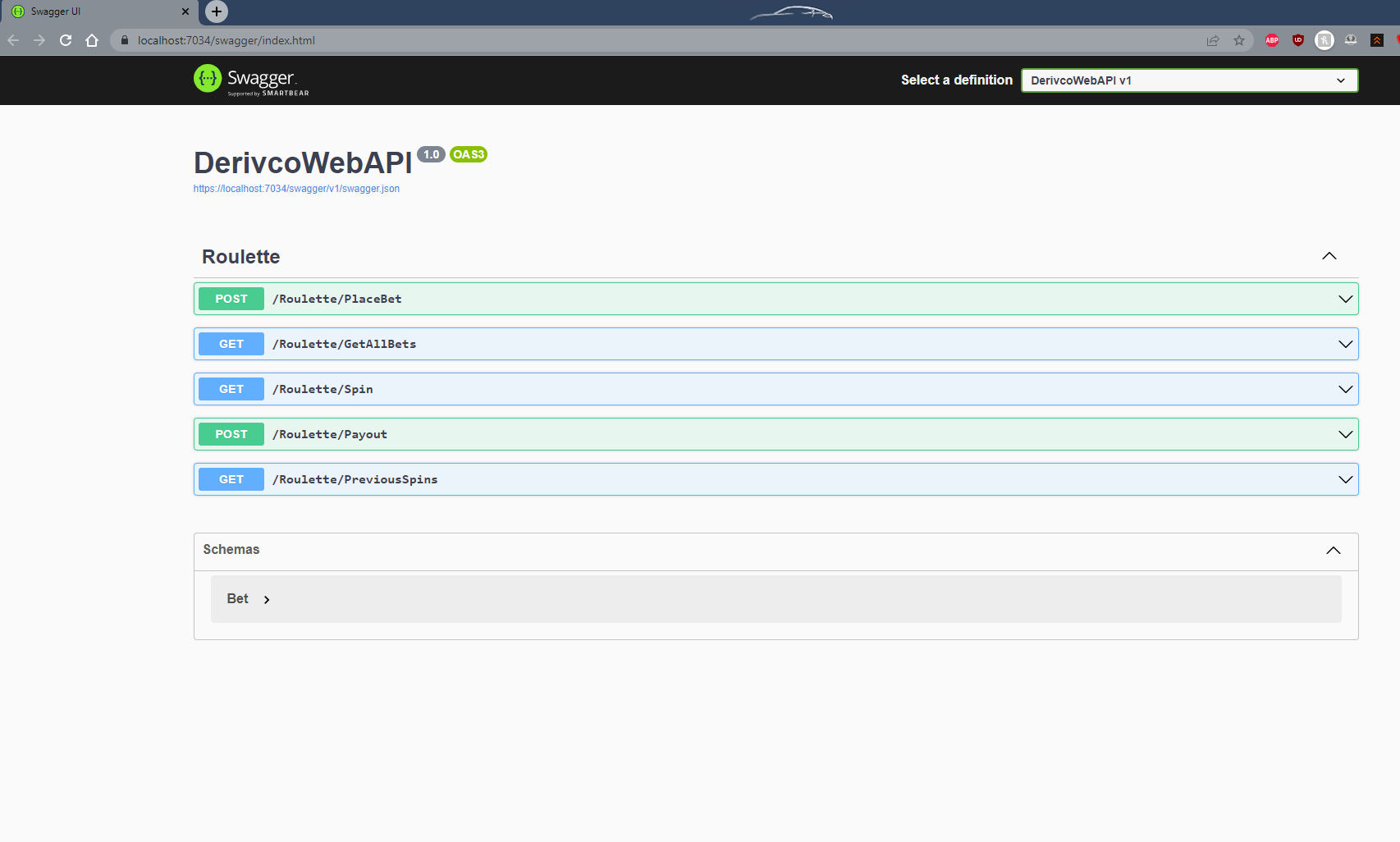
* 1. This should bring you to a locally hosted web page with a swagger API interface
     1. 

Figure : Swagger API Interface

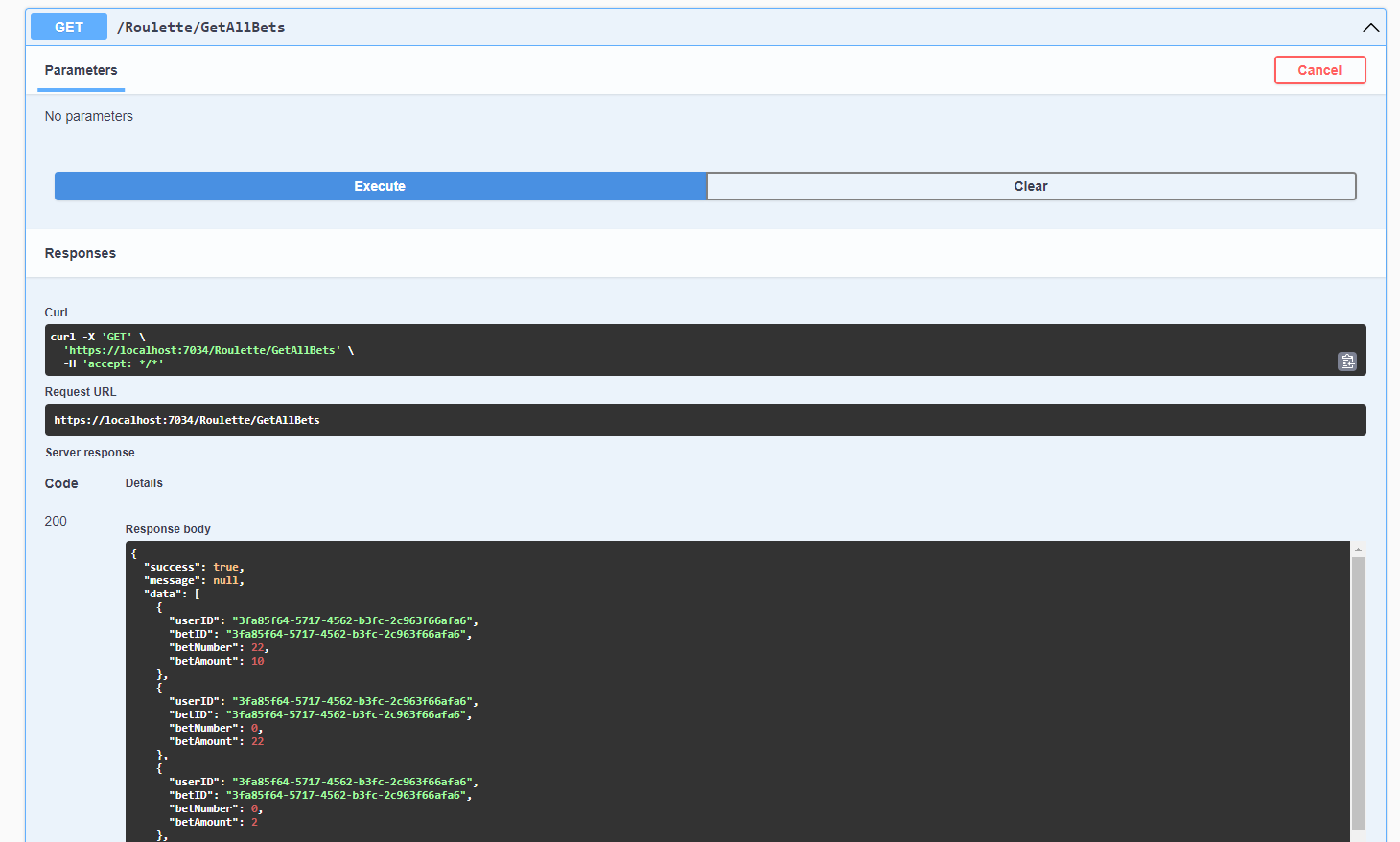
* 1. You can make requests by clicking on any endpoint and click the “Try it out” button
     1. 

Figure : Swagger API Requests

* 1. Alternatively, you can use an API platform such as PostMan to send your requests.

## References

Solid Principles in C# - a Complete Guidance.” InterviewBit, 27 May 2022, [www.interviewbit.com/blog/solid-principles-in-c-sharp/](http://www.interviewbit.com/blog/solid-principles-in-c-sharp/). Accessed 20 Jan. 2023.