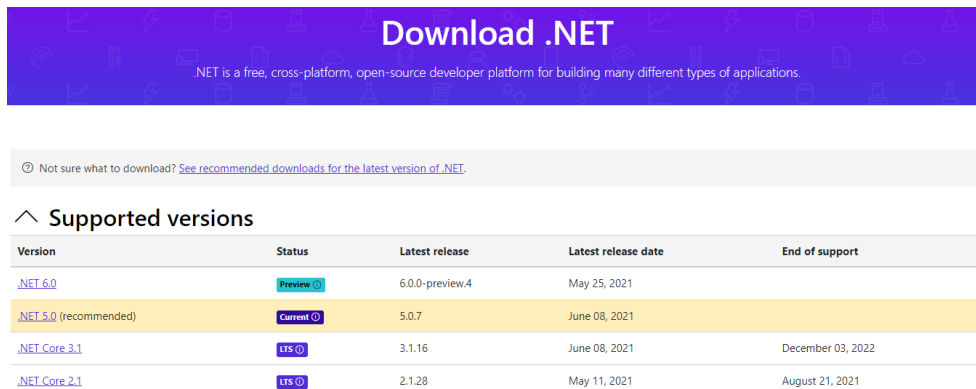


Expectations/Assumptions

1. We expect knowledge of programming in C#/TypeScript/OOP, but do not expect knowledge of 3rd party libraries (e.g. Dapper, moment, etc.). Please ask if something is unfamiliar.
2. In the interest of time, all data inputs will be valid, and no database connectivity errors will occur.
3. Be prepared to explain the code you write for this activity.

Prerequisites:

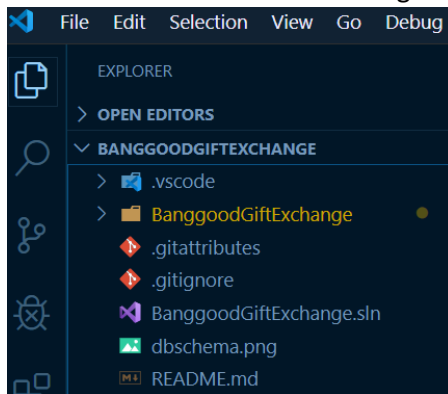
1. Install **Visual Studio Code** for your operating system (<https://code.visualstudio.com/>)
2. Install version 5.0 of the **.NET SDK**, the runtime isn't necessary (<https://dotnet.microsoft.com/download/dotnet-core>)



The screenshot shows the 'Download .NET' page. It features a purple header with the text 'Download .NET' and a sub-header '.NET is a free, cross-platform, open-source developer platform for building many different types of applications.' Below this is a section titled 'Supported versions' which contains a table with the following data:

Version	Status	Latest release	Latest release date	End of support
.NET 6.0	Preview	6.0.0-preview.4	May 25, 2021	
.NET 5.0 (recommended)	Current	5.0.7	June 08, 2021	
.NET Core 3.1	LTS	3.1.16	June 08, 2021	December 03, 2022
.NET Core 2.1	LTS	2.1.28	May 11, 2021	August 21, 2021

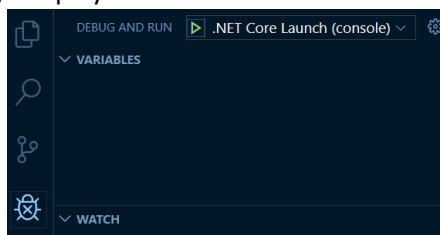
- a. Install the Visual Studio Code extension called **SQLite**.
3. Unlike Visual Studio where programs are opened from their solution file, open this program in Visual Studio Code from the top-level folder. In this case, it would be the folder created as a result of unzipping the file. Your file explorer should look like the following image. If it doesn't, you may run into issues when debugging the app.



a.

Running the app:

1. To run the app, click the debug tab in Visual Studio Code and select the ".NET Core Launch (console)" setting. Clicking the play button should build and run the app.



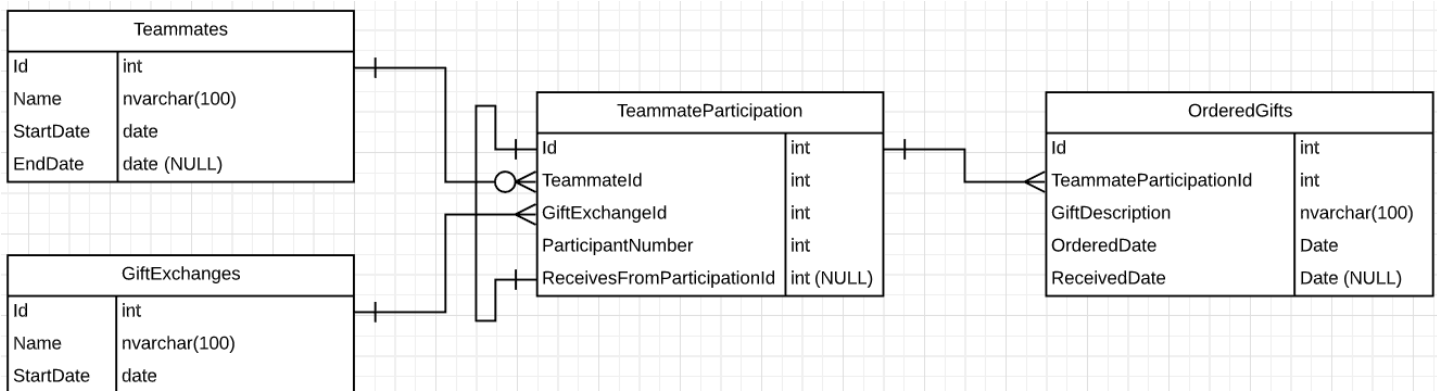
a.

2. If you are prompted to restore dependencies, click the action provided to restore them. This installs any nuget packages used by the program.
3. For portability, the app uses a SQLite database contained in BanggoodDb.sqlite. If you need to reset data, delete this file and restart the app – there is code that will recreate the database if that file isn't found
 - a. To run queries against this database, use Ctrl-Shift-P to open the task selector, enter SQLite, and choose "SQLite: Use Database". Select the BanggoodDb.sqlite file.
 - b. Open QueryRunner.sql to type your query. Hit Ctrl-Shift-Q to run the query or open the task selector and choose "SQLite: Run Query".
 - c. SQLite syntax may differ slightly from T-SQL (selects, filters, and joins are the same). Check out the SQLite [cheat sheet](#) to find any syntax differences you may need to use.

App Tour

1. This console app was created using the standard .NET Core Console app project. The console printing and command handling all happens in Program.cs. You should not need to change this file.
2. The app has the following existing classes:
 - a. BanggoodService.cs should contain all business logic. It has an instance of BanggoodDbService for retrieving/saving data in the database.
 - b. BanggoodDbService.cs should contain all code for retrieving/saving data in the database. The class uses an extension NuGet package called [Dapper](#) that runs raw SQL queries, supports SQL parameters, and can map properties to and from objects.
 - c. There are a few pre-existing classes representing the database tables in the Models folder. You are welcome to add new models or change existing ones if needed.

Database Schema



1. A teammate's EndDate will be null if they are still employed with the company
2. TeammateParticipation represents a teammate participating in a gift exchange. Each participant is assigned a sequential ParticipantNumber to be used in determining package recipients (see requirements and example).
3. ReceivesFromParticipationId is used to track which other participant a teammate receives gifts from. It won't have a value until the sender/recipient pairing is determined
4. Each participant will order one or more gifts as tracked in the OrderedGifts table. Gifts that haven't arrived yet will have a null value for ReceivedDate.
5. The database initialized by the app contains data for 2 gift exchanges for testing and debugging.

Requirements

A Banggood Gift Exchange is becoming an annual tradition for the BHTP IT department. Banggood is a company that sells an assortment of oddities that are all shipped from China. Due to US Customs, the shipping process can take a few weeks to several months. We gamified this shipping process as a twist on the traditional white elephant game with the basic rules below:

- Each teammate is assigned a unique, sequential participant number. Items are ordered from Banggood using this assigned number in the shipping address (e.g. #1 IT BGE, #2 IT BGE)
- Packages typically arrive in an unpredictable order. Teammates receive packages in order of their assigned number
 - e.g. teammate #1 will receive all packages with the participant number (as determined by the shipping address) from the first package to arrive
 - A teammate can receive their own gifts

The team would like an app to track shipping assignments and package arrivals with the following functionality:

1. Complete the CreateGiftExchange method in the BanggoodService to generate a new Banggood Gift Exchange and store it in the database
 - a. Method is passed the following data
 - i. name for the Banggood Gift Exchange (e.g. "2019 Holiday BGE")
 - ii. Start date of the new Banggood Gift Exchange
 - b. Method should do the following
 - i. Sequentially (starting at 1) assign participant numbers to teammates who are employed on the date provided (check your resulting data to make sure only eligible teammates are included).
 1. A randomly sorted list of all teammates is already provided
 2. Teammates who have an end date after the gift exchange start date are eligible to participate (e.g. a teammate in a 2-week notice period could still participate)
 - ii. Store the generated GiftExchange and TeammateParticipation records in the database
 - iii. Return the generated list of participant numbers with the names of the teammates eligible to participate
2. Complete the HandleGiftArrival method by marking the gift as received and determining which teammate receives that gift
 - a. Method should accept the Id of the OrderedGift record that arrived
 - b. The gift should be marked as received by setting the ReceivedDate on the OrderedGifts record
 - c. Assign the gift to the correct team member by filling in the ReceivesFromParticipationId on the receiving member's participation record.
 - i. The receiving member is determined by the gift sender's ParticipantNumber—if this is the first gift from a ParticipantNumber, use the count of unique ParticipantNumbers that have a gift that has arrived. Assign it to the team member whose ParticipationNumber is next in line. See "Example Exchange" for further clarification of this.
 - d. Return the name of the recipient even if they have already been determined.
3. Complete the query in RecapAllGiftsForExchange by returning the description of each gift, the name of the team member who sent it, the name of the team member who received it, and the received date. Filter by a specific gift exchange and return in the order that gifts were received.

Example Exchange

Teammates in Exchange (ParticipantNumber in parenthesis)

Andy (1)

Josh (2)

Erin (3)

OrderedGifts

Andy – Newton’s Cradle, Nesting Dolls

Josh – Bathroom Décor, Solar Lights, Keychain

Erin – Wind-up Toy, Chessboard, Fidget Spinner

- 4 weeks after ordering gifts the first one arrives. It’s the Chessboard and it goes to Andy because his ParticipantNumber is 1. Andy will receive all future gifts arriving from Erin. Andy’s TeammateParticipation record would reference Erin’s TeammateParticipation record using the ReceivesFromParticipationId foreign key.
- 2 weeks later the Nesting Dolls arrive. They go to Josh because his ParticipantNumber is 2 and this is the second unique sender to have gifts arrive from. Josh will receive all future gifts from Andy in this gift exchange
- The Fidget Spinner arrives on the next day. Despite this being the third gift to arrive, it still goes to Andy because he has already received a different gift from the Fidget Spinner’s sender (Erin)
- A week later the Keychain arrives. This is the third unique sender ParticipantNumber to arrive, so it goes to Erin. Erin will receive all future gifts from Josh in this gift exchange.
- All recipients are paired with a sender at this point. As gifts arrive, they will continue going from the paired sender to receiver